1326911

पंजीकरण सं. Registration No.: M117/45251/0015

केन्द्रीय माध्यमिक शिक्षा बोर्ड
Central Board of Secondary Education

माध्यमिक विद्यालय परीक्षा (वर्ष : 2017)
SECONDARY SCHOOL EXAMINATION (YEAR : 2017)

कक्षा X ग्रेड शीट सह निष्पादन प्रमाण पत्र
Class X Grade Sheet cum Certificate of Performance

यह प्रमाणित किया जाता है कि This is to Certify that **ROHAN CHAUHAN**

अनुक्रमांक Roll No.: **4177596**

माता का नाम Mother's Name **NEELAM CHAUHAN**

पिता/संरक्षक का नाम Father's / Guardian's Name **RAJESH CHAUHAN**

जन्म तिथि Date of Birth **14/10/2001 14TH OCTOBER TWO THOUSAND ONE**

विद्यालय School **45251-GURUKUL ENGLISH MEDIUM SCHOOL YAVATMAL MR**

का निष्पादन निम्नानुसार रहा has performed as follows :

शैक्षणिक निष्पादन – कक्षा X Academic Performance - Class X शैक्षिक क्षेत्र Scholastic Areas

विषय कोड तथा नाम Subject Code and Name	Formative Assessment (FA) Grade	Summative Assessment (SA) Grade	Overall Grade (FA+SA)	
			Grade	Grade Point (GP)
101 ENGLISH COMM.	A1	A2	A1	10
085 HINDI COURSE-B	A1	A2	A1	10
041 MATHEMATICS	A1	A2	A1	10
086 SCIENCE	A1	A2	A1**	10
087 SOCIAL SCIENCE	A1	A1	A1	10
462 INFO TECHNOLOGY(C)	A1	A2	A1	10

संचित ग्रेड बिन्दु का औसत (सीजीपीए) Cumulative Grade Point Average (CGPA) : **10.0**

* कथन और श्रवण कौशलों (एएसएल) के आकलन में ग्रेड Grade in Assessment of Speaking and Listening Skills (ASL) **A1**

** सह-शैक्षिक क्षेत्र में निष्पादन के अनुसार उन्नत ग्रेड Upgraded Grade based on performance in the Co-Scholastic areas of the candidate

परिणाम Result : **QUALIFIED FOR ADMISSION TO HIGHER CLASSES**

दिल्ली Delhi
दिनांक Dated 03-06-2017

परीक्षा नियंत्रक
Controller of Examinations

1 Foundations of Deep Learning

Unit I

Syllabus

What is machine learning and deep learning? Supervised and Unsupervised Learning, bias variance trade off, hyper parameters, under/over fitting regularization, Limitations of machine learning, History of deep learning, Advantage and challenges of deep learning. Learning representations from data , Understanding how deep learning works in three figures, Common Architectural Principles of Deep Network, Architecture Design, Applications of Deep learning, Introduction and use of popular industry tools such as TensorFLow, Keras, PyTorch, Caffe, Shogun.

1.1 What is Machine Learning and Deep Learning ?

1.1.1 Machine Learning

- Machine learning has its roots in the field of artificial intelligence (AI), which aims to create intelligent machines that can perform tasks that normally require human intelligence, such as learning, problem solving, and decision making.
- The need for machine learning arose because there are many tasks that are too complex or time-consuming for humans to perform manually, but that computers can do quickly and accurately. For example, it would be impractical for a human to analyze millions of data points to make a prediction or decision, but a machine learning algorithm can do this in a matter of seconds.
- Overall, machine learning has come into existence as a way to enable computers to learn from data and make intelligent decisions, in order to solve complex problems and improve efficiency in a wide range of industries.
- Machine learning is a method of teaching computers to learn from data, without being explicitly programmed. It involves feeding a computer system a large amount of data and allowing it to learn patterns and features in the data, so that it can make intelligent decisions or predictions on new data.

  Here are a few more examples of how machine learning is being used today:

  - **Natural language processing (NLP) :** Machine learning is used to teach computers to understand, interpret, and generate human language. This has applications in language translation, language generation, and language understanding for virtual assistants and chatbots.
  - **Healthcare :** Machine learning is being used in healthcare to predict patient outcomes, identify potential outbreaks of infectious diseases, and help with diagnosis and treatment planning.
  - **Agriculture :** Machine learning is being used to optimize crop yields and improve resource efficiency in agriculture.

- **Self-driving cars :** Machine learning is a key technology behind self-driving cars, allowing them to make intelligent decisions about how to navigate roads and avoid accidents.
- **Cybersecurity :** Machine learning is being used to identify and prevent cyber threats, such as malware and phishing attacks.

- It is used to analyze patterns in email content and sender information to determine whether an email is spam or not.
- As we can see, machine learning has a wide range of applications in many different industries, and is being used to solve a variety of problems.
- These are just a few examples of how machine learning is being used today. It is a rapidly growing field with many applications in a wide range of industries.

1.1.2 Machine Learning Process

- The machine learning process is a general framework for building machine learning models. It typically consists of the following steps:

1. **Data Collection :** The first step in the machine learning process is to collect the data that will be used to train the model. This may involve collecting data from various sources or creating synthetic data.
2. **Data Preparation :** Once the data has been collected, it needs to be pre-processed and cleaned to remove any noise or inconsistencies. This may involve steps such as data normalization, missing value imputation, and feature selection.
3. **Model Selection :** The next step is to select an appropriate machine learning model for the task at hand. This will depend on the type of problem being addressed, the amount and quality of data available, and other factors.
4. **Model Training :** After selecting a model, the next step is to train the model using the prepared data. During training, the model is optimized to minimize a chosen loss function, typically by adjusting the model's parameters.
5. **Model Evaluation :** Once the model has been trained, it needs to be evaluated to determine its performance. This is typically done by measuring the model's accuracy or some other performance metric using a held-out validation set or through cross-validation.
6. **Model Tuning :** If the model's performance is not satisfactory, the next step is to tune the model's hyperparameters. This may involve adjusting parameters such as learning rate, regularization strength, and the number of hidden units in the model.
7. **Deployment :** Finally, once the model has been trained and tuned, it can be deployed in a production environment to make predictions on new data.

- This is a high-level overview of the machine learning process, and the specific details of each step may vary depending on the problem and the available resources. However, these steps provide a general framework for building and deploying machine learning models

1.1.3 Applications of Machine Learning

- Machine learning has a wide range of applications across various industries and domains. Here are some examples of the applications of machine learning:

  1. **Image and speech recognition :** Machine learning is used in image and speech recognition systems, such as facial recognition, voice recognition, and natural language processing. This technology is used in applications such as virtual assistants, automated customer support, and self-driving cars.
  2. **Fraud detection :** Machine learning is used to detect fraud in financial transactions and other industries, such as insurance and healthcare. It can identify patterns in large volumes of data to detect anomalies and prevent fraudulent activities.
  3. **Personalized marketing :** Machine learning is used to personalize marketing messages and offers for individual customers. It can analyze customer data to identify patterns and preferences and tailor marketing messages to specific segments of customers.
  4. **Healthcare :** Machine learning is used in healthcare to assist in diagnosis, treatment, and drug discovery. It can help to identify risk factors for diseases, predict patient outcomes, and optimize treatment plans.
  5. **Predictive maintenance :** Machine learning is used to predict equipment failures and prevent downtime in manufacturing and other industries. It can analyze data from sensors and other sources to identify patterns and predict when equipment is likely to fail.
  6. **Financial analysis :** Machine learning is used in finance to analyze market data, predict trends, and make investment decisions. It can help to identify patterns in market data and optimize investment strategies.
  7. **Transportation :** Machine learning is used in transportation to optimize routes, predict demand, and reduce congestion. It can help to optimize logistics and reduce travel time.

- These are just a few examples of the many applications of machine learning. As the technology continues to advance, we can expect to see many more applications in a wide range of industries and domains.

1.1.4 Deep Learning

- Deep learning is a subset of a more general field of artificial intelligence called machine learning, which is predicated on the idea of learning from example.
- Deep learning involves training artificial neural networks on large datasets, allowing the network to learn and make intelligent decisions on its own.

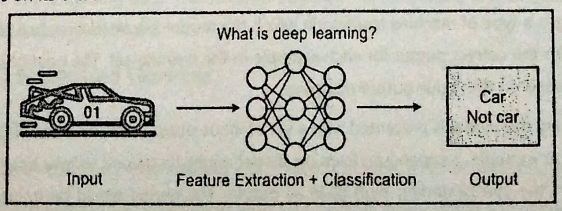


Fig. 1.1.1 : Deep learning

- As above Fig. 1.1.1 depicts deep learning algorithms can extracts features by themselves which is not the case with machine learning.

- Deep learning has proven to be a powerful tool for tasks that require the ability to learn from large amounts of unstructured data, and has achieved state-of-the-art results in many fields.
- However, it is important to note that deep learning is not always the best choice, and that traditional machine learning methods can still be effective in some cases.

1.1.5 Machine Learning Vs. Deep Learning

- Machine learning and deep learning are both subfields of artificial intelligence, but they differ in how they process and learn from data.
- Machine learning involves developing algorithms that can learn from data to make predictions or decisions. These algorithms can be trained on large datasets using various techniques such as linear regression, decision trees, and support vector machines. Machine learning algorithms can be used for a wide range of tasks, including image classification, speech recognition, and natural language processing.
- Deep learning is a subset of machine learning that uses neural networks with many layers to learn from data. These neural networks are inspired by the structure of the human brain and can learn complex patterns and relationships in data. Deep learning is especially useful for tasks such as image and speech recognition, natural language processing, and playing games.
- The main difference between machine learning and deep learning is the complexity of the models and the amount of data required to train them. Machine learning algorithms typically require less data and are easier to interpret, while deep learning algorithms require more data and computational resources and can be more difficult to interpret. However, deep learning can often achieve higher levels of accuracy on complex tasks and is currently the state-of-the-art approach for many applications.
- In summary, machine learning is a broad field that encompasses a wide range of techniques for learning from data, while deep learning is a specific subset of machine learning that uses neural networks with many layers to learn complex patterns and relationships in data

1.2 Supervised Learning and Unsupervised Learning

Machine learning is the capacity of a machine to automatically learn from data. Based on how this learning is happening, it can be Supervised or unsupervised learning.

1.2.1 Supervised Learning

- Supervised learning is a type of machine learning in which the model is trained on labelled data, meaning that the data is provided with the correct output for each example in the training set. The goal of supervised learning is to make predictions based on this input-output mapping.
- In supervised learning, the model is presented with a set of input-output pairs and must learn a function that maps inputs to outputs. For example, a supervised learning model might be trained to take images of handwritten digits as input and predict the corresponding digit (0-9) as output. The model would be trained on a large dataset of labelled images, and would learn to recognize patterns and features in the images that are correlated with the correct output.

- Once the model has been trained, it can be used to make predictions on new, unseen data. For example, given a new image of a handwritten digit, the model would predict which digit it represents.
- Some common applications of supervised learning include image classification, speech recognition, and natural language processing.
- The below Fig. 1.2.1 shows the difference in learning manner of student which is similar to learning from data supervised and unsupervised environment.

1.2.2 How Supervised Learning Work ?

- Supervised learning is a type of machine learning where the algorithm learns to predict a target variable based on labelled examples or input-output pairs in the training data. The goal of supervised learning is to build a model that can make accurate predictions on new, unseen data.
- The process of supervised learning typically involves the following steps:

1. **Data Preparation :** The first step is to collect and prepare the data for training the model. This involves cleaning the data, selecting the relevant features, and splitting the data into training and test sets.
2. **Model Selection :** The next step is to select an appropriate machine learning algorithm for the task at hand. This may involve choosing between different types of algorithms, such as linear regression, decision trees, or neural networks.
3. **Model Training :** After selecting an algorithm, the next step is to train the model on the labelled training data. The model is optimized to minimize a chosen loss function, typically by adjusting the model's parameters.
4. **Model Evaluation :** Once the model has been trained, it needs to be evaluated to determine its performance. This is typically done by measuring the model's accuracy or some other performance metric using a held-out validation set or through cross-validation.
5. **Model Tuning :** If the model's performance is not satisfactory, the next step is to tune the model's hyperparameters. This may involve adjusting parameters such as learning rate, regularization strength, and the number of hidden units in the model.
6. **Deployment :** Finally, once the model has been trained and tuned, it can be deployed in a production environment to make predictions on new data.

- In summary, supervised learning works by training a model to predict a target variable based on labelled examples or input-output pairs in the training data. The model is optimized to minimize a loss function, and its performance is evaluated on a held-out validation set.

1.2.3 Advantages of Supervised Learning

Supervised learning has several advantages :

1. Supervised learning can be used for a wide range of tasks, including regression, classification, and prediction.
2. Supervised learning algorithms can be easily evaluated using performance metrics such as accuracy, precision, and recall.

3. Supervised learning can be more accurate than unsupervised learning methods since it makes use of labelled data to learn from.
4. Supervised learning models can be used to make predictions on new data that was not seen during training.
6. Supervised learning is a well-studied and widely-used approach, with many available algorithms and libraries.

1.2.3 Disadvantages of Supervised Learning

Supervised learning has several disadvantages :

1. Supervised learning requires labelled data, which can be time-consuming and expensive to collect and label.
2. The performance of supervised learning models is highly dependent on the quality and representativeness of the training data.
3. Supervised learning models can be prone to overfitting if the model is too complex or if the training data is too small or not diverse enough.
4. Supervised learning models can also be sensitive to outliers in the training data.
5. The choice of algorithm and hyperparameters can be challenging and requires some expertise.

- In summary, supervised learning has advantages and disadvantages, and its suitability for a given task depends on the availability and quality of labelled data, the complexity of the problem, and the performance requirements.

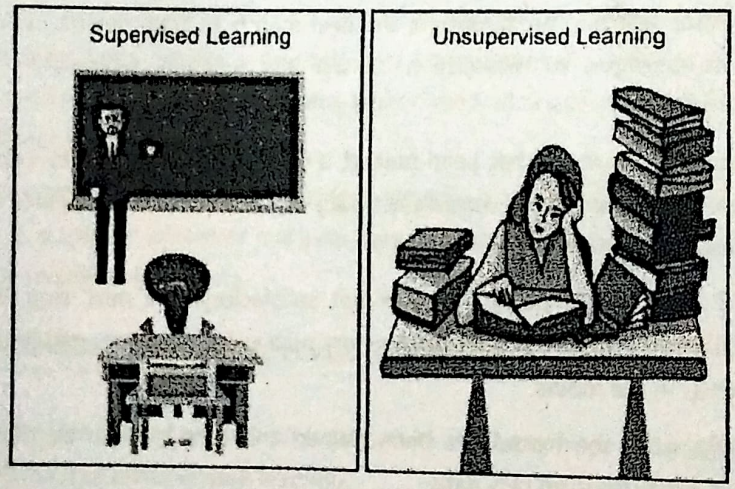


Fig. 1.2.1 : Supervised learning and Unsupervised learning

1.2.4 Unsupervised Learning

- This is a type of machine learning in which the model is not provided with labelled training examples. Instead, the goal of unsupervised learning is to find patterns and relationships in the data.
- In unsupervised learning, the model is given a dataset and must discover the inherent structure of the data by itself. This is in contrast to supervised learning, where the model is given both input data and corresponding labelled output data.
- One common approach to unsupervised learning is clustering, in which the model groups the data into clusters based on some measure of similarity. For example, a clustering algorithm might group customers together based

1.2.4(A) How Unsupervised Learning Work ?

- Unsupervised learning is a type of machine learning where the algorithm learns to identify patterns and structure in unlabelled data without any specific guidance or feedback. Unlike supervised learning, unsupervised learning does not have a target variable to predict or learn from.
- The process of unsupervised learning typically involves the following steps:
  1. **Data Preparation :** The first step is to collect and prepare the data for analysis. This involves cleaning the data, selecting the relevant features, and potentially reducing the dimensionality of the data.
  2. **Model Selection :** The next step is to select an appropriate unsupervised learning algorithm for the task at hand. This may involve choosing between different types of algorithms, such as clustering, dimensionality reduction, or anomaly detection.
  3. **Model Training :** After selecting an algorithm, the next step is to train the model on the unlabelled data. The model is optimized to identify patterns, structure, or outliers in the data.
  4. **Model Evaluation :** Unlike supervised learning, unsupervised learning does not have a target variable to evaluate performance. Instead, the performance of an unsupervised learning model is typically evaluated based on its ability to identify meaningful patterns or structure in the data.
  5. **Model Tuning :** If the model's performance is not satisfactory, the next step is to tune the model's hyperparameters. This may involve adjusting parameters such as the number of clusters or the dimensionality of the data.
  6. **Deployment :** Finally, once the model has been trained and tuned, it can be used to explore and analyze new data.
- In summary, unsupervised learning works by identifying patterns and structure in unlabelled data without any specific guidance or feedback. Unsupervised learning can be useful for exploratory data analysis, data visualization, and discovering hidden structure in data.

1.2.4(B) Advantages of Unsupervised Learning

1. Unsupervised learning can be used for exploratory data analysis and to uncover hidden structure in the data.
2. Unsupervised learning algorithms can identify patterns in the data that may not be immediately apparent to a human analyst.
3. Unsupervised learning can be used to cluster data points into groups or identify anomalies in the data.
4. Unsupervised learning does not require labelled data, which can be time-consuming and expensive to collect and label.
5. Unsupervised learning can be used to pre-process data before supervised learning is applied, which can improve the performance of supervised learning models.

1.2.4(C) Disadvantages of Unsupervised Learning

1. Unsupervised learning can be difficult to evaluate since there is no target variable to predict or learn from.
2. Unsupervised learning algorithms can be sensitive to the quality and representativeness of the input data.

3. Unsupervised learning algorithms can be difficult to tune since there are typically no explicit performance metrics to optimize for.
4. Unsupervised learning algorithms can be computationally expensive and may require significant computational resources to train on large datasets.
5. Unsupervised learning algorithms can be prone to identifying spurious or irrelevant patterns in the data.

In summary, unsupervised learning has advantages and disadvantages, and its suitability for a given task depends on the nature of the data, the complexity of the problem, and the performance requirements.

1.2.4(D) Comparison between Supervised Learning and Unsupervised Learning

- Supervised and unsupervised learning are two broad categories of machine learning approaches with distinct characteristics. Here are some key differences between the two:
  1. **Labelled vs. Unlabelled Data :** The most fundamental difference between supervised and unsupervised learning is that supervised learning requires labelled data, while unsupervised learning works with unlabelled data.
  2. **Task :** Supervised learning is used for tasks that involve predicting a target variable or classifying data into predefined categories. Unsupervised learning is used for tasks that involve discovering patterns or structure in the data.
  3. **Feedback :** Supervised learning algorithms receive feedback on the correctness of their predictions, while unsupervised learning algorithms do not receive explicit feedback.
  4. **Training :** Supervised learning algorithms train on labelled data, while unsupervised learning algorithms train on unlabelled data.
  5. **Evaluation :** Supervised learning algorithms are evaluated based on how accurately they predict the target variable on unseen data, while unsupervised learning algorithms are evaluated based on how well they identify patterns and structure in the data.
  6. **Complexity :** Supervised learning can be more complex than unsupervised learning since it involves learning a mapping from input features to a target variable. Unsupervised learning can be less complex since it does not require a specific output.
  7. **Performance :** Supervised learning can be more accurate than unsupervised learning since it makes use of labelled data to learn from. However, unsupervised learning can be more flexible and can be used for a wide range of tasks.
- In summary, supervised and unsupervised learning are two distinct approaches to machine learning with different strengths and weaknesses. The choice of approach depends on the specific task, the available data, and the performance requirements

1.3 Bias Variance Trade-off

- In supervised machine learning, a model is created by an algorithm using training data. When we discuss the machine learning model, we actually discuss the model's performance and accuracy, which are referred to as prediction errors.

- In machine learning, error is used to measure the precision with which a model can forecast outcomes based both on previously learned data and newly available data. We select the machine learning model that performs the best for a specific dataset based on our error.
- Our model will examine our data and look for patterns before making predictions. We can draw conclusions about specific cases in our data using these patterns. Following training, our model picks up on these trends and uses them to predict the test set.
- Most commonly found errors in machine learning are Bias and Variance.
- The bias is the discrepancy between our actual values and the predictions. In order for our model to be able to forecast new data, it must make some basic assumptions about our data.
- Variance is the variability of a model's forecast for a certain data point or value, which indicates how widely distributed our data are.
- Low bias and low variance are the aims of any supervised machine learning method. The algorithm should then perform well in terms of predictions.
- It's crucial to comprehend prediction errors whenever we talk about model prediction (bias and variance). The ability of a model to reduce bias and variation must be balanced. Building accurate models and avoiding the overfitting and underfitting errors would both be made easier with a thorough grasp of these flaws.

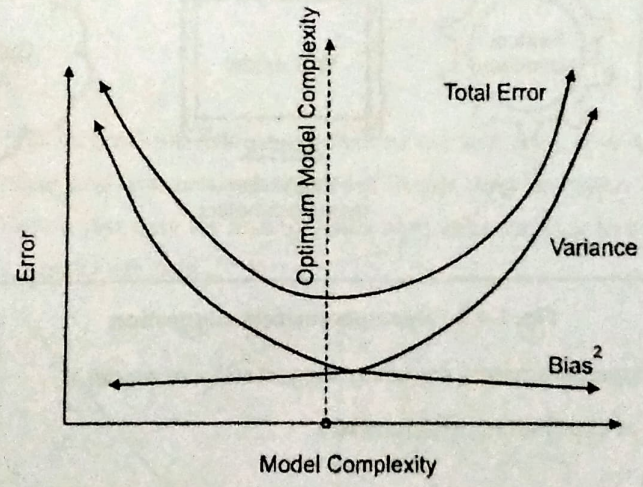


Fig. 1.3.1 : The Bias-Variance Trade-off

- With optimal model complexity and minimum error a compromised values of bias and variance are to be accepted as shown in Fig. 1.3.1.
- The model would never be overfit or underfit if bias and variance were balanced optimally.
- As shown in Fig. 1.3.1, our model might have high bias and low variance if it is overly straightforward and has few parameters. However, if our model includes many parameters, it will have a high variance and a low bias. Therefore, we must strike a balance between the two without overfitting or underfitting the data.

$$\text{Bias} + \text{Variance} + \text{Irreducible Error} = \text{Total Error.}$$

- The "noise" that cannot be removed through modelling, known as irreducible error, would be decreased with the use of data purification.

- The bias and variance issues must be balanced, and we can achieve optimal results for our problem depending on the algorithms we select and how we configure them. It is impossible for an algorithm to be both more and less complex at the same time. A trade off among bias and variance has to be established to have optimum results from algorithm.

1.4 Hyperparameters

- The model parameters in machine learning and deep learning represent a model.
- Hyperparameters are variables whose values influence the learning process and define the model parameter values. The prefix "hyper_" implies that these parameters are "top-level" controls over the learning process and the model parameters that emerge from it.
- The learning algorithm uses hyperparameters when learning, but they are not included in the model that is produced. We have the trained model parameters effectively, what we refer to as the model.

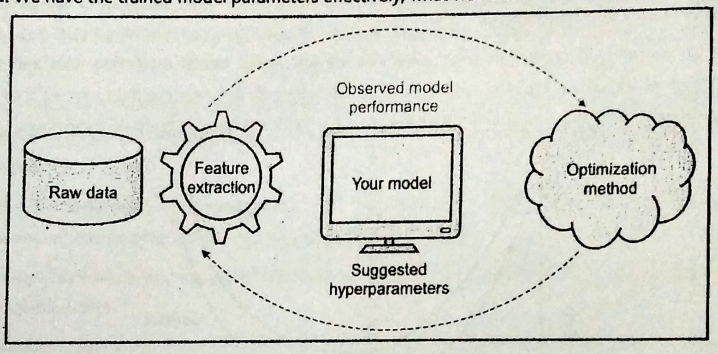


Fig. 1.4.1 : Hyperparameters suggestion

- Fig. 1.4.1 explains how the hyperparameters are being suggested from model.
- Following are the examples of common hyperparameters
  - Test-to-train ratio
  - In optimization methods, learning rate (e.g. gradient descent)
  - The optimization algorithm(e.g., gradient descent, stochastic gradient descent, or Adam optimizer)
  - Choosing an activation function for a layer of a neural network (NN)(e.g. Sigmoid, ReLU, Tanh)
  - The cost or loss function
  - Number of hidden layers
  - each layer's number of activation units
  - Number of epochs (iterations)
  - Size of the pool
  - Sample size

1.5 Underfitting / Overfitting Regularization

- If a model generalizes any new input data from the problem domain in an appropriate way, it is said to be a good machine learning model. This enables us to forecast the data which model has never seen.
- Let's say we want to assess how well our machine learning model picks up new information and adapts to it. Overfitting and underfitting are the main causes of the poor performance of machine learning methods for this.

1.5.1 Underfitting

- When a statistical model or machine learning algorithm is unable to recognise the underlying pattern in the data, or when it only performs well on training data but badly on testing data, this is referred to as underfitting. Such models are over simplified.
- It typically occurs when we try to develop a linear model with fewer non-linear data or when we have insufficient non-linear data to build an accurate model. The machine learning model will likely produce a lot of incorrect predictions in such circumstances since the rules are too simple and flexible to be applied to such sparse data. More data can be used to prevent underfitting, while feature selection can be used to reduce the number of features.
- In a word, underfitting describes a model that is unable to generalise new data also not doing well with training data.

1.5.2 Overfitting

- When a statistical model fails to produce reliable predictions on test data, it is said to be overfitted. A model begins to learn from the noise and erroneous data entries in our data set when it is trained with such a large amount of data. And when using test data for testing yields high variance. Due to too many details and noise, the model fails to appropriately identify the data.

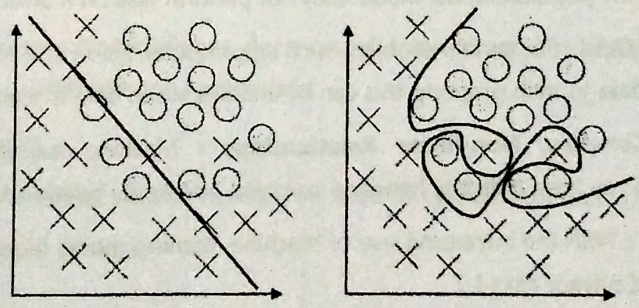


Fig. 1.5.1 : Underfitting and overfitting of data

- Fig. 1.5.1 above shows how the data points are classified by underfitted and overfitted model respectively.
- Simply put, overfitting is a problem when the evaluation of machine learning algorithms on training data differs from the evaluation on unknown data.
- Predicting the object in an image would be a more machine-learning-like case. Let's say our goal is to create a model that can identify tomatoes in images :
  - Model A has a red circle with a green star-shaped top and a few water drops.
  - Model B : Circle, Red

- Not all tomatoes in model A have water droplets on them, which is an issue. This device is overly precise and susceptible to selecting moist tomatoes. It cannot be applied universally to tomatoes. It cannot forecast dry tomatoes in an image since it will look for water droplets. It fits too well(Overfitting)
- Model B, on the other hand, incorrectly believes that any object that is red and circular is a tomato. This model is too broad and cannot identify crucial tomato characteristics. It doesn't fit well(underfitting).

1.5.3 Regularization

- The fundamental cause of overfitting is adding unnecessary complexity to a model. The overfitting problem can be resolved if a method to lessen complexity is found. Complex models are penalized by regularization.
- Regularization increases the penalty for more complex terms in the model, which reduces the complexity of the model. When regularization terms are included, the model seeks to reduce both loss and model complexity. Regularization decreases variance without significantly increasing bias.

1.6 Limitations of Machine Learning

There are several limitations of machine learning, including:

- **Lack of explainability :** Many machine learning models are considered "black boxes" because it can be difficult to understand how they arrived at a particular decision.
- **Overfitting :** Overfitting occurs when a model is trained too well on the training data and performs poorly on new, unseen data.
- **Data bias :** Machine learning models can perpetuate and even amplify bias present in the training data.
- **Lack of diversity :** Machine learning models are only as good as the data they are trained on. If the data is not diverse or representative of the population, the model may not perform well on a diverse set of inputs
- **Requires large amount of Data :** For certain problem such as computer vision and NLP, machine learning model requires a large amount of data to train properly, this can be an issue when data is scarce or expensive.
- **Difficulty in Handling Complex, Non-linear Relationships :** Machine learning models are based on mathematical equations and can have difficulty handling complex, non-linear relationships in the data.
- **Security and Privacy Risks :** With the increasing use of machine learning model in sensitive areas, it is important to consider their security and privacy risks.1.7

1.7 History of Deep Learning

- Deep learning is a subfield of machine learning that is inspired by the structure and function of the human brain, specifically the neural networks that make up the brain. The history of deep learning can be traced back to the 1940s and 1950s, when researchers first began developing models of artificial neural networks.
- In the 1960s and 1970s, the field of artificial intelligence experienced a resurgence of interest and funding, and researchers began experimenting with more complex neural network architectures.
- In the 1980s and 1990s, the foundations for modern deep learning include the backpropagation algorithm for training multi-layer neural networks and the development of convolutional neural networks for image recognition.

- In the early 2000s, the availability of powerful computers and large amounts of data, along with advances in optimization algorithms, allowed for the training of deeper and more complex neural networks. This led to breakthroughs in a variety of tasks, such as image and speech recognition.
- In the 2010s, deep learning began achieving state-of-the-art performance on a wide range of tasks. This led to an explosion of interest in the field, and deep learning is now widely used in industry, academia, and research.
- Recently, the field is rapidly advancing with the development of more complex architectures like transformer and GPT-3 and also applying deep learning in more and more diverse fields like recommendation systems, drug discovery, self-driving cars.

1.8 Advantages and Challenges of Deep Learning

Advantages of Deep Learning

1. **Feature engineering :** Deep learning can perform feature engineering on its own .A deep learning system will examine the data in search of complex features that correlate and combine them to enable faster learning.
2. **Best possible results with unstructured data :** Different data formats can be used to train deep learning algorithms. For instance, to forecast future stock values for a specific firm, a deep learning algorithm can find any connections between images, social media activity, industry research, weather forecast, and more.
3. **No need for well-labeled data :** When employing a deep learning strategy, the usage of well-labeled data is no longer required because the algorithms are excellent at learning without any rules.
4. **High quality results :** A deep learning model can execute thousands of repetitive activities in a relatively short amount of time after being properly trained. Additionally, If the training data is highly related to the addressed problem, the work's quality never declines.

Challenges of Deep Learning

- In contrast to other machine learning techniques, deep neural networks require a huge amount of training data. Make sure your training data is sufficient and appropriate.
- It includes bias-related problems.
- Depending on the quantity and size of your DL models, we need to optimize computing expenditures.
- Data security measures to ensure privacy.

1.9 Learning Representations from Data

- It is usually possible to learn representations of data by transforming it or extracting features from it by some method. These representations make classification and prediction easier.
- When a model learns a function, the learning is stored in the parameters of the model, since during training the function translates input to output. This process is known as representation learning. In this context, representation learning refers to the features of the transformed input. Understanding the representations of given dataset from particular perspective is representation learning. The model's architecture and the learned parameters both play a crucial role in mapping input to output.

1.10 Understanding How Deep Learning Works in Three Figures

- Neurai networks are used to implement deep learning, biological neurons serve as the inspiration for these networks.
- So in essence, deep networks which are nothing more than neural networks with several hidden layers.
- We are aware that the goal of machine learning is to map inputs to targets. This is accomplished by looking at numerous samples of input and targets. We also know that deep neural networks use a complex series of straightforward data transformations to carry out this input-to-target translation. These layers of data transformations are discovered by exposure to examples.
- The first figure that can help in understanding deep learning is the architecture of a neural network. A neural network is composed of layers of interconnected "neurons," which are inspired by the structure and function of biological neurons in the brain. Each neuron in a layer receives input from the previous layer, processes it, and sends it on to the next layer. The input layer receives the raw data, and the output layer produces the final output of the network. The layers in between are called "hidden layers," and they are used to extract features and representations of the data.
- The second figure that can help in understanding deep learning is the process of training a neural network. In supervised learning, a dataset with labelled examples is used to train the network. The network is presented with inputs and the corresponding desired outputs, and its weights and biases are adjusted to minimize the difference between the network's predictions and the desired outputs. This process is repeated for many examples in the dataset, and the network gradually learns to make accurate predictions on new, unseen examples.
- The third figure that may help understanding deep learning is the forward and backward propagation, it is the process of passing input data through the layers of the neural network and computing the output, as well as adjusting the weights of the network in the backward pass by using a optimization algorithm, like Stochastic Gradient Descent (SGD), this is the process which the network learn from data by minimizing the error between the predicted output and the actual output

1.11 Common Architectural Principles of Deep Network

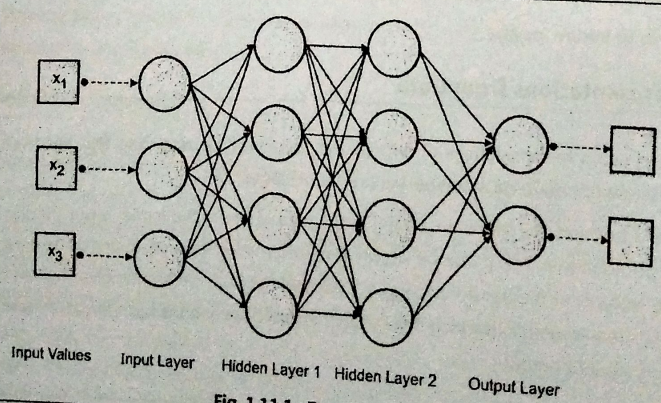


Fig. 1.11.1 : Deep network

- To develop non-linear models that best classify our data, The combination the models and the hidden layers. When our data is too complicated, we must combine non-linear models to produce even more non-linear models in order to classify it.
- This can be repeated numerous times with even more hidden layers to produce extremely complicated models.
- It is more difficult to classify this kind of data. To create a model that accurately classifies this data, numerous hidden layers of models must be combined with a specific set of weights.
- After that, we can use a feed-forward technique to generate some output. Prior to creating an output, the input would need to pass through the complete depth of the neural network. Just a stacked perceptron, really. Since the trend of our data is not straightforward in a deep neural network, this non-linear boundary is only a reliable model that can appropriately categorise a very complicated set of data.
- **Use of multiple layers :** Deep networks are composed of multiple layers of interconnected nodes, which allow them to learn increasingly abstract representations of the input data.
- **Use of non-linear activation functions :** Non-linear activation functions, such as ReLU or sigmoid, are used in the nodes of the network to introduce non-linearity into the computation.
- **Gradient-based learning :** Deep networks are trained using gradient-based optimization algorithms, such as stochastic gradient descent, to adjust the weights of the network to minimize the error on the training data.
- **Use of backpropagation :** Backpropagation is a method for training neural networks, which allows the gradients of the error with respect to the weights of the network to be efficiently computed.
- **Use of Dropout :** Dropout is a method to prevent overfitting in deep networks, which consists of randomly dropping out some nodes during training, effectively averaging over multiple different models.
- **Use of Batch Normalization :** Batch normalization is a technique that helps to improve the stability and speed of the training deep neural networks.
- Use of Convolutional layers for images and sequences
- Using Recurrent layers for sequential data
- These are just a few examples of architectural principles that are commonly used in deep neural networks. The specific architecture of a network will depend on the task it is being used for and the amount and characteristics of the data available.

1.12 Architecture Design

The architecture design of a deep neural network typically involves several decisions, such as:

- **The number of layers :** This depends on the complexity of the task and the size of the dataset. A deeper network with more layers can potentially learn more complex functions, but it also requires more data to prevent overfitting.
- **The number of neurons in each layer :** This determines the capacity of the network. A larger number of neurons in a layer means the network has more parameters to learn from the data, but it also increases the risk of overfitting.

- **The type of activation function :** Activation functions are used to introduce non-linearity into the network, allowing it to learn more complex functions. Common activation functions include sigmoid, tanh, ReLU, and LeakyReLU.
- **The type of layer :** Different types of layers, such as convolutional layers and recurrent layers, are suitable for different types of data. Convolutional layers are well-suited for image data, while recurrent layers are well-suited for sequential data such as time series or natural language.
- **The strategy for regularization :** Regularization is used to prevent overfitting. Common techniques include dropout, weight decay, and early stopping
- Overall, the architecture design of a deep neural network is a iterative process, involving a lot of trial and error.
- It is also important to note that the architecture design of deep networks can also be done using automated techniques such as Neural Architecture Search (NAS)
- Where NAS is a method for automating the design of neural network architectures. It uses a combination of machine learning techniques and gradient-based optimization to search for architectures that are well-suited for a given task and dataset.

1.13 Applications of Deep Learning

- Deep learning is a subset of machine learning that involves training artificial neural networks with many layers to learn from large amounts of data. Deep learning has been applied to a wide range of tasks across different domains, including:

1. **Computer vision**

   Deep learning has been used to develop state-of-the-art models for tasks such as object detection, image classification, facial recognition, and video analysis.

2. **Natural language processing**

   Deep learning has been used to develop models for tasks such as language translation, sentiment analysis, text summarization, and speech recognition.

3. **Autonomous systems**

   Deep learning has been applied to develop models for autonomous vehicles, robotics, and drones, enabling them to perform tasks such as navigation, obstacle avoidance, and object detection.

4. **Healthcare**

   Deep learning has been used to develop models for medical image analysis, disease diagnosis, drug discovery, and personalized medicine.

5. **Finance**

   Deep learning has been applied to develop models for financial forecasting, fraud detection, and credit risk assessment.

6. **Gaming**

   Deep learning has been used to develop models for game playing, enabling computers to beat human champions in games such as Go, chess, and poker.

7. **Agriculture**

   Deep learning has been used to develop models for crop yield prediction, disease detection, and precision agriculture.

- Image recognition and object detection in computer vision tasks.
- Language processing tasks such as natural language understanding, machine translation, and text-to-speech synthesis.
- Speech recognition and audio processing tasks such as speech-to-text and music generation.
- Recommender systems for suggesting products or content to users.
- Predictive maintenance for manufacturing and industrial systems.
- Self-driving cars
- Drug discovery and generative chemistry
- GANs (Generative Adversarial Networks) for generating images, audio, and text.
- Robotics and reinforcement learning
- Fraud Detection, anomaly detection in financial transactions and stock prices etc.
- These are just a few examples of the many applications of deep learning. Deep learning has the potential to transform many industries and domains, as it can learn to extract features and patterns from large and complex data sources, and can provide insights that are difficult or impossible to obtain with traditional approaches

1.14 Introduction and Use of Popular Industry Tools for Machine Learning and Deep Learning

- TensorFlow is an end-to-end open-source platform for machine learning. It provides a comprehensive set of tools for building and deploying machine learning models. It allows for easy deployment of models to a variety of platforms, including mobile devices and web browsers. TensorFlow can be used for a variety of tasks, including image and speech recognition, natural language processing, and time series forecasting.
- Keras is a high-level neural networks API written in Python. It is a user-friendly API that makes it easy to build, train, and evaluate deep learning models. Keras is compatible with both TensorFlow and Theano, and it can be used to quickly build and experiment with deep learning architectures.
- PyTorch is another open-source machine learning library based on the Torch library. It is used for natural language processing and computer vision. It has dynamic computation graphs, which allow for more flexibility and faster debugging compared to static computation graphs. Pytorch's ecosystem has strong visualization and model interpretability tools.
- Caffe is a deep learning framework developed by the Berkeley Vision and Learning Center (BVLC). It is primarily used for image classification and segmentation. It is known for its speed and efficiency, and it has been used to train large-scale models on image datasets such as ImageNet.
- Shogun is a machine learning library that provides a wide range of algorithms for a variety of tasks, including classification, regression, clustering, and dimensionality reduction. It is implemented in C++ and provides interfaces to several programming languages, including Python and Matlab.

- All those libraries are widely used in research and Industry, the final choice of library mostly depends on the task, the availability of resources and personal preference of developers.

Review Questions

Q. 1 What is Machine Learning ?

Q. 2 What is the importance of Machine Learning?

Q. 3 Explain Machine Learning process.

Q. 4 What are the applications of Machine Learning?

Q. 5 Explain Supervised learning with the help of example.

Q. 6 How supervised learning works?

Q. 7 What is unsupervised learning?

Q. 9 What are the advantages and disadvantages of unsupervised learning?

Q. 10 Explain advantages and disadvantages of deep learning.

□□□

2 Deep Neural Networks (DNNs)

Unit II

Syllabus

Introduction to Neural Networks : The Biological Neuron, The Perceptron, Multilayer Feed-Forward Networks, **Training Neural Networks :** Backpropagation and Forward propagation **Activation Functions :** Linear, Sigmoid, Tannh, Hard Tanh, Softmax, Rectified Linear, **Loss Functions :** Loss Function Notation, Loss Functions for Regression, Loss Functions for Classification, Loss Functions for Reconstruction, **Hyperparameters :** Learning Rate, Regularization, Momentum, Sparsity, Deep Feedforward Networks – Example of Ex OR, Hidden Units, cost functions, error backpropagation, Gradient-Based Learning, Implementing Gradient Descent, vanishing and Exploding gradient descent, Sentiment Analysis, Deep Learning with Pytorch, Jupyter, colab.

2.1 Introduction to Neural Networks

- The structure and operation of the human brain served as the inspiration for a particular form of machine learning model called a neural network.
- It is made up of layers of neurons, which are interconnected processing units. The input data is processed by each neuron, and the result is passed on to the next layer until it reaches the output layer, where the final prediction is made.
- Neural networks are trained to perform tasks, such as image classification, language translation, and even playing games. They are widely used in artificial intelligence applications due to their ability to learn complex relationships in large amounts of data.

2.2 Biological Neuron

- The nervous system is responsible for receiving sensory information from the environment, processing it, and sending signals to the appropriate organs and muscles to respond.
- The fundamental units of the nervous system are biological neurons, which collaborate to process and send information. They play a critical role in sensory perception, motor control, and various cognitive processes, including learning and memory.

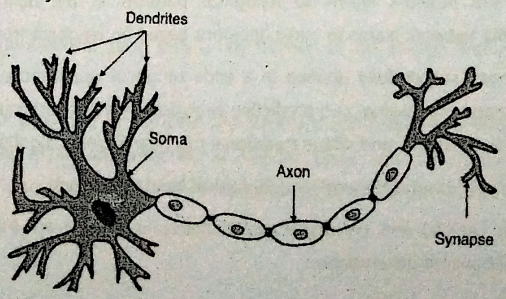


Fig. 2.2.1

- The basic structure of a neuron includes the following:
  1. **Cell body (soma) :** The cell body contains the nucleus, which houses the genetic material of the cell, and other organelles that are necessary for the cell to function. Signals are sent to neighboring neurons or muscles by the long, thin axon, which extends from the cell body.
  2. **Dendrites :** Dendrites are the branch-like structures that extend from the cell body and receive signals from other neurons.
  3. **Axon :** The axon is a long, slender fibre that extends from the cell body and carries signals away from the cell body to other neurons, muscles, or glands.
  4. **Synapse :** A synapse is a junction between the axon of one neuron and the dendrites or cell body of another neuron, where the two neurons communicate through the release of chemicals called neurotransmitters.
- When a neuron receives a signal from another neuron through its dendrites, the signal is processed in the cell body and, if it is strong enough, an electrical signal is generated that travels down the axon. At the end of the axon, the electrical signal triggers the release of neurotransmitters, which bind to receptors on the dendrites or cell body of the next neuron in the chain. This process of receiving, processing, and transmitting signals between neurons forms the basis of information processing in the nervous system.
- The structure and function of biological neurons have inspired the development of artificial neural networks, which are the basis of modern machine learning and deep learning algorithms. Artificial neurons are designed to mimic the function of biological neurons, with input signals coming in through the "dendrites" and output signals being sent out through the "axon."

2.2 Artificial Neural Network

- An artificial neural network is a type of machine learning model that is inspired by the structure and function of biological neural networks, such as the human brain. It consists of interconnected nodes, or artificial neurons, that process and transmit information using mathematical algorithms.
- The basic building block of a neural network is the artificial neuron, which receives input from other neurons or external sources, computes a weighted sum of those inputs, and applies a non-linear activation function to the result. This output is then passed on to other neurons in the network, ultimately resulting in a final output.
- Neural networks can be trained using supervised learning, unsupervised learning, or reinforcement learning. In supervised learning, the network is trained using labeled data to learn to make predictions or classify inputs. In unsupervised learning, the network learns to recognize patterns in the data without explicit labeling. In reinforcement learning, the network learns to make decisions based on feedback from its environment.
- Neural networks have been successfully applied in a wide range of applications, including image and speech recognition, natural language processing, and predictive analytics. They are particularly effective in tasks where the data is complex or high-dimensional, and where traditional machine learning algorithms may struggle.
- Comparison between Artificial Neural Network and Biological Neural Network
- Artificial neural networks (ANNs) and biological neural networks (BNNs) are both systems of interconnected neurons that process and transmit information.

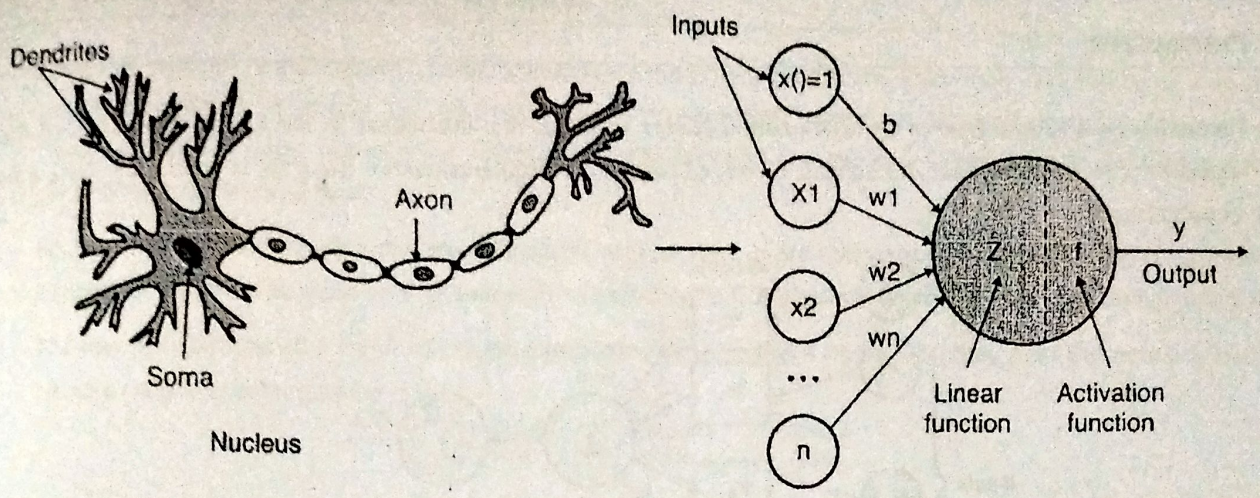


Fig. 2.2.1

However, there are several key differences between them:

1. **Structure**

   ANNs are designed to replicate the structure of BNNs, but there are some differences in their organization. ANNs typically have a layered structure, where each layer is made up of multiple neurons that perform specific tasks. In contrast, BNNs are more complex and have a highly interconnected structure that allows for parallel processing and feedback loops.

2. **Processing**

   The neurons in ANNs perform mathematical computations on input data, whereas the neurons in BNNs communicate via electrical and chemical signals. ANNs use digital signals, while BNNs use analog signals.

3. **Learning**

   ANNs are designed to learn from data through a process called backpropagation, where errors in the output are fed back through the network to adjust the weights of the neurons. BNNs, on the other hand, learn through a complex process of experience-dependent plasticity, where the strength of connections between neurons is modified based on experience and feedback from the environment.

4. **Robustness**

   BNNs are highly robust and adaptable, able to withstand damage and reorganize in response to changes in the environment. ANNs, while effective at specific tasks, are much less robust and adaptable.

5. **Energy efficiency**

   BNNs are incredibly energy-efficient, using only a tiny fraction of the energy required by ANNs to perform similar tasks.

Overall, while ANNs have been designed to replicate the structure and function of BNNs, they still have some fundamental differences. While ANNs are effective at specific tasks and can outperform BNNs in certain applications, they still have a long way to go in replicating the adaptability and robustness of their biological counterparts.

2.3 Perceptron

- The Perceptron is a simple type of artificial neural network that was first introduced in the 1950s. It was one of the first algorithms to be developed for solving binary classification problems, where the goal is to classify data into one of two classes.

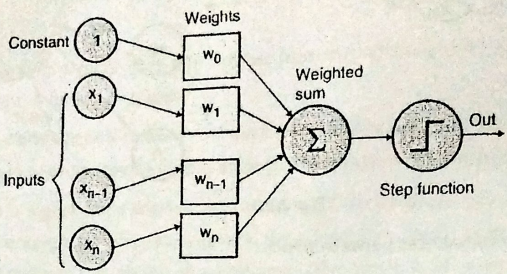


Fig. 2.3.1

- A perceptron is a type of artificial neuron that is used in neural networks to classify input data into one of two categories, based on a set of weights and thresholds.
- The perceptron takes input from several sources and applies a weight to each input. These weights determine how important each input is in making a decision. The perceptron then calculates the weighted sum of these inputs, and compares this sum to a threshold value. If the sum is greater than the threshold, the perceptron outputs a 1, indicating that the input belongs to one category. If the sum is less than the threshold, the perceptron outputs a 0, indicating that the input belongs to the other category.
- The process of adjusting the weights and threshold values of the perceptron to improve its accuracy is called training. A Perceptron consists of a single layer of artificial neurons, each of which is connected to inputs and produces an output. The inputs to the Perceptron can be any real-valued features, and the outputs are binary values that indicate the class membership of the input data. The Perceptron uses a set of weights to make predictions, and these weights are updated based on the error between the actual and predicted outputs.
- During training, the perceptron is presented with a set of labeled data, and the weights and thresholds are adjusted based on the errors in the output. This process continues until the perceptron reaches a level of accuracy that is acceptable for the given task.
- Perceptrons can be used in single-layer neural networks to classify linearly separable data. Linearly separable data is data that can be separated by a straight line, or hyperplane, in n-dimensional space. However, perceptrons are limited in their ability to handle non-linearly separable data. To handle non-linearly separable data, more complex neural networks, such as multilayer perceptrons, are needed.
- Although the Perceptron is a simple algorithm, it laid the foundation for more complex neural network architectures, such as multi-layer perceptrons (MLPs), which consist of multiple layers of artificial neurons and can be used to solve more complex problems.
- Perceptrons have been used in a wide range of applications, including image and speech recognition, natural language processing, and predictive analytics. They are particularly effective in tasks where the data is linearly separable, and where there are only two categories.

2.4 Multilayer Feedforward Networks

- A sort of artificial neural network called a multilayer feedforward network, commonly referred to as a multi-layer perceptron (MLP), is made up of numerous interconnected layers of artificial neurons. They are called "feedforward" networks because the information flows through the network in only one direction, from input to output, without looping back.
- Each layer in an MLP consists of a set of artificial neurons, which are connected to the neurons in the previous and subsequent layers. The input layer provides the input data, and the output layer generates the final prediction.
- The layers in between the input and output layers are called hidden layers, and they help the network learn more complex representations of the data.

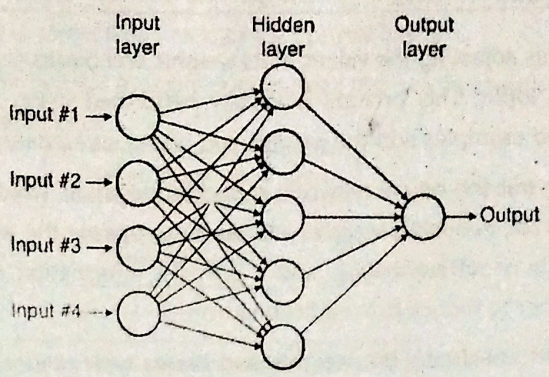


Fig. 2.4.1

- MLPs are trained using supervised learning algorithms, such as backpropagation, where the network is presented with labeled examples and adjusts its weights based on the prediction error. The training process continues until the network converges to a set of weights that produce accurate predictions on the training data.
- MLPs are widely employed in many different applications, such as speech recognition, image classification, and natural language processing. They have also been used as building blocks for more complex deep neural networks, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs)
- Simple feedforward neural networks for image categorization are one type of neural network. An input layer, one or more hidden layers, and an output layer make up this structure.
- The input layer receives the pixel values of an image, the hidden layers process the data through matrix operations and activation functions, and the output layer gives the predicted class label for the image. During the training process, the model adjusts the weights and biases of the neurons based on the error between the predicted output and the actual label. Once the training is complete, the neural network can be used to predict the class of new images.

A simple case study of neural network

A simple case study of a neural network could be a binary classification problem where the goal is to predict if a person is diabetic or not based on their health information.

Here's how the study could be performed :

1. **Data collection :** Collect a dataset of health information of people along with their diabetic status (yes/no).
2. **Data preparation :** Clean and preprocess the data to prepare it for feeding into the neural network.
3. **Model training :** Train a feedforward neural network with one or multiple hidden layers on the prepared data using an optimization algorithm, such as gradient descent.

4. **Model evaluation :** Evaluate the performance of the trained model on a separate test dataset. Measure the accuracy, precision, recall, and other performance metrics.
5. **Model improvement :** Based on the evaluation results, improve the model by adjusting its architecture, changing the activation functions, or tuning the hyperparameters.
6. **Deployment :** Once the model has satisfactory performance, deploy it in a real-world scenario to make predictions on new patients' information.

This is just one example of how a neural network can be applied. The choice of architecture, preprocessing, and evaluation metrics can vary based on the specific problem and the nature of the data.

2.5 Training Neural Networks

- Training a neural network involves adjusting the values of its weights and biases so that the network can accurately predict the output for a given input. This process is usually performed using supervised learning, where the network is presented with labeled examples and the weights and biases are updated based on the prediction error.
- The most common algorithm for training neural networks is backpropagation, which involves using the gradient of the loss function with respect to the network's weights and biases to update the weights and biases. The gradient is calculated using the chain rule of differentiation, and it provides information on how much the weights and biases need to be adjusted in order to reduce the prediction error.
- The training process begins with initializing the weights and biases with random values. Then, the network is presented with a batch of training examples, and the weights and biases are updated using gradient descent or a similar optimization algorithm. The process is repeated multiple times, using different batches of examples, until the prediction error reaches a satisfactory level or the maximum number of training iterations is reached.
- It's important to carefully monitor the training process to avoid overfitting, which occurs when the network becomes too complex and starts to memorize the training examples instead of learning the underlying relationships between the inputs and outputs. To prevent overfitting, various regularization techniques can be used, such as dropout, early stopping, and L1 or L2 regularization.
- Once the training process is complete, the network can be used to make predictions on new, unseen examples. The accuracy of the predictions depends on the quality of the training data and the choice of hyperparameters, such as the number of hidden layers, the number of neurons in each layer, and the learning rate.

2.5.1 Back Propagation

- Backpropagation and forward propagation are two important algorithms used in the training of neural networks.
- Backpropagation is the process of updating the weights and biases of a neural network during training. It is used to adjust the weights and biases so that the network can make more accurate predictions for the training examples. Backpropagation utilizes the gradient of the loss function with regard to the network's weights and biases to calculate the updates.
- The backpropagation algorithm is a supervised learning algorithm used to train artificial neural networks with multiple layers, also known as multilayer perceptrons (MLPs). The architecture of a backpropagation network typically consists of an input layer, one or more hidden layers, and an output layer. The number of hidden layers and neurons in each layer can vary depending on the complexity of the task.
- Here is a step-by-step explanation of the backpropagation algorithm and a flowchart of the process:

1. **Initialize the network weights :** The weights in the network are randomly initialized to small values, typically between -0.5 and 0.5.

2. **Forward propagation :** The input data is fed into the network and propagated forward through the layers, with each neuron performing a weighted sum of its inputs and applying an activation function to the result. This produces an output for the network.
3. **Calculate the error :** The output of the network is compared to the true value of the data and an error is calculated using a loss function such as mean squared error.
4. **Backward propagation :** The error is propagated backward through the network from the output layer to the hidden layers, using the chain rule of calculus to calculate the derivative of the error with respect to the weights.
5. **Weight update :** The weights in the network are updated using the derivative of the error with respect to the weights, multiplied by a learning rate. The learning rate determines the step size in the weight update and is usually set to a small value to avoid overshooting the optimal weights.
6. **Repeat :** Steps 2-5 are repeated for multiple epochs, with the weights being updated after each forward and backward pass through the network.

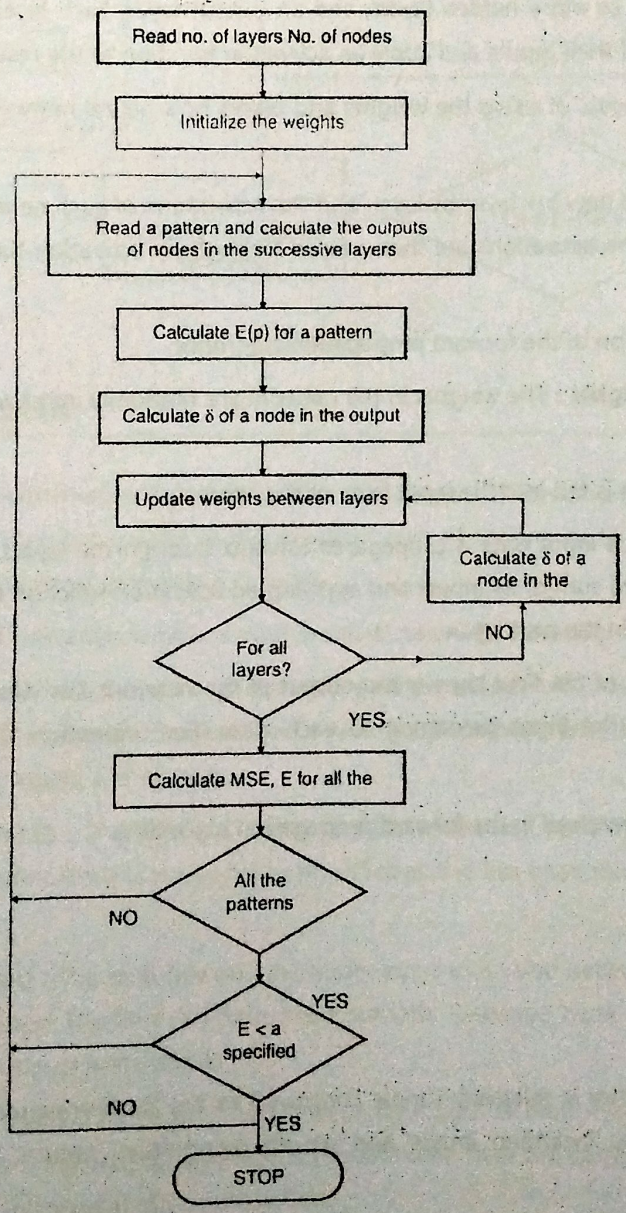


Fig. 2.5.1

- The process of backpropagation starts with the calculation of the prediction error for the current training example. The error is used to calculate the gradients of the loss function with respect to the output activations. These gradients are then propagated backwards through the network, layer by layer, to calculate the gradients with respect to the inputs and weights of each neuron. Finally, the weights and biases are updated using gradient descent or a similar optimization algorithm.
- Backpropagation is a computationally efficient way to train neural networks, as it allows the gradients to be calculated efficiently using the chain rule of differentiation. It is also highly parallelizable, as the computations for each neuron can be performed in parallel.

2.5.2 Forward Propagation

- The forward propagation algorithm, also known as feedforward neural network, is a type of artificial neural network that is commonly used for classification and regression tasks. The architecture of a feedforward neural network consists of an input layer, one or more hidden layers, and an output layer. Each layer contains multiple neurons that perform a weighted sum of their inputs and apply an activation function to the result.
- Forward propagation is the process of using the weights and biases of a neural network to make predictions for a given input.
- The input is passed through the network layer by layer, and the activations of each neuron are calculated using the weighted sum of the inputs. The activations are then passed through an activation function, which produces the final outputs of the network.
- Here is a step-by-step explanation of the forward propagation algorithm:
  1. **Initialize the network weights :** The weights in the network are randomly initialized to small values, typically between -0.5 and 0.5.
  2. **Input data :** The input data is fed into the input layer of the network.
  3. **Forward propagation :** The input data is propagated forward through the layers, with each neuron in each layer performing a weighted sum of its inputs and applying an activation function to the result. This produces an output for each neuron in the next layer.
  4. **Output layer :** The output of the final layer is the output of the network. For classification tasks, the output can be the probability of the input belonging to each class. For regression tasks, the output can be a continuous value.
- Here is a summary of the steps involved in the forward propagation algorithm:
  1. Initialize network weights
  2. Input data
  3. Forward propagation
  4. Output layer
- The forward propagation algorithm is relatively simple compared to the backpropagation algorithm, and can be used for a wide range of tasks, including image and speech recognition, natural language processing, and predictive analytics

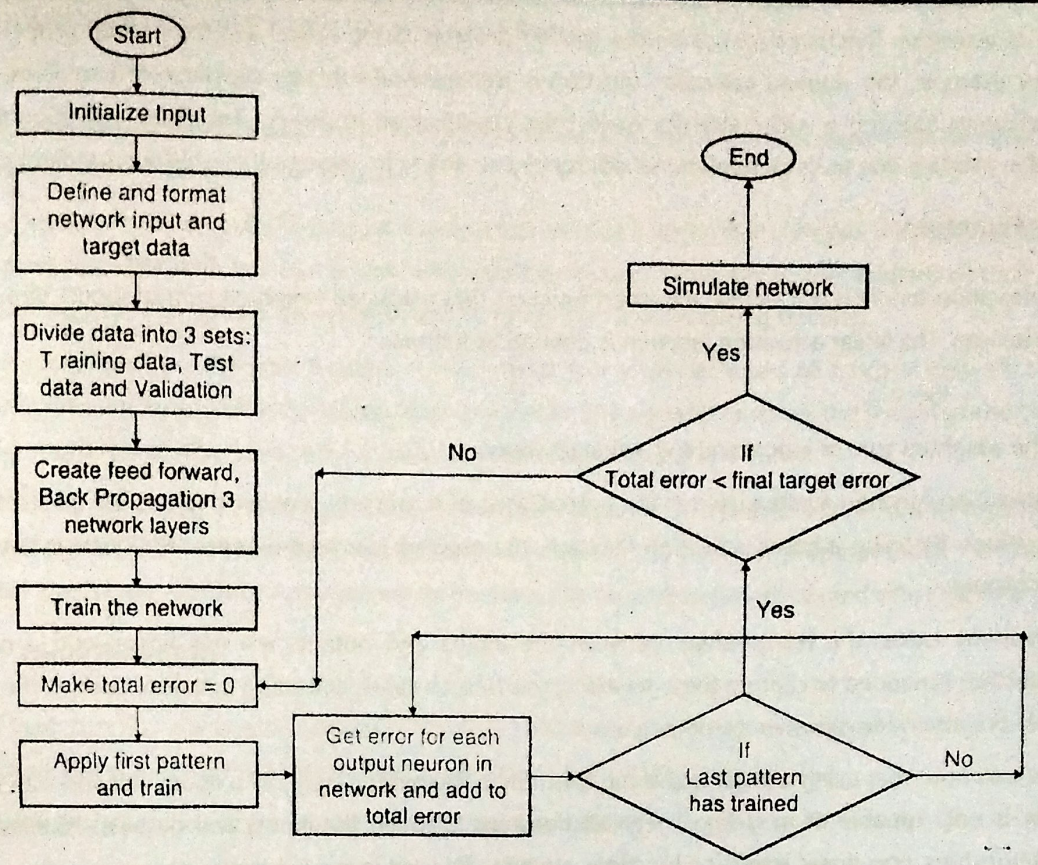


Fig. 2.5.2

2.6 Activation Functions

- Activation functions are mathematical functions used in artificial neural networks to introduce non-linearity into the output of each neuron. They are applied to the weighted sum of the inputs to produce the neuron's activation, which is used as input to the next layer in the network.
- There are several types of activation functions used in neural networks, including:
- **Sigmoid :** The sigmoid function maps any input to a value between 0 and 1, making it useful for binary classification problems. The sigmoid function has a smooth, S-shaped curve that allows it to model complex relationships between the inputs and outputs.
- **Rectified Linear Unit (ReLU) :** The ReLU activation function maps any input below zero to zero, and any input above zero to the same value. ReLU is computationally efficient and has been found to work well in many practical applications.
- **Hyperbolic Tangent (tanh) :** The tanh function maps any input to a value between -1 and 1, making it useful for data that has a similar range. The tanh function has a smooth, S-shaped curve that allows it to model complex relationships between the inputs and outputs.
- **SoftMax :** The SoftMax activation function is used in the output layer of multi-class classification problems. It maps the activations of the output neurons to a probability distribution over the possible classes, making it possible to interpret the network's predictions as class probabilities.

- The choice of activation function depends on the specific problem being solved and the desired properties of the network. For example, the sigmoid activation function is well-suited for binary classification problems, while the SoftMax activation function is well-suited for multi-class classification problems. The ReLU activation function is widely used in practice due to its computational efficiency and ability to alleviate the vanishing gradient problem.

2.6.1 Linear Function

- The linear activation function is a simple activation function that maps the weighted sum of inputs directly to the output of a neuron. The linear activation function is defined as follows :

$$f(x) = x$$

where x is the weighted sum of inputs and f(x) is the activation.

- The linear activation function is often used in the output layer of regression problems, where the goal is to predict continuous values. By using a linear activation function, the network can model linear relationships between the inputs and outputs.
- However, in many cases, the relationships between the inputs and outputs are non-linear, and a non-linear activation function is needed to capture these relationships. In such cases, activation functions such as the sigmoid, tanh, or ReLU activation functions are commonly used.
- It is important to note that using a linear activation function in the hidden layers of a neural network can lead to a network that is only capable of modeling linear relationships between the inputs and outputs. To model more complex relationships, non-linear activation functions are typically used in the hidden layers.

2.6.2 Sigmoid Function

- The sigmoid function is a type of activation function commonly used in artificial neural networks. The sigmoid function maps any input to a value between 0 and 1, making it useful for binary classification problems where the goal is to predict one of two classes.
- The mathematical definition of the sigmoid function is as follows:

$$f(x) = 1 / (1 + e^{(-x)})$$

- where x is the weighted sum of inputs to the neuron, e is the mathematical constant approximately equal to 2.71828, and f(x) is the output of the activation function.
- The sigmoid function has a smooth, S-shaped curve, which allows it to model complex relationships between the inputs and outputs. The sigmoid function is also differentiable, making it possible to use gradient descent or a similar optimization algorithm to update the weights and biases of the network during training.
- It is important to note that the sigmoid activation function can suffer from the vanishing gradient problem, where the gradients of the activation function become very small, making it difficult for the network to learn. This can be mitigated by using alternative activation functions such as the tanh or ReLU activation functions.

2.6.3 Tanh Function

- The tanh (hyperbolic tangent) function is a type of activation function commonly used in artificial neural networks. The tanh function maps any input to a value between -1 and 1, making it useful for data that has a similar range.

- The mathematical definition of the tanh function is as follows:

$$f(x) = \tanh(x) = 2 / (1 + e^{(-2x)}) - 1$$

where x is the weighted sum of inputs to the neuron, e is the mathematical constant approximately equal to 2.71828, and f(x) is the output of the activation function.

- The tanh function has a smooth, S-shaped curve, which allows it to model complex relationships between the inputs and outputs. The tanh function is also differentiable, making it possible to use gradient descent or a similar optimization algorithm to update the weights and biases of the network during training.
- Like the sigmoid function, the tanh function is often used in the hidden layers of a neural network to introduce non-linearity into the output of each neuron. However, unlike the sigmoid function, the output of the tanh function is centered around zero, which can make it easier for the network to learn.
- It is important to note that the choice of activation function depends on the specific problem being solved and the desired properties of the network. Different activation functions have different properties, and the best activation function for a particular problem may depend on the data, the network architecture, and other factors.

2.6.4 Hard Tanh Function

- The Hard Tanh function is a type of activation function used in artificial neural networks. It is similar to the standard tanh function, but with the added feature that it clips the output to the range [-1, 1].
- The mathematical definition of the Hard Tanh function is as follows:

$$\begin{aligned} f(x) &= -1, \text{ if } x < -1 \\ &= x, \text{ if } -1 <= x <= 1 \\ &= 1, \text{ if } x > 1 \end{aligned}$$

where x is the weighted sum of inputs to the neuron and f(x) is the output of the activation function.

- The Hard Tanh function is non-linear, and can be used to introduce non-linearity into the output of a neuron. The clipping property of the Hard Tanh function can help prevent the output from exploding, which can occur in some cases with other activation functions.
- It is important to note that the choice of activation function depends on the specific problem being solved and the desired properties of the network. Different activation functions have different properties, and the best activation function for a particular problem may depend on the data, the network architecture, and other factors.

2.6.5 SoftMax Function

- The SoftMax function is a type of activation function commonly used in artificial neural networks, particularly in the output layer of multi-class classification problems. The SoftMax function takes a vector of real numbers as input and returns a corresponding vector of non-negative values that sum to 1, making it suitable for representing a probability distribution over a set of classes.
- The mathematical definition of the SoftMax function is as follows:

$$f_i(x) = e^{(x_i)} / \text{sum}(e^{(x_j)})$$

- where x is a vector of inputs to the neuron, e is the mathematical constant approximately equal to 2.71828, and f_i(x) is the output of the activation function for the i-th class.

- The SoftMax function is differentiable, making it possible to use gradient descent or a similar optimization algorithm to update the weights and biases of the network during training. The SoftMax function has the desirable property of being able to spread the probability mass over all classes in a smooth manner, making it well suited for multi-class classification problems.
- It is important to note that the SoftMax function is computationally expensive to compute, especially for large number of classes, due to the need to compute the exponential function for each class. In some cases, alternative activation functions, such as the sigmoid function, can be used in conjunction with cross-entropy loss to achieve similar results for binary classification problems

2.6.6 Rectified Linear Function (ReLU)

- The rectified linear unit (ReLU) function is a type of activation function commonly used in artificial neural networks. The ReLU function takes a single real number as input and returns the input if it is positive, and 0 if it is negative.
- The mathematical definition of the ReLU function is as follows:

$$f(x) = \max(0, x)$$

where x is the weighted sum of inputs to the neuron and f(x) is the output of the activation function.

- The ReLU function has several advantages over other activation functions, including its simplicity, speed, and ability to alleviate the vanishing gradient problem. The vanishing gradient problem refers to the issue that can occur in deep neural networks when the gradient signal becomes very small during training, making it difficult for the network to learn.
- The ReLU function can help mitigate this issue by providing a non-linear activation function with a well-defined derivative, which makes it easier for the network to learn.
- It is important to note that the choice of activation function depends on the specific problem being solved and the desired properties of the network.
- Different activation functions have different properties, and the best activation function for a particular problem may depend on the data, the network architecture, and other factors.
- Additionally, while the ReLU function has many advantages, it can also be susceptible to dying ReLU problem, which refers to the issue that can occur when neurons with negative weights never activate, making it difficult for the network to learn. This can be mitigated by using variants of the ReLU function, such as the leaky ReLU or parametric ReLU.

2.7 Loss Functions

- Loss functions, also known as cost functions or objective functions, are used in machine learning to measure the discrepancy between the predicted outputs of a model and the true outputs. The goal of training a machine learning model is to minimize the value of the loss function.
- There are several different types of loss functions, including mean squared error (MSE), mean absolute error (MAE), categorical cross-entropy, binary cross-entropy, and hinge loss, among others. The choice of loss function depends on the specific problem being solved and the type of model being used.
- Mean Squared Error (MSE) is a commonly used loss function for regression problems. It measures the average squared difference between the predicted and true outputs.

- Mean Absolute Error (MAE) is similar to MSE, but instead measures the average absolute difference between the predicted and true outputs.
- Categorical Cross-Entropy is a commonly used loss function for multi-class classification problems. It measures the distance between the predicted probability distribution and the true distribution.
- Binary Cross-Entropy is a similar loss function to categorical cross-entropy, but is used for binary classification problems.
- Hinge Loss is a commonly used loss function for support vector machines (SVMs) and other models for binary classification problems.
- In addition to the choice of loss function, the optimization algorithm used to minimize the loss function is also important. Common optimization algorithms include gradient descent, stochastic gradient descent, and others. The choice of optimization algorithm will depend on the specific problem being solved and the characteristics of the data and model.

2.7.1 Loss Function Notation

- The notation for loss functions in machine learning varies, but a common notation is to use $J(\theta)$ to represent the loss function, where θ represents the parameters of the model. The goal of training a machine learning model is to minimize the value of $J(\theta)$ by updating the parameters θ.
- For example, if we have a regression problem with n training examples, the mean squared error (MSE) loss function can be written as:

$$J(\theta) = 1/n * \Sigma(i=1 \text{ to } n) (y_i - \hat{y}_i)^2$$

where y_i is the true output for the ith training example, $\hat{y}$_i is the predicted output, and Σ represents the sum over all training examples.

- For a binary classification problem, the binary cross-entropy loss function can be written as:

$$J(\theta) = -1/n * \Sigma(i=1 \text{to } n) y_i * \log(\hat{y}_i) + (1 - y_i) * \log(1 - \hat{y}_i)$$

where y_i is the true binary label for the i-th training example (0 or 1), $\hat{y}$_i is the predicted probability of the positive class, and log is the natural logarithm.

- In both cases, the loss function measures the discrepancy between the predicted outputs and the true outputs, and the goal of training is to find the parameters θ that minimize $J(\theta)$. The optimization algorithm used to minimize the loss function will depend on the specific problem being solved and the characteristics of the data and model

2.7.2 Loss Functions for Regression

There are several loss functions that can be used for regression problems, including:

1. **Mean Squared Error (MSE) :** The MSE loss function is defined as the average of the squared differences between the predicted values and the true values. The equation for MSE is given by:

$$J(\theta) = 1/n * \Sigma(i=1 \text{ to } n) (y_i - \hat{y}_i)^2$$

Where y_i is the true output for the i-th training example, $\hat{y}$_i is the predicted output, and n is the number of training examples.

2. **Mean Absolute Error (MAE) :** The MAE loss function is defined as the average of the absolute differences between the predicted values and the true values. The equation for MAE is given by:

$$J(\theta) = 1/n * \Sigma(i=1 \text{ to } n) |y_i - \hat{y}_i|$$

3. **Huber Loss :** The Huber loss function is a combination of mean squared error and mean absolute error, which makes it more robust to outliers than MSE. The equation for Huber loss is given by:

$$J(\theta) = 1/n * \Sigma(i=1 \text{ to } n) \{0.5 * (y_i - \hat{y}_i)^2 \text{ if } |y_i - \hat{y}_i| <= \delta, \delta * |y_i - \hat{y}_i| - 0.5 * \delta^2 \text{ otherwise}\}$$

where δ is a positive parameter that determines the transition between the mean squared error and mean absolute error regimes.

The choice of loss function will depend on the specific requirements of the problem and the characteristics of the data. In general, MSE is a good default choice for regression problems, while MAE is a good choice when the data contains outliers, and Huber loss is a good choice when the data contains both outliers and large errors

2.7.3 Loss Function for Classification

There are several loss functions that can be used for classification problems, including :

1. Binary Cross-Entropy Loss

The binary cross-entropy loss function is used for binary classification problems, where the goal is to predict one of two classes. The equation for binary cross-entropy loss is given by:

$$J(\theta) = -1/n * \Sigma(i=1 \text{ to } n) [y_i * \log(\hat{y}_i) + (1 - y_i) * \log(1 - \hat{y}_i)]$$

Where y_i is the true binary label for the i-th training example (0 or 1), $\hat{y}_i$ is the predicted probability of the positive class, and n is the number of training examples.

2. **Categorical Cross-Entropy Loss**

The categorical cross-entropy loss function is used for multiclass classification problems, where the goal is to predict one of multiple classes. The equation for categorical cross-entropy loss is given by:

$$J(\theta) = -1/n * \Sigma(i=1 \text{ to } n) \Sigma(j=1 \text{ to } C) [y_{ij} * \log(\hat{y}_{ij})]$$

Where y_{ij} is the true binary label for the j-th class of the i-th training example (0 or 1), $\hat{y}_{ij}$ is the predicted probability of the j-th class, C is the number of classes, and n is the number of training examples.

3. **Hinge Loss**

The hinge loss function is used for support vector machine (SVM) models and can be used for both binary and multiclass classification problems. The equation for hinge loss is given by:

$$J(\theta) = 1/n * \Sigma(i=1 \text{ to } n) [\max(0, 1 - y_i * \hat{y}_i)]$$

Where $y_i$is the true binary label for the i-th training example (-1 or 1), $\hat{y}_i$ is the predicted binary label (-1 or 1), and n is the number of training examples.

The choice of loss function will depend on the specific requirements of the problem and the characteristics of the data. In general, binary cross-entropy loss is a good default choice for binary classification problems, categorical cross-entropy loss is a good choice for multiclass classification problems, and hinge loss is a good choice when a linear decision boundary is desired

2.7.4 Loss Functions for Reconstruction

There are several loss functions that can be used for reconstruction problems, including:

1. **Mean Squared Error (MSE) :** The mean squared error loss function is a common choice for reconstruction problems, where the goal is to predict a target output that is as close as possible to the true target. The equation for mean squared error loss is given by:

$$J(\theta) = 1/2n * \Sigma(i=1 \text{ to } n) (\hat{y}_i - y_i)^2$$

where $\hat{y}_i$ is the predicted output for the i-th training example, y_i is the true target output for the i-th training example, and n is the number of training examples.

2. **Mean Absolute Error (MAE) :** The mean absolute error loss function is another common choice for reconstruction problems. The equation for mean absolute error loss is given by:

$$J(\theta) = 1/n * \Sigma(i=1 \text{ to } n) |\hat{y}_i - y_i|$$

where $\hat{y}_i$ is the predicted output for the i-th training example, y_i is the true target output for the i-th training example, and n is the number of training examples.

3. **Binary Cross-Entropy Loss :** The binary cross-entropy loss function can also be used for reconstruction problems where the target output is binary. The equation for binary cross-entropy loss is given by:

$$J(\theta) = -1/n * \Sigma(i=1 \text{ to } n) [y_i * \log(\hat{y}_i) + (1 - y_i) * \log(1 - \hat{y}_i)]$$

Where y_i is the true binary target output for the i-th training example (0 or 1), $\hat{y}_i$ is the predicted probability of the positive class, and n is the number of training examples.

The choice of loss function will depend on the specific requirements of the problem and the characteristics of the data. In general, mean squared error loss is a good choice when the target outputs are continuous and the magnitude of the errors is important, mean absolute error loss is a good choice when the target outputs are continuous and the magnitude of the errors is not as important, and binary cross-entropy loss is a good choice when the target outputs are binary.

2.8 Hyperparameters

- Hyperparameters are parameters in a machine learning model that are set prior to training and control the overall behavior of the model. They are different from model parameters, which are learned from data during the training process.
- Some common hyperparameters in neural networks include:
  1. **Number of layers :** The number of layers in a neural network can have a large impact on its performance. A deeper network with more layers can learn more complex representations of the data, but can also be more difficult to train due to the risk of overfitting.
  2. **Number of units per layer :** The number of units (neurons) in each layer can also impact the performance of the network. A larger number of units can allow the network to learn more complex representations of the data, but can also lead to overfitting.

3. **Learning rate :** The learning rate controls the step size used to update the model parameters during training. A smaller learning rate will cause the model to converge more slowly, but will also reduce the risk of overshooting the optimal solution. A larger learning rate will cause the model to converge more quickly, but can cause the model to oscillate or miss the optimal solution.
4. **Mini-batch size :** The mini-batch size is the number of training examples used in each iteration of stochastic gradient descent (SGD). A larger mini-batch size can speed up training, but can also increase the risk of overfitting. A smaller mini-batch size can slow down training, but can also help prevent overfitting.
5. **Regularization strength :** Regularization is a technique used to reduce overfitting in a model. Common forms of regularization in neural networks include L1 regularization, L2 regularization, and dropout. The regularization strength controls the amount of regularization applied to the model.

- The optimal values for the hyperparameters can vary greatly depending on the specific problem and the data, and finding the best values is often a process of trial and error. Hyperparameter tuning methods are grid search, random search and Bayesian optimization.

2.8.1 Learning Rate

- The learning rate is a hyperparameter in many machine learning algorithms that determines the step size at which the optimizer makes updates to the model parameters. The learning rate controls the speed at which the model learns and makes changes to the parameters.
- A high learning rate can cause the model to converge quickly but may result in overshooting the optimal solution or not converging at all. A low learning rate may converge to a better solution, but it may take a long time to do so.
- The choice of learning rate can have a significant impact on the performance of a machine learning model. A good learning rate should be set such that the model is able to make progress in a reasonable amount of time without overshooting the optimal solution.
- In some cases, a fixed learning rate may not work well for all parts of the training process. In such cases, a learning rate schedule or adaptive learning rate methods can be used to adjust the learning rate during training.
- In summary, the learning rate is a crucial hyperparameter that determines how quickly a model learns and changes its parameters during training. The right choice of learning rate can greatly impact the performance of a machine learning model.

2.8.2 Regularization of Hyperparameters

- Regularization is a technique used in machine learning to prevent over fitting, which is the phenomenon where a model fits the training data too well, but performs poorly on new, unseen data.
- Regularization works by adding a penalty term to the loss function that the model optimizes during training. The penalty term discourages the model from fitting the training data too closely, encouraging it to have a more general solution.
- There are different forms of regularization, including L1 regularization, L2 regularization, and dropout, each with its own strengths and weaknesses.

- L1 regularization adds a penalty term to the loss function that is proportional to the absolute value of the model parameters. This encourages the model to have sparse parameters, where many parameters are close to zero.
- L2 regularization adds a penalty term to the loss function that is proportional to the square of the magnitude of the model parameters. This encourages the model to have small parameters, but allows for a dense solution.
- Dropout is a regularization technique that randomly sets some of the activations in the model to zero during each forward pass. This forces the model to rely on different subsets of the neurons in each forward pass, making it more robust and less likely to overfit.
- The amount of regularization applied to a model is controlled by a hyperparameter, usually referred to as the regularization strength or regularization coefficient. The value of this hyperparameter can be determined through techniques such as cross-validation.
- In summary, regularization is a technique used in machine learning to prevent overfitting. There are different forms of regularization, including L1 regularization, L2 regularization, and dropout, each with its own strengths and weaknesses. The amount of regularization applied is controlled by a hyperparameter, and its value can be determined through techniques such as cross-validation.

2.8.3 Momentum

- Momentum is a hyperparameter in many optimization algorithms used in machine learning, including gradient descent. The momentum hyperparameter determines the extent to which the optimizer takes the past gradients into account when making updates to the model parameters.
- In gradient descent, the optimizer updates the parameters in the direction of the negative gradient of the loss function with respect to the parameters. The momentum hyperparameter controls the amount of the past gradient that is carried over to the current update.
- When the momentum is set to a high value, the optimizer takes into account a larger portion of the past gradient, which can lead to faster convergence and better performance in some cases. On the other hand, a high momentum can also cause the optimizer to oscillate and slow down convergence.
- A low momentum allows the optimizer to quickly adapt to changes in the gradient, but it can result in slow convergence in some cases. The optimal value of the momentum hyperparameter depends on the specifics of the problem and the model being optimized.
- In summary, momentum is a hyperparameter in many optimization algorithms used in machine learning, including gradient descent. The momentum hyperparameter determines the extent to which the optimizer takes the past gradients into account when making updates to the model parameters. The optimal value of the momentum hyperparameter depends on the specifics of the problem and the model being optimized.

2.8.4 Sparsity in Hyperparameters

- Sparsity is a property of a model in which many of the model parameters have a value close to zero. In the context of machine learning, sparsity can be used as a form of regularization to prevent overfitting and improve the interpretability of a model.
- There are several ways to encourage sparsity in a model, including L1 regularization and sparse constraint methods.

- L1 regularization adds a penalty term to the loss function proportional to the absolute value of the parameters. This penalty encourages the model to have sparse parameters, where many parameters are close to zero.
- Sparse constraint methods enforce a hard constraint on the number of non-zero parameters in the model. For example, a sparsity constraint may enforce that only a fixed percentage of the parameters can be non-zero. This forces the model to use only a small subset of the parameters, encouraging sparsity.
- The amount of sparsity in a model is controlled by a hyperparameter, which determines the strength of the sparsity constraint or regularization. The optimal value of this hyperparameter depends on the specifics of the problem and the model being optimized.
- In summary, sparsity is a property of a model in which many of the parameters have a value close to zero. Sparsity can be used as a form of regularization to prevent overfitting and improve the interpretability of a model. There are several ways to encourage sparsity in a model, including L1 regularization and sparse constraint methods, and the amount of sparsity is controlled by a hyperparameter.

2.9 Deep Feed Forward Network

- A deep feedforward network, also known as a fully connected network, is a type of artificial neural network that is composed of multiple layers of interconnected artificial neurons, or nodes. In a deep feedforward network, information flows through the network in a straightforward manner, from the input layer to the output layer, without looping back.
- The nodes in each layer are connected to the nodes in the next layer, and the connections between the nodes have weights that can be adjusted during the training process. The weights represent the strength of the connection between two nodes and determine the flow of information through the network.
- The nodes in the input layer receive input from the outside world, while the nodes in the output layer produce the network's predictions. The nodes in the intermediate layers, also known as hidden layers, perform computations to extract features from the input data.
- Deep feedforward networks can be used to perform a variety of tasks, including classification, regression, and generation. By using multiple hidden layers, a deep feedforward network can learn complex representations of the input data and make highly accurate predictions.
- In summary, a deep feedforward network is a type of artificial neural network that is composed of multiple layers of interconnected artificial neurons. Information flows through the network in a straightforward manner, from the input layer to the output layer, without looping back. The nodes in the network perform computations to extract features from the input data and make predictions.

2.9.1 Ex OR Deep Feed Forward Network

- The XOR (exclusive or) problem is a simple binary classification problem in which the goal is to classify a given input into one of two classes, based on whether the input satisfies the XOR relationship or not. The XOR relationship states that an input is in one class if exactly one of its two input values is 1, and in the other class otherwise.
- A deep feedforward network can be used to solve the XOR problem by learning a non-linear decision boundary that separates the two classes. This can be achieved by using multiple hidden layers in the network, which allow the network to learn complex representations of the input data.

- The input to the network consists of two binary values, which are fed into the input layer. The hidden layers perform computations to extract features from the input data and produce a high-dimensional representation of the input. The output layer uses the representation produced by the hidden layers to make a prediction about the class of the input.
- The weights of the connections between the nodes are adjusted during the training process so that the network can accurately classify the inputs. The training process involves adjusting the weights to minimize the difference between the network's predictions and the actual class labels of the training examples.
- In summary, the XOR problem is a binary classification problem in which the goal is to classify a given input based on whether it satisfies the XOR relationship or not. A deep feedforward network can be used to solve the XOR problem by learning a non-linear decision boundary that separates the two classes. The network is trained by adjusting the weights of the connections between the nodes so that it can accurately classify the inputs.
- A hidden unit in a deep feedforward network is a node or neuron in the hidden layer(s) of the network. It receives inputs from the previous layer, processes the information using an activation function, and passes the result to the next layer as output. The hidden units work together to perform complex non-linear transformations on the inputs, allowing the network to learn intricate relationships between inputs and outputs. The number of hidden units in a network can have a significant impact on its ability to learn and generalize to new data

2.10 Cost Function

- The cost function in a Deep Feedforward Neural Network (DFNN) is a measure of the difference between the predicted output and the true output. It is used to guide the learning process and to determine the optimal weights for the network. The cost is calculated for each training example and the average cost over the entire training set is used to update the weights.
- The cost function (also known as loss or objective function) in neural networks is a measure of how well the model's predictions match the actual (ground truth) values. It is used during the training phase to adjust the model's weights and biases to minimize the error between predicted values and actual values. The specific cost function used depends on the type of problem being solved, such as classification or regression, and the activation function used in the output layer. Common cost functions include mean squared error, cross-entropy loss, and binary cross-entropy loss.
- Some commonly used cost functions for DFNNs are :

1. **Mean Squared Error (MSE) :** This measures the average squared difference between the predicted and actual output.
2. **Binary Cross-Entropy :** This is used in binary classification problems, where the goal is to predict one of two possible outcomes.
3. **Categorical Cross-Entropy :** This is used in multi-class classification problems, where the goal is to predict one of multiple possible outcomes.
4. **Hinge Loss :** This is used in support vector machines (SVMs) for binary classification problems.
5. **SoftMax Loss :** This is used in multi-class classification problems with the SoftMax activation function.

The choice of cost function depends on the specific problem and the desired output of the network.

2.11 Error Backpropagation

- Backpropagation is the process of calculating the gradients of the cost function with respect to the weights in a Deep Feedforward Neural Network (DFNN). The gradients are then used to update the weights during the training process. The goal is to minimize the cost function by finding the optimal weights.
- The backpropagation algorithm works by first forward propagating the input through the network to calculate the predicted output. Then, the error between the predicted output and the true output is calculated using the cost function. This error is then backpropagated through the network, starting from the output layer and working backwards to the input layer, to calculate the gradients of the cost with respect to each weight in the network.
- Once the gradients have been calculated, the weights are updated in the direction of the negative gradients, i.e., in the direction that will reduce the cost. This process is repeated multiple times, until the cost reaches a minimum value or the performance on a validation set plateaus.
- Backpropagation is the core of the learning process in DFNNs and is used in most deep learning algorithms.

2.12 Gradient based Learning

- Gradient-based learning is a method for optimizing the weights in a Deep Feedforward Neural Network (DFNN) by minimizing the cost function. The cost function measures the difference between the predicted output and the true output, and the goal of the training process is to find the optimal weights that minimize this cost.
- Gradient-based learning algorithms use the gradients of the cost with respect to the weights to update the weights. The gradients are calculated using the backpropagation algorithm, which involves forward propagating the input through the network to calculate the predicted output and then backpropagating the error between the predicted output and the true output to calculate the gradients.
- Once the gradients have been calculated, the weights are updated in the direction of the negative gradients, i.e., in the direction that will reduce the cost. This process is repeated multiple times, until the cost reaches a minimum value or the performance on a validation set plateaus.
- Gradient-based learning is widely used in deep learning because it is simple, efficient, and can handle high-dimensional problems. The optimization process is performed using optimization algorithms, such as gradient descent, stochastic gradient descent, Adam, and others. These algorithms differ in how they update the weights based on the gradients, but the overall goal is the same: to find the optimal weights that minimize the cost function.

2.12.1 Implementing Gradient Descent DFNN

Here is an example of how to implement the gradient descent optimization algorithm in a Deep Feedforward Neural Network (DFNN) in Python:

```
import numpy as np
def sigmoid(x):
return 1/(1 + np.exp(-x))
defsigmoid_derivative(x):
return x * (1 - x)
```

```
class DFNN:
def __init__(self, input_size, hidden_size, output_size):
self.input_size = input_size
self.hidden_size = hidden_size
self.output_size = output_size
# Initialize the weights
self.weights_input_to_hidden = np.random.normal(0, self.input_size**-0.5, (self.input_size, self.hidden_size))
self.weights_hidden_to_output = np.random.normal(0, self.hidden_size**-0.5,
(self.hidden_size, self.output_size))
defforward_propagate(self, X):
self.hidden_layer_input = np.dot(X, self.weights_input_to_hidden)
self.hidden_layer_output = sigmoid(self.hidden_layer_input)
self.output_layer_input = np.dot(self.hidden_layer_output, self.weights_hidden_to_output)
self.output = sigmoid(self.output_layer_input) return
self.output
defbackward_propagate(self, X, y, learning_rate):
output_error = y - self.output
output_delta = output_error * sigmoid_derivative(self.output)
hidden_error = np.dot(output_delta, self.weights_hidden_to_output.T)
hidden_delta = hidden_error * sigmoid_derivative(self.hidden_layer_output)
self.weights_hidden_to_output += learning_rate * np.dot(self.hidden_layer_output.T, output_delta)
self.weights_input_to_hidden += learning_rate * np.dot(X.T, hidden_delta)
def train(self, X, y, learning_rate, epochs):
fori in range(epochs):
self.forward_propagate(X)
self.backward_propagate(X, y, learning_rate)
```

- In this example, the DFNN class is implemented with the forward propagate method for forward propagation, the backward propagate method for backpropagation, and the train method for training the network. The sigmoid function and its derivative, sigmoid_derivative, are used as activation functions. The weights are initialized randomly, and the train method performs the optimization by updating the weights in the direction of the negative gradients, using the learning rate specified.
- This is just a simple example, and the architecture, activation functions, and optimization algorithm can be adjusted depending on the specific problem and desired outcome.

2.12.2 Vanishing and Exploding Gradients

- The vanishing and exploding gradients are two common problems in deep learning, particularly in Deep Feedforward Neural Networks (DFNNs).
- The vanishing gradient problem occurs when the gradients become very small during backpropagation, making it difficult for the optimization algorithm to update the weights effectively. This results in slow convergence or poor performance. The vanishing gradient problem is often encountered in deep networks because the gradients are

multiplied multiple times during backpropagation, making them smaller and smaller as they move from the output layer to the input layer.

- The exploding gradient problem occurs when the gradients become very large during backpropagation, causing the optimization algorithm to overshoot and produce large, unstable updates to the weights. This results in unstable convergence or divergence. The exploding gradient problem is often encountered when the activation functions used in the network have high values, leading to large gradients during backpropagation.
- To avoid the vanishing and exploding gradient problems, several techniques have been developed, including weight initialization, activation functions with bounded outputs, normalization techniques, and gradient clipping. Weight initialization methods like Glorot and He initialization try to balance the scale of the gradients in both the forward and backward passes. Activation functions like ReLU and leaky ReLU have outputs that are limited in magnitude, preventing the gradients from exploding. Normalization techniques like batch normalization help to keep the inputs to the activation functions in a reasonable range. Gradient clipping involves limiting the magnitude of the gradients, which can help to prevent the gradients from exploding.
- Choosing the appropriate combination of these techniques depends on the specific problem and network architecture, and may require some experimentation to find the best solution

2.13 Sentiment Analysis

- Sentiment analysis is the process of determining the sentiment expressed in a piece of text, such as a review, tweet, or news article. The goal of sentiment analysis is to categorize the text as positive, negative, or neutral in terms of the sentiment expressed.
- Sentiment analysis is a common task in Natural Language Processing (NLP) and is used in a wide range of applications, including customer feedback analysis, brand monitoring, and opinion mining.
- There are several approaches to sentiment analysis, including rule-based methods, which use a set of predefined rules to categorize the text, and machine learning-based methods, which train models on large annotated datasets to make predictions about the sentiment of new texts.
- Machine learning-based approaches to sentiment analysis typically involve preprocessing the text to remove stop words, stemming, and lemmatization, and then converting the text into numerical features that can be input into a model. Commonly used models for sentiment analysis include Naive Bayes, decision trees, random forests, and neural networks.
- Sentiment analysis can be a challenging task due to the subjectivity of language and the nuances of sentiment expression. The accuracy of sentiment analysis models can be improved through the use of large annotated datasets, transfer learning, and the combination of multiple models.
- Deep Feedforward Neural Networks (DFNNs) can be used for sentiment analysis as a form of machine learning-based approach.
- In a DFNN for sentiment analysis, the input to the network is a representation of the text, such as a bag of words representation or a word embedding. The output of the network is a predicted sentiment label, such as positive, negative, or neutral.
- To train a DFNN for sentiment analysis, a labeled dataset of texts and their corresponding sentiments is used. The network is trained to minimize the difference between its predictions and the true sentiments. During the training

process, the weights of the network are updated using an optimization algorithm such as gradient descent to minimize the error.

- Once the DFNN is trained, it can be used to predict the sentiment of new texts. The DFNN can be fine-tuned on domain-specific datasets to improve its accuracy for a specific application.
- DFNNs are powerful models that can capture complex relationships between the input features and the sentiment labels. However, they can be computationally expensive to train and may require large labeled datasets to achieve good performance. In addition, the interpretability of DFNNs can be limited, making it difficult to understand why a particular prediction was made.

2.14 PyTorch

- PyTorch is an open-source machine learning library that provides a high-level interface for building and training deep learning models. PyTorch is designed for ease of use and speed, making it a popular choice for both researchers and practitioners in the deep learning community.
- With PyTorch, building and training a deep learning model can be done in a few lines of code. The library provides a range of pre-built neural network architectures, including feedforward networks, convolutional neural networks (CNNs), recurrent neural networks (RNNs), and transformer networks. In addition, PyTorch provides tools for data loading and preprocessing, as well as for model evaluation and prediction.
- To get started with PyTorch, you will need to install the library and familiarize yourself with the basic concepts of tensors, automatic differentiation, and model building. Once you have a basic understanding of these concepts, you can start building and training your own deep learning models with PyTorch.
- To train a model, you will need to define a loss function, an optimizer, and a training loop. The loss function measures the difference between the model's predictions and the true labels, while the optimizer updates the model weights to minimize the loss. The training loop iterates over the training data, updating the model weights and evaluating the performance of the model on a validation set.
- Once the model is trained, you can use it to make predictions on new data. PyTorch also provides tools for deploying models to production, including exporting models to ONNX format, which can be used in a wide range of production environments.
- Overall, PyTorch is a flexible and user-friendly library for building and training deep learning models, and it is a great choice for those looking to get started with deep learning.

2.15 Jupyter

- Jupyter is an open-source web-based platform for interactive computing that is commonly used for data science and machine learning tasks, including deep learning. With Jupyter, you can create and share documents that contain live code, equations, visualizations, and narrative text.
- Jupyter supports a wide range of programming languages, including Python, which is a popular choice for deep learning. With Jupyter, you can write and run Python code directly in the browser, making it easy to experiment with different deep learning models and algorithms.

- To use Jupyter for deep learning, you will need to install the Jupyter platform, as well as any necessary deep learning libraries, such as PyTorch or TensorFlow. Once you have Jupyter and the necessary libraries installed, you can start building and training deep learning models directly in the Jupyter environment.
- Jupyter provides a range of tools for visualizing and exploring your data, including matplotlib and Seaborn for data visualization, and pandas for data manipulation. These tools can be used to preprocess your data, visualize your results, and explore your models in real-time.
- Jupyter also supports interactive widgets, which allow you to control your models using sliders, drop-down menus, and other interactive elements. This makes it easy to experiment with different hyperparameters and model configurations, and to explore the results of your experiments.
- Overall, Jupyter is a powerful platform for deep learning, and it is widely used by data scientists and machine learning practitioners. Its combination of interactive computing and data visualization makes it an ideal environment for building and experimenting with deep learning models.

2.16 Colab

- Google Colab is a free, cloud-based platform for machine learning and deep learning research. It allows you to run Jupyter notebooks in the cloud, eliminating the need to install and maintain software and hardware locally.
- Colab provides access to GPUs and TPUs, making it easy to run large and computationally intensive deep learning models. You can also connect to Google Drive to access your data, and you can save and share your notebooks with others.
- To get started with Colab, you will need to sign up for a Google account and create a new Colab notebook. From there, you can write and run code, including deep learning models using libraries such as PyTorch or TensorFlow.
- Colab also provides integration with Google Drive, allowing you to save your notebooks and data to the cloud, and to collaborate with others in real-time. You can also export your notebooks to GitHub, making it easy to share your code and models with others.
- In addition to its computational capabilities, Colab provides a range of pre-installed libraries and tools, including popular machine learning libraries such as scikit-learn, and visualization tools such as matplotlib. This makes it easy to get started with deep learning, and to experiment with different models and algorithms.
- Overall, Google Colab is a powerful and accessible platform for deep learning research, and it is widely used by researchers and practitioners in the machine learning community.

A deep feedforward neural network (DFNN) can be used for music genre classification, which involves classifying music into various categories or genres. Here's an overview of how a DFNN could be used for this task:

1. **Data collection :** First, a dataset of music samples and their corresponding genre labels must be collected. This dataset can be created by downloading audio files from the internet and labeling them based on the genre.
2. **Data preprocessing :** The audio files must then be preprocessed to extract relevant features, such as spectral characteristics or rhythm patterns. These features will be used as input to the DFNN.
3. **Model architecture :** Next, a DFNN architecture must be designed, taking into account the number of input features, the number of hidden layers, and the number of neurons in each layer.
4. **Model training :** The DFNN must then be trained using the preprocessed data, using an optimization algorithm such as gradient descent to minimize the cost function.

We Do not Support Piracy. Please Purchase the book if you can.

SUBJECT CODE : 410251

As per Revised Syllabus of

SAVITRIBAI PHULE PUNE UNIVERSITY

Choice Based Credit System (CBCS)

B.E. (Computer) Semester : VIII

DEEP LEARNING

Iresh A. Dhotre

M.E. (Information Technology)

Ex-Faculty, Sinhgad College of Engineering,

Pune.

Abhijit D. Jadhav

Ph.D. CSE Scholar, M.Tech.(CSE),

B.E. (Computer Engineering),

Memberships : LMISTE, IAENG, ACM,

Assistant Professor, Department of Computer Engineering,

Pimpri Chinchwad College of Engineering and Research,

Ravet, Pune.

TECHNICAL PUBLICATIONS

SINCE 1993

An Up Thrust for Knowledge

(i)

DEEP LEARNING

Subject Code : 410251

B.E. (Computer Engineering) Semester : VIII

© Copyright with Authors

All publishing rights (printed and ebook version) reserved with Technical Publications. No part of this book should be reproduced in any form, Electronic, Mechanical, Photocopy or any information storage and retrieval system without prior permission in writing, from Technical Publications, Pune.

Published by :

TECHNICAL PUBLICATIONS

SINCE 1993

An Up Thrust for Knowledge

Amit Residency, Office No. 1, 412, Shaniwar Peth,

Pune - 411030, M.S. INDIA Ph. : +91-020-24495496/97

Email : info@technicalpublications.in Website : www.technicalpublications.in

Printer :

Yogiraj Printers & Binders

Sr. No. 10/1A,

Ghule Industrial Estate, Nanded Village Road,

Tal. - Haveli, Dist. - Pune - 411041

ISBN 978-93-8483-488-4

APRIL 19

(ii)

SYLLABUS

Deep Learning - (410251)

Credit	Examination Scheme :
03	In-Sem (Paper) : 30 Marks End-Sem (Paper) : 70 Marks

Unit I Foundations of Deep learning

What is machine learning and deep learning ?,Supervised and Unsupervised Learning, bias variance tradeoff, hyper parameters, under/over fitting regularization, Limitations of machine learning, History of deep learning, Advantage and challenges of deep learning. Learning representations from data, Understanding how deep learning works in three figures, Common Architectural Principles of Deep Network, Architecture Design, Applications of Deep learning, Introduction and use of popular industry tools such as TensorFLow, Keras, PyTorch, Caffe, Shogun. **(Chapter - 1)**

Unit II Deep Neural Networks (DNNs)

Introduction to Neural Networks : The Biological Neuron, The Perceptron, Multilayer Feed-Forward Networks, **Training Neural Networks :** Backpropagation and Forward propagation **Activation Functions :** Linear, Sigmoid, Tannh, Hard Tanh, Softmax, Rectified Linear, **Loss Functions :** Loss Function Notation, Loss Functions for Regression, Loss Functions for Classification, Loss Functions for Reconstruction, **Hyperparameters :** Learning Rate, Regularization, Momentum, Sparsity, Deep Feedforward Networks - Example of Ex OR, Hidden Units, cost functions, error backpropagation, Gradient-Based Learning, Implementing Gradient Descent, vanishing and Exploding gradient descent, Sentiment Analysis, Deep Learning with Pytorch, Jupyter, colab. **(Chapter - 2)**

Unit III Convolution Neural Network (CNN)

Introduction, CNN architecture overview, The Basic Structure of a Convolutional Network-Padding, Strides, Typical Settings, the ReLU layer, Pooling, Fully Connected Layers, The Interleaving between Layers, Local Response Normalization, Training a Convolutional Network. **(Chapter - 3)**

Unit IV Recurrent Neural Networks (RNN)

Recurrent and Recursive Nets : Unfolding Computational Graphs, Recurrent Neural Networks, Bidirectional RNNs, Encoder-Decoder Sequence-to-Sequence Architectures, Deep Recurrent Networks, Recursive Neural Networks, The Challenge of Long-Term Dependencies, Echo State Networks, Leaky Units and Other Strategies for Multiple Time Scales, The Long Short-Term Memory and Other Gated RNNs, Optimization for Long-Term Dependencies, Explicit Memory. **Practical Methodology :** Performance Metrics, Default Baseline Models, Determining Whether to Gather More Data, Selecting Hyper parameters. **(Chapter - 4)**

Unit V Deep Generative Models

Introduction to deep generative model, Boltzmann Machine, Deep Belief Networks, Generative adversarial network (GAN), discriminator network, generator network, types of GAN, Applications of GAN networks. **(Chapter - 5)**

Unit VI Reinforcement Learning

Introduction of deep reinforcement learning, Markov Decision Process, basic framework of reinforcement learning, challenges of reinforcement learning, Dynamic programming algorithms for reinforcement learning, Q Learning and Deep Q-Networks, Deep Q recurrent networks, Simple reinforcement learning for Tic-Tac-Toe. **(Chapter - 6)**

Scanned with OKEN Scanner

Unit I

1 Foundations of Deep Learning

Syllabus

What is machine learning and deep learning ? Supervised and Unsupervised Learning, bias variance tradeoff, hyper parameters, under/over fitting regularization, Limitations of machine learning, History of deep learning, Advantage and challenges of deep learning, Learning representations from data, Understanding how deep learning works in three figures, Common Architectural Principles of Deep Network, Architecture Design, Applications of Deep learning, Introduction and use of popular industry tools such as TensorFLow, Keras, PyTorch, Caffe, Shogun.

Contents

1.1 Introduction to Machine Learning

1.2 What is Deep Learning ?

1.3 Supervised Learning

1.4 Unsupervised Learning

1.5 Bias Variance Tradeoff

1.6 Generalization : Overfitting vs Underfitting

1.7 History of Deep Learning

1.8 Learning Representations from Data

1.9 Understanding How Deep Learning Works in Three Figures

1.10 Common Architectural Principles of Deep Network

1.11 Architecture Design

1.12 Applications of Deep Learning

1.13 Introduction and use of Popular Industry Tools

1.1 Introduction to Machine Learning

- Machine Learning (ML) is a sub-field of Artificial Intelligence (AI) which concerns with developing computational theories of learning and building learning machines.
- **Learning** is a phenomenon and process which has manifestations of various aspects. Learning process includes gaining of new symbolic knowledge and development of cognitive skills through instruction and practice. It is also discovery of new facts and theories through observation and experiment.
- Machine Learning Definition : A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.
- Machine learning is programming computers to optimize a performance criterion using example data or past experience. Application of machine learning methods to large databases is called **data mining**.
- It is very hard to write programs that solve problems like recognizing a human face. We do not know what program to write because we don't know how our brain does it. Instead of writing a program by hand, it is possible to collect lots of examples that specify the correct output for a given input.
- A machine learning algorithm then takes these examples and produces a program that does the job. The program produced by the learning algorithm may look very different from a typical hand-written program. It may contain millions of numbers. If we do it right, the program works for new cases as well as the ones we trained it on.
- Main goal of machine learning is to devise learning algorithms that do the learning automatically without human intervention or assistance. The machine learning paradigm can be viewed as "programming by example." Another goal is to develop computational models of human learning process and perform computer simulations.
- The goal of machine learning is to build computer systems that can adapt and learn from their experience.
- Algorithm is used to solve a problem on computer. An algorithm is a sequence of instruction. It should carry out to transform the input to output. For example, for addition of four numbers is carried out by giving four number as input to the algorithm and output is sum of all four numbers. For the same task, there may be various algorithms. It is interested to find the most efficient one, requiring the least number of instructions or memory or both.
- For some tasks, however, we do not have an algorithm.

Scanned with OKEN Scanner

Why Is Machine Learning Important ?

- Machine learning algorithms can figure out how to perform important tasks by generalizing from examples.
- Machine Learning provides business insight and intelligence. Decision makers are provided with greater insights into their organizations. This adaptive technology is being used by global enterprises to gain a competitive edge.
- Machine learning algorithms discover the relationships between the variables of a system (input, output and hidden) from direct samples of the system.
- **Following are some of the reasons :**
  1. Some tasks cannot be defined well, except by examples. For example: recognizing people.
  2. Relationships and correlations can be hidden within large amounts of data. To solve these problems, machine learning and data mining may be able to find these relationships.
  3. Human designers often produce machines that do not work as well as desired in the environments in which they are used.
  4. The amount of knowledge available about certain tasks might be too large for explicit encoding by humans.
  5. Environments change time to time.
  6. New knowledge about tasks is constantly being discovered by humans.
- Machine learning also helps us find solutions of many problems in computer vision, speech recognition and robotics. Machine learning uses the theory of statistics in building mathematical models, because the core task is making inference from a sample.

How Machines Learn ?

- Machine learning typically follows three phases :
  1. **Training :** A training set of examples of correct behavior is analyzed and some representation of the newly learnt knowledge is stored. This is some form of rules.
  2. **Validation :** The rules are checked and, if necessary, additional training is given. Sometimes additional test data are used, but instead, a human expert may validate the rules, or some other automatic knowledge - based component may be used. The role of the tester is often called the opponent.
  3. **Application :** The rules are used in responding to some new situation.

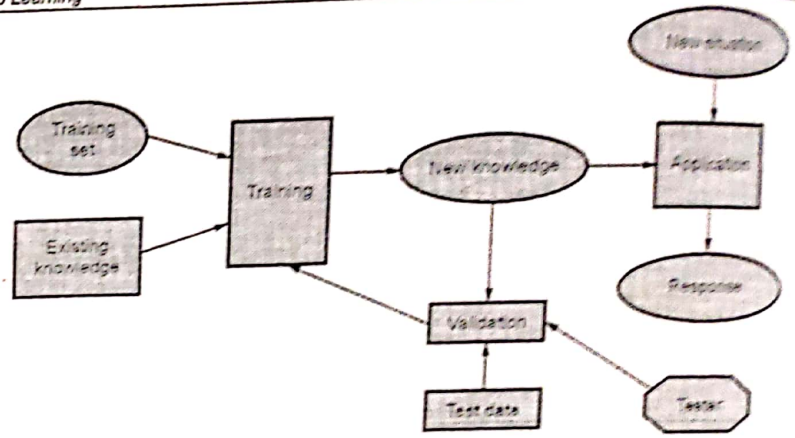


Fig. 1.1.1 Phases of ML

1.1.1 Why Machine Learning is Important ?

- Machine learning algorithms can figure out how to perform important tasks by generalizing from examples.
- Machine learning provides business insight and intelligence. Decision makers are provided with greater insights into their organizations. This adaptive technology is being used by global enterprises to gain a competitive edge.
- Machine learning algorithms discover the relationships between the variables of a system (input, output and hidden) from direct samples of the system.
- **Following are some of the reasons :**
  1. Some tasks cannot be defined well, except by examples. For example : Recognizing people.
  2. Relationships and correlations can be hidden within large amounts of data. To solve these problems, machine learning and data mining may be able to find these relationships.
  3. Human designers often produce machines that do not work as well as desired in the environments in which they are used.
  4. The amount of knowledge available about certain tasks might be too large for explicit encoding by humans.
  5. Environments change time to time.
  6. New knowledge about tasks is constantly being discovered by humans.

Scanned with OKEN Scanner

- Machine learning also helps us find solutions of many problems in computer vision, speech recognition and robotics. Machine learning uses the theory of statistics in building mathematical models, because the core task is making inference from a sample.
- Learning is used when :
  1. Human expertise does not exist (navigating on Mars).
  2. Humans are unable to explain their expertise (speech recognition).
  3. Solution changes in time (routing on a computer network).
  4. Solution needs to be adapted to particular cases (user biometrics).

1.2 What is Deep Learning ?

- The term "deep" usually refers to the number of hidden layers in the neural network.
- Deep learning is a subset of machine learning, which is predicated on idea of learning from example. In machine learning, instead of teaching a computer a massive list of rules to solve the problem, we give it a model with which it can evaluate examples, and a small set of instructions to modify the model when it makes a mistake.
- The basic idea of deep learning is that repeated composition of functions can often reduce the requirements on the number of base functions (computational units) by a factor that is exponentially related to the number of layers in the network.
- Deep learning eliminates some of data pre-processing that is typically involved with machine learning.
- Fig. 1.2.1 shows relation between AI, ML and Deep learning.

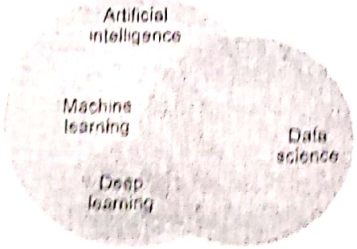


Fig. 1.2.1 Relation between AI, ML and Deep learning

- For example, let's say that we had a set of photos of different pets, and we wanted to categorize by "cat" and "dog". Deep learning algorithms can determine which features (e.g. ears) are most important to distinguish each animal from another. In machine learning, this hierarchy of features is established manually by a human expert.
- In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.
- Deep learning classifies information through layers of neural networks, which have a set of inputs that receive raw data. For example, if a neural network is trained with images of birds, it can be used to recognize images of birds. More layers enable more precise results, such as distinguishing a crow from a raven as compared to distinguishing a crow from a chicken.
- Deep Learning consists of the following methods and their variations :
  a) Unsupervised learning systems such as Boltzman machines for preliminary training, auto-encoders, generative adversarial network.
  b) Supervised learning such as Convolution neural networks which brought technoogy of pattern recognition to a new level.
  c) Recurrent neural networks, allowing to train on processes in time.
  d) Recursive neural networks, allowing to include feedback between circuit elements and chains.

1.2.1 Reasons for Using Deep Learning

1. **Analyzing unstructured data :** Deep learning algorithms can be trained to look at text data by analyzing social media posts, news, and surveys to provide valuable business and customer insights.
2. **Data labelling :** Deep learning requires labeled data for training. Once trained, it can label new data and identify different types of data on its own.
3. **Feature engineering :** A deep learning algorithm can save time because it does not require humans to extract features manually from raw data.
4. **Efficiency :** When a deep learning algorithm is properly trained, it can perform thousands of tasks over and over again, faster than humans.

Scanned with OKEN Scanner

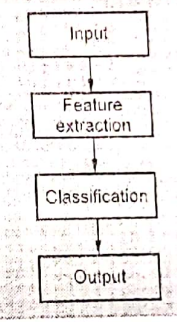
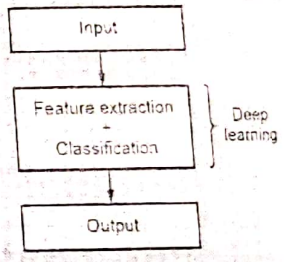
5. **Training :** The neural networks used in deep learning have the ability to be applied to many different data types and applications. Additionally, a deep learning model can adapt by retraining it with new data.

1.2.2 Application of Deep Learning

1. **Aerospace and defense :** Deep learning is utilized extensively to help satellites identify specific objects or areas of interest and classify them as safe or unsafe for soldiers.
2. **Financial services :** Financial institutions regularly use predictive analytics to drive algorithmic trading of stocks, assess business risks for loan approvals, detect fraud, and help manage credit and investment portfolios for clients.
3. **Medical research :** The medical research field uses deep learning extensively. For example, in ongoing cancer research, deep learning is used to detect the presence of cancer cells automatically.
4. **Industrial automation :** The heavy machinery sector is one that requires a large number of safety measures. Deep learning helps with the improvement of worker safety in such environments by detecting any person or objects that comes within the unsafe radius of a heavy machine.
5. **Facial recognition :** This feature utilizing deep learning is being used not just for a range of security purposes but will soon enable purchases at stores. Facial recognition is already being extensively used in airports to enable seamless, paperless check-ins.

1.2.3 Difference between Machine Learning and Deep Learning

Sr. No.	Machine Learning	Deep Learning
1.	Machine learning uses algorithms to parse data, learn from that data, and make informed decisions based on what it has learned.	Deep learning structures algorithms in layers to create an "artificial neural network" that can learn and make intelligent decisions on its own.
2.	Machine learning gives lesser accuracy.	Deep learning gives more accuracy.

3.	Machine learning requires less time for training.	Deep learning requires more time for training.
4.	Needs accurately identified features by human intervention.	It can create new features.
5.	Machine learning models mostly require data in a structured form.	Deep Learning models can work with structured and unstructured data both as they rely on the layers of the Artificial neural network.
6.	Algorithms are detected by data analysts to examine specific variables in data sets.	Algorithms are largely self-directed on data analysis once they are put into production.
7.	Machine learning can work on low-end machines.	Deep learning model needs a huge amount of data to work efficiently, so they need GPU's and hence the high-end machine
8.		

1.2.4 Difference between ML, AI and Data Science

Sr. No.	Machine Learning	Artificial Intelligence	Data Science
1.	Focuses on providing a means for algorithms and systems to learn from experience with data and use that experience to improve over time.	Focuses on giving machines cognitive and intellectual capabilities similar to those of humans.	Focuses on extracting information needles from data haystacks to aid in decision-making and planning.

Scanned with OKEN Scanner

2.	Machine Learning uses statistical models.	Artificial Intelligence uses logic and decision trees.	Data Science deals with structured data.
3.	A form of analytics in which software programs learn about data and find patterns.	Development of computerized applications that simulate human intelligence and interaction.	The process of using advanced analytics to extract relevant information from data.
4.	Objective is to maximize accuracy.	Objective is to maximize the chance of success.	Objective is to extract actionable insights from the data.
5.	ML can be done through supervised, unsupervised or reinforcement learning approaches.	AI encompasses a collection of intelligence concepts, including elements of perception, planning and prediction.	Uses statistics, mathematics, data wrangling, big data analytics, machine learning and various other methods to answer analytics questions.
6.	ML is concerned with knowledge accumulation.	AI is concerned with knowledge dissemination and conscious machine actions.	Data science is all about data engineering.

1.2.5 Difference between AI, ML and Deep Learning

Sr. No.	AI	ML	DL
1.	AI aims towards building machines that are capable to think like humans.	ML aims to learn through data to solve the problem.	DL aims to build neural networks that automatically discover patterns for feature detection.
2.	AI is subset of data science.	ML is subset of AI and data science.	DL is subset of AI, ML and data science.

3.	All systems of artificial intelligence fall into three types : a) Artificial Narrow Intelligence b) Artificial General Intelligence c) Artificial Super Intelligence	ML algorithms can be broadly classified into three categories a) Supervised learning b) Unsupervised learning c) Reinforcement learning	Deep learning architectures are as follows : a) Convolutional Neural Networks b) Recurrent Neural Networks c) Recursive Neural Networks
4.	Making machines intelligent may or may not need high computational power.	These algorithms can work easily on normal low performance computers without GPUs.	Algorithms are dependent on high performance hardware components that include GPUs.

1.2.6 Advantages and Disadvantages of Deep Learning

Advantages of Deep Learning

- No need for feature engineering.
- DL solves the problem on the end-to-end basis.
- Deep learning gives more accuracy.

Disadvantages of Deep Learning

- DL needs high-performance hardware.
- DL needs much more time to train.
- It is very difficult to assess its performance in real world applications.
- It is very hard to understand.

1.3 Supervised Learning

- Supervised learning is the machine learning task of inferring a function from supervised training data. The training data consist of a set of training examples. The task of the supervised learner is to predict the output behavior of a system for any set of input values, after an initial training phase.
- **Supervised learning** in which the network is trained by providing it with input and matching output patterns. These input-output pairs are usually provided by an external teacher.

Scanned with OKEN Scanner

- Human learning is based on the past experiences. A computer does not have experiences.
- A computer system learns from data, which represent some "past experiences" of an application domain.
- To learn a target function that can be used to predict the values of a discrete class attribute, e.g., approve or not-approved and high-risk or low risk. The task is commonly called : Supervised learning, Classification or inductive learning.
- Training data includes both the input and the desired results. For some examples the correct results (targets) are known and are given in input to the model during the learning process. The construction of a proper training, validation and test set is crucial. These methods are usually fast and accurate.
- Have to be able to generalize : give the correct results when new data are given in input without knowing a priori the target.
- Supervised learning is the machine learning task of inferring a function from supervised training data. The training data consist of a set of training examples. In supervised learning, each example is a pair consisting of an input object and a desired output value.
- A supervised learning algorithm analyzes the training data and produces an inferred function, which is called a classifier or a regression function. Fig. 1.3.1 shows supervised learning process.

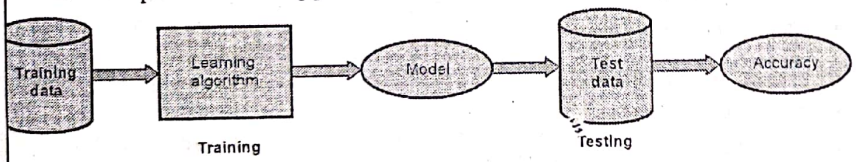


Fig. 1.3.1 Supervised learning process

- The learned model helps the system to perform task better as compared to no learning.
- Each input vector requires a corresponding target vector.

  Training Pair = (Input Vector, Target Vector)
- Fig. 1.3.2 shows input vector. (See Fig. 1.3.2 on next page)

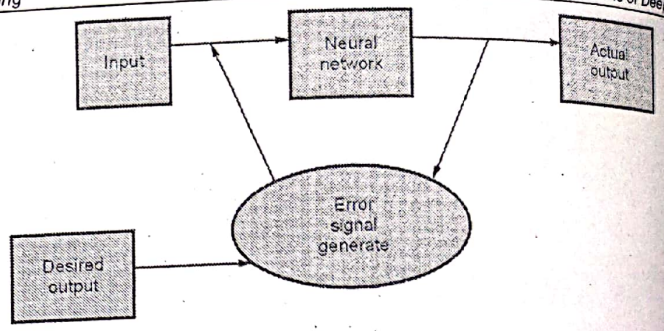


Fig. 1.3.2 Input vector

- Supervised learning denotes a method in which some input vectors are collected and presented to the network. The output computed by the network is observed and the deviation from the expected answer is measured. The weights are corrected according to the magnitude of the error in the way defined by the learning algorithm.
- Supervised learning is further divided into methods which use reinforcement or error correction. The perceptron learning algorithm is an example of supervised learning with reinforcement.

Data formats in supervised learning :

- Supervised learning always uses a dataset to define finite set of real vectors with m features each :

$$X = \{\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n\} \text{ where } \bar{x}_i \in \Re^m$$

- Considering that user approach is always probabilistic, we need to consider each X as drawn from a statistical multivariate distribution D. It is also useful to add an important condition upon the whole dataset X. Here we consider that all samples to be independent and identically distributed. This means all variables belong to the same distribution D and considering an arbitrary subset of m values, it happens that :

$$P(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_m) = \prod_{i=1}^{m} P(\bar{x}_i)$$

- The corresponding output values can be both numerical - continuous and categorical. In the first case, the process is called regression, while in the second, it is called classification.

Scanned with OKEN Scanner

- Example : Dataset contains city populations by year for the past 100 years and user want to know what the population of a specific city will be four years from now. The outcome uses labels that already exist in the data set : population, city and year.
- In order to solve a given problem of supervised learning, following steps are performed :
  1. Find out the type of training examples.
  2. Collect a training set.
  3. Determine the input feature representation of the learned function.
  4. Determine the structure of the learned function and corresponding learning algorithm.
  5. Complete the design and then run the learning algorithm on the collected training set.
  6. Evaluate the accuracy of the learned function. After parameter adjustment and learning, the performance of the resulting function should be measured on a test set that is separate from the training set.
- Supervised learning is divided into two types : Classification and Regression.

1. Classification :

- Classification predicts categorical labels (classes), prediction models continuous - valued functions. Classification is considered to be supervised learning.
- Classifies data based on the training set and the values in a classifying attribute and uses it in classifying new data. Prediction means models continuous-valued functions, i.e., predicts unknown or missing values.
- Preprocessing of the data in preparation for classification and prediction can involve data cleaning to reduce noise or handle missing values, relevance analysis to remove irrelevant or redundant attributes and data transformation, such as generalizing the data to higher level concepts or normalizing data.
- Numeric prediction is the task of predicting continuos values for given input. For example, we may wish to predict the salary of college employee with 15 years of work experience or the potential sales of a new product given its price.
- Some of the classification methods like back - propagation, support vector machines and k-nearest-neighbor classifiers can be used for prediction.

2. Regression :

- For an input x, if the output is continuous, this is called a regression problem. For example, based on historical information of demand for tooth paste in supermarket, user are asked to predict the demand for the next month.

- Regression is concerned with the prediction of continuous quantities. Linear regression is the oldest and most widely used predictive model in the field of machine learning. The goal is to minimize the sum of the squared errors to fit a straight line to a set of data points.
- Regression algorithm used in supervised learning is linear regression, Bayesian linear regression, polynomial regression, regression tree etc.

1.3.1 Advantages and Disadvantages of Supervised Learning

1. Advantages of supervised learning

- It performs classification and regression tasks.
- It allows estimating or mapping the result to a new sample.
- We have complete control over choosing the number of classes we want in the training data.

2. Disadvantages of supervised learning

- Supervised learning cannot handle all complex tasks in Machine Learning.
- Computation time is vast for supervised learning.
- It requires a labelled data set.
- It requires a training process.

1.4 Unsupervised Learning

- Unsupervised learning is a type of machine learning in which models are trained using unlabeled dataset and are allowed to act on that data without any supervision.
- In unsupervised learning, a dataset is provided without labels and a model learns useful properties of the structure of the dataset. The main goal of unsupervised learning is to discover hidden and interesting patterns in unlabeled data.
- They are called unsupervised because they do not need a teacher or super-visor to label a set of training examples. Only the original data is required to start the analysis.
- Unsupervised learning tasks typically involve grouping similar examples together, dimensionality reduction and density estimation.
- Common algorithms used in unsupervised learning include clustering, anomaly detection, neural networks.

Scanned with OKEN Scanner

- Fig. 1.4.1 shows unsupervised learning.

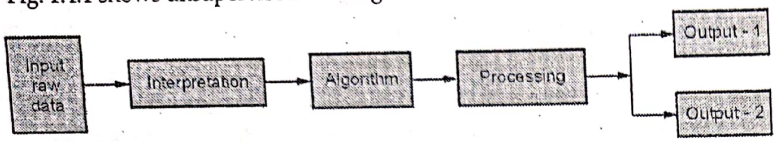


Fig. 1.4.1 Unsupervised learning

- The most common unsupervised learning method is cluster analysis, which applies clustering methods to explore data and find hidden patterns or groupings in data. Unsupervised learning is typically applied before supervised learning, to identify features in exploratory data analysis and establish classes based on groupings.
- Unsupervised machine learning is mainly used to :
  1. Cluster datasets on similarities between features or segment data.
  2. Understand relationship between different data point such as automated music recommendations.
  3. Perform initial data analysis.
- Unsupervised learning algorithms have the capability of analyzing large amounts of data and identifying unusual points among the dataset. Once those anomalies have been detected, they can be brought to the awareness of the user, who can then decide to act or not on this warning.
- Anomaly detection can be very useful in the financial and banking sectors. Indeed, financial fraud has become a daily problem, due to the ease with which credit card details can be stolen. Using unsupervised learning models, unauthorized or fraudulent transactions on a bank account can be identified as it will most often constitute a change in the user's normal pattern of spending.
- Example : Using customer data and user want to create segments of customers who like similar products. The data that user are providing is not labeled and the labels in the outcome are generated based on the similarities that were discovered between data points.
- Types of unsupervised learning is clustering and association analysis.
- There is a wide range of algorithms that can be deployed under unsupervised learning. A few of them includes : K-means clustering, Principal component analysis, Hierarchical clustering and Dendrogram.

1.4.1 Advantages and Disadvantages of Unsupervised Learning

1. Advantages of unsupervised learning

- It does not require a training data to be labelled.
- Dimensionality reduction can be easily accomplished using unsupervised learning.
- Capable of finding previously unknown patterns in data.

2. Disadvantages of unsupervised learning

- Difficult to measure accuracy or effectiveness due to lack of predefined answers during training.
- The results often have lesser accuracy.
- The user needs to spend time interpreting and label the classes which follow that classification.

1.4.2 Difference between Supervised and Unsupervised Learning

Sr. No.	Supervised learning	Unsupervised learning
1.	Desired output is given.	Desired output is not given.
2.	It is not possible to learn larger and more complex models than with supervised learning.	It is possible to learn larger and more complex models with unsupervised learning.
3.	Use training data to infer model.	No training data is used.
4.	Every input pattern that is used to train the network is associated with an output pattern.	The target output is not presented to the network.
5.	Trying to predict a function from labeled data.	Try to detect interesting relations in data.
6.	Supervised learning requires that the target variable is well defined and that a sufficient number of its values are given.	For unsupervised learning typically either the target variable is unknown or has only been recorded for too small a number of cases.
7.	Example : Optical character recognition.	Example : Find a face in an image.

Scanned with OKEN Scanner

8.	We can test our model.	We can not test our model.
9.	Supervised learning is also called classification.	Unsupervised learning is also called clustering.

1.5 Bias Variance Tradeoff

- In the experimental practice we observe an important phenomenon called the bias variance dilemma.
- In supervised learning, the class value assigned by the learning model built based on the training data may differ from the actual class value. This error in learning can be of two types, errors due to 'bias' and error due to 'variance'.
- Fig. 1.5.1 shows bias-variance trade off.

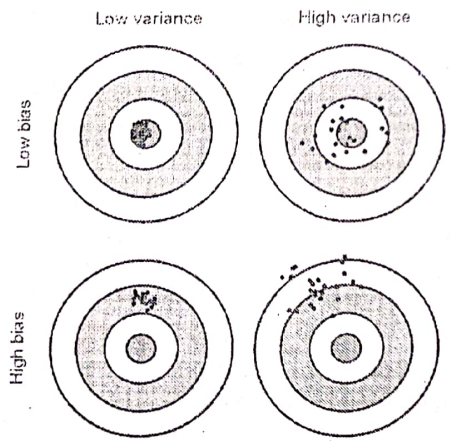


Fig. 1.5.1 Bias-variance trade off

- Give two classes of hypothesis (e.g. linear models and k-NNs) to fit to some training data set, we observe that the more flexible hypothesis class has a low bias term but a higher variance term. If we have parametric family of hypothesis, then we can increases the flexibility of the hypothesis but we still observe the increase of variance.
- The bias-variance-dilemma is the problem of simultaneously minimizing two sources of error that prevent supervised learning algorithm from generalizing beyond their training set :

1. The bias is error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs.
2. The variance is error from sensitivity to small fluctuations in the training set. High variance can cause overfitting : modeling the random noise in the training data, rather than the intended outputs.

- In order to reduce the model error, the designer can aim at reducing either the bias or the variance, as the noise components is irreducible.
- As the model increases in complexity, its bias is likely to diminish. However, as the number of training examples is kept fixed, the parametric identification of the model may strongly vary from one DN to another. This will increase the variance term.
- At one stage, the decrease in bias will be inferior to the increase in variance, warning that the model should not be too complex. Conversely, to decrease the variance term, the designer has to simplify its model so that it is less sensitive to a specific training set. This simplification will lead to a higher bias.

Example 1.5.1

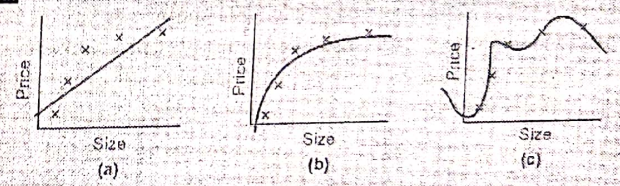
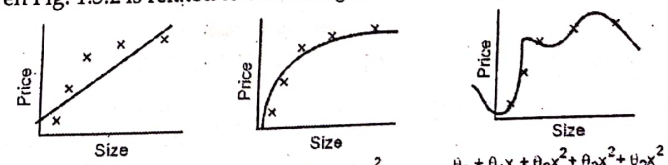


Fig. 1.5.2

Explain the above Fig. 1.5.2 (a), (b) and (c).

Solution :

- Given Fig. 1.5.2 is related to overfitting and underfitting.



$\theta_0 + \theta_1 x$
High bias (underfit)

$\theta_0 + \theta_1 x + \theta_2 x^2$
High bias (underfit)

$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_2 x^2 + \theta_2 x^2$
High variance (overfit)

Fig. 1.5.3

Scanned with OKEN Scanner

Underfitting (High bias and low variance) :

- A statistical model or a machine learning algorithm is said to have underfitting when it cannot capture the underlying trend of the data.
- It usually happens when we have less data to build an accurate model and also when we try to build a linear model with a non-linear data.
- In such cases the rules of the machine learning model are too easy and flexible to be applied on such minimal data and therefore the model will probably make a lot of wrong predictions.
- Underfitting can be avoided by using more data and also reducing the features by feature selection.

Overfitting (High variance and low bias) :

- A statistical model is said to be overfitted, when we train it with a lot of data.
- When a model gets trained with so much of data, it starts learning from the noise and inaccurate data entries in our data set.
- Then the model does not categorize the data correctly because of too many details and noise.
- The causes of overfitting are the non-parametric and non-linear methods because these types of machine learning algorithms have more freedom in building the model based on the dataset and therefore they can really build unrealistic models.
- A solution to avoid overfitting is using a linear algorithm if we have linear data or using the parameters like the maximal depth if we are using decision trees.

1.6 Generalization : Overfitting vs Underfitting

- In addition to using models for prediction, the ability to interpret what a model has learned is receiving an increasing amount of attention.
- Interpretability has to do with how accurate a machine learning model can associate a cause to an effect.
- If a model can take the inputs, and routinely get the same outputs, the model is interpretable :
  1. If you overeat your magi at dinnertime and you always have troubles sleeping, the situation is interpretable.
  2. If all 2019 polls showed " ABC party" win and the "XYZ party" candidate took office, all those models showed low interpretability.
- Interpretability poses no issue in low-risk scenarios. If a model is recommending movies to watch, that can be a low-risk task

- Fitness of a target function approximated by a learning algorithm determines how correctly it is able to classify a set of data it has never seen.

1.6.1 Underfitting and Overfitting

- Training error can be reduced by making the hypothesis more sensitive to training data, but this may lead to overfitting and poor generalization.
- Overfitting occurs when a statistical model describes random error or noise instead of the underlying relationship. Overfitting is when a classifier fits the training data too tightly. Such a classifier works well on the training data but not on independent test data. It is a general problem that plagues all machine learning methods.
- Underfitting : If we put too few variables in the model, leaving out variables that could help explain the response, we are underfitting. Consequences :
  1. Fitted model is not good for prediction of new data - prediction is biased
  2. Regression coefficients are biased
  3. Estimate of error variance is too large
- Because of overfitting, low error on training data and high error on test data. Overfitting occurs when a model begins to memorize training data rather than learning to generalize from trend.
- The more difficult a criterion is to predict, the more noise exists in past information that need to be ignored. The problem is determining which part to ignore.
- Overfitting generally occurs when a model is excessively complex, such as having too many parameters relative to the number of observations. We can determine whether a predictive model is underfitting or overfitting the training data by looking at the prediction error on the training data and the evaluation data. Fig. 1.6.1 shows underfitting and overfitting.

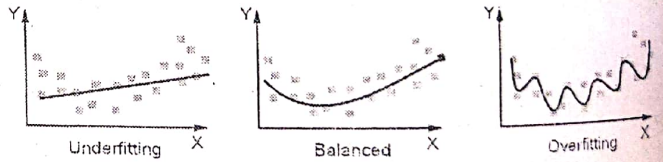


Fig. 1.6.1

- Reasons for overfitting
  1. Noisy data
  2. Training set is too small
  3. Large number of features

Scanned with OKEN Scanner

- In the machine learning the more complex model is said to show signs of overfitting, while the simpler model underfitting. Often several heuristic are developed in order to avoid overfitting, for example, when designing neural networks one may :
  1. Limit the number of hidden nodes
  2. Stop training early to avoid a perfect explanation of the training set, and
  3. Apply weight decay to limit the size of the weights, and thus of the function class implemented by the network

1.7 History of Deep Learning

- The history of Deep Learning can be traced back to 1943, when Walter Pitts and Warren McCulloch created a computer model based on the neural networks of the human brain. They used a combination of algorithms and mathematics they called "threshold logic" to mimic the thought process
- There basic aim was to mimic thought process of human brain; they used algorithms and mathematics to make the threshold logic to mimic human thought process. Alan Turing called the father of AI concluded in 1951 that the machines would not take much time in started thinking of their own; at some point of time, they would be able to talk to each other and it is also expected that they would take the control of the universe.
- Warren McCulloch and Walter Pitts used a combination of mathematics and algorithms they called threshold logic to mimic the thought process. Since then, deep learning has evolved steadily, over the years with two significant breaks in its development.
- The development of the basics of a continuous Back Propagation Model is credited to Henry J. Kelley in 1960. Stuart Dreyfus came up with a simpler version based only on the chain rule in 1962. The concept of back propagation existed in the early 1960s but only became useful until 1985.
- The next significant evolutionary step for deep learning took place in 1999, when computers started becoming faster at processing data and graphics processing units were developed. Neural networks also have the advantage of continuing to improve as more training data is added.
- Around the year 2000, The Vanishing Gradient Problem appeared. It was discovered "features" formed in lower layers were not being learned by the upper layers, because no learning signal reached these layers.

- In 2001, a research report by META Group described the challenges and opportunities of data growth as three-dimensional. The report described the increasing volume of data and the increasing speed of data as increasing the range of data sources and types. This was a call to prepare for the onslaught of Big Data, which was just starting.
- In 2009, Fei-Fei Li, an AI professor at Stanford launched ImageNet, assembled a free database of more than 14 million labeled images. The Internet is and was, full of unlabeled images. Labeled images were needed to "train" neural nets.
- By 2011, the speed of GPUs had increased significantly, making it possible to train convolutional neural networks "without" the layer-by-layer pre-training. With the increased computing speed, it became obvious deep learning had significant advantages in terms of efficiency and speed.
- Generative Adversarial Network (GAN) is a class of machine learning system invented by Ian Goodfellow and his colleagues in 2014. Coming up in history in 2016 Google DeepMind challenge match between Alpha Go versus Lee Sedol, the AlphaGo win all the matches from a world champion Lee Sedol.
- AlfaGo and AlfaZero are computer programs developed by artificial intelligence research company called DeepMind in (2016 - 2017); it plays the board game Go.
- The transformer introduced in 2017 - 19 a deep learning model used specially used for Natural language Processing (NLP).
- Although there is a lot of community contributed to the deep learning but Yann LeCun, Geoffrey Hinton, and Yoshua Bengio have received Turing awards in 2018.

1.8 Learning representations from Data

- A machine learning model can't directly see, hear, or sense input examples. Instead, we must create a representation of the data to provide the model with a useful vantage point into the data's key qualities. That is, in order to train a model, we must choose the set of features that best represent the data.
- Representation learning is a class of machine learning approaches that allow a system to discover the representations required for feature detection or classification from raw data. The requirement for manual feature engineering is reduced by allowing a machine to learn the features and apply them to a given activity.

Scanned with OKEN Scanner

- In representation learning, data is sent into the machine and it learns the representation on its own. It is a way of determining a data representation of the features, the distance function, and the similarity function that determines how the predictive model will perform.
- Representation learning works by reducing high-dimensional data to low-dimensional data, making it easier to discover patterns and anomalies while also providing a better understanding of the data's overall behavior.
- Representation learning is concerned with training machine learning algorithms to learn useful representations.
- Deep neural networks can be considered representation learning models that typically encode information which is projected into a different subspace. These representations are then usually passed on to a linear classifier to, for instance, train a classifier.
- Representation learning can be divided into :
  a) Supervised representation learning : learning representations on task A using annotated data and used to solve task B
  b) Unsupervised representation learning : learning representations on a task in an unsupervised way. These are then used to address downstream tasks and reducing the need for annotated data when learning news tasks.

1.8.1 Greedy Layer - Wise Unsupervised Pretraining

- Greedy algorithms break a problem into many components, then solve for the optimal version of each component in isolation. Unfortunately, combining the individually optimal components is not guaranteed to yield an optimal complete solution.
- Greedy Layer-Wise Unsupervised Pretraining relies on single-layer representation learning algorithm. Each layer is pretrained using unsupervised learning, taking the output of previous layer and producing as output a new representation of the data, whose distribution is hopefully simpler.

Why it is called Greedy layer-wise pretraining ?

- Greedy because, it is a greedy algorithm that optimizes each piece of the solution independently
- Layer-wise because, independent pieces are the layers of the network and training proceeds one layer at a time.
- Pretraining because, it is only a first step before applying a joint training algorithm is applied to fine-tune all layers together.

- Unsupervised feature learning algorithm L, which takes a training set of examples and returns an encoder or feature function f. The X is raw input data.

$f \leftarrow$ Identity function

$\bar{X} = X$

for k = 1, ..., m do

$f^{(k)} = \mathcal{L}(\bar{X})$

$f \leftarrow f^{(k)} \circ f$

$\bar{X} \leftarrow f^{(k)}(\bar{X})$

end for

if fine-tuning then

$f \leftarrow T(f.X.Y)$

end if

Return f

- Gready : Optimize each piece of the solution independently, on piece at a time.
- Layer-Wise : The independent pieces are the layer of the network. Training proceeds once layer at a time, training the kth layer while keeping the previous ones fixed.
- Unsupervised : each layer is trained with an unsupervised representation learning algorithm

When and why does unsupervised pretraining ideas work ?

1. **Idea1 :** Choice of initial parameters for a deep NN can have a significant regularizing effect on the model.

- It remains possible that pretraining initializes the model in a location that would otherwise be inaccessible. For example : a region surrounded by areas where the cost function varies so much from one example to another that mini-batches give only a very noisy estimate of the gradient a region surrounded by areas where the Hessian matrix is so pooly conditioned that gradient descent methods must use very small steps.
- We cannot characterize the exactly what aspects of the pretrained parameters are retained during the supervised training stages.

Scanned with OKEN Scanner

2. **Idea2 :** Learning about the input distribution can help with learning about the mapping from input to output.

- Some features that are useful for the unsupervised task may also be useful for supervised task. This is not yet understood at a mathematical, theoretical level. Many aspects of this approach are highly dependent on the specific models used.
- For example, if we wish to add a linear classifier on top of pretrained features, the features mush make the underlying classes linearly separable. This is another reason that simultaneous supervised and unsupervised learning can be preferable.

Disadvantages of Unsupervised Pretraining :

1. Unsupervsied pretraining does not offer a clear way to adjust the strength of the regularization arising from the unsupervised stage. When we perform unsupervised and supervised learning simultaneously, instead of using the pretraining strategy, there is a single hyperparameter, usually a coefficient attatched to the unsupervised cost, that determine hows strongly supervsied objective will regularize the supervsied model.
2. Two separate training phases has its own hyperparameters. The performance of the second phase cannot be predicted during the first phase, so there is a long delay between proposing hyperparameters for the first phase and being able to update them using feedback from the second phase. Most principled approach: use validation set error in the supervised phase to select hyperparameters of the pretraining phase

1.8.2 Transfer Learning and Domain Adaption

- Transfer learning and domain adaptation refer to situation where what has been learned in one setting is exploited to improve generalization in another settings.
- Transfer learning, multitask learning and domain adaptation can be achieved via representation learning when there exist features that are useful for different settings or tasks, corresponding to underlying factors that appear in more than one setting.
- Two extreme form of transfer learning :

1. **One-shot learning :** Only one example of transfer task is given for one-shot learning. It is possible because the representation learns cleanly separate the underlying classes during first stage. During the transfer learning stage, only one labeled example is needed to infer the label of many possible test examples that all cluster around the same point in representation space.

2. **Zero-shot learning :** No labeled examples are given at all. It is only possible because additional information has been exploited during training.

1.9 Understanding How Deep Learning Works in Three Figures

- In machine learning we take some data, train a model on that data, and use the trained model to make predictions on new data. The process of training a model can be seen as a learning process where the model is exposed to new, unfamiliar data step by step.
- At each step, the model makes predictions and gets feedback about how accurate its generated predictions were. This feedback, which is provided in terms of an error according to some measure is used to correct the errors made in prediction.
- Deep learning algorithms attempt to draw similar conclusions as humans would by continually analyzing data with a given logical structure. To achieve this, deep learning uses a multi-layered structure of algorithms called neural networks.
- Neural networks enable us to perform many tasks, such as clustering, classification or regression. With neural networks, we can group or sort unlabeled data according to similarities among samples in the data.
- Deep Learning is a machine learning method. It allows us to train an AI to predict outputs, given a set of inputs. Both supervised and unsupervised learning can be used to train the AI.
- Let us consider airplane ticket price estimation service. We want our airplane ticket price estimator to predict the price using the following inputs :
  1. Origin Airport
  2. Destination Airport
  3. Departure Date
  4. Airline
- The neurons are grouped into three different types of layers : Input Layer, Hidden Layer(s) and Output Layer.
- The input layer receives input data. Here we have four neurons in the input layer: Origin city, Destination city, Departure Date, and Travel Company. The input layer passes the inputs to the first hidden layer.
- The hidden layers perform mathematical computations on inputs. One of the challenges in creating neural networks is deciding the number of hidden layers, as well as the number of neurons for each layer.

Scanned with OKEN Scanner

- The "Deep" in Deep Learning refers to having more than one hidden layer. The output layer returns the output data. In this case, it gives us the price prediction. Fig. 1.9.1 shows deep neural network.

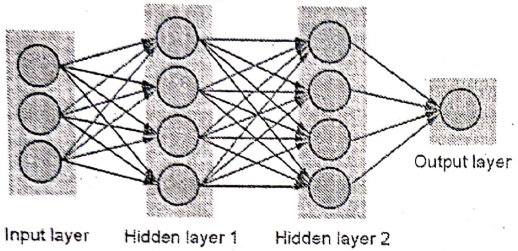


Fig. 1.9.1 Deep neural network

- Fig. 1.9.2 shows ticket reservation using deep neural network.

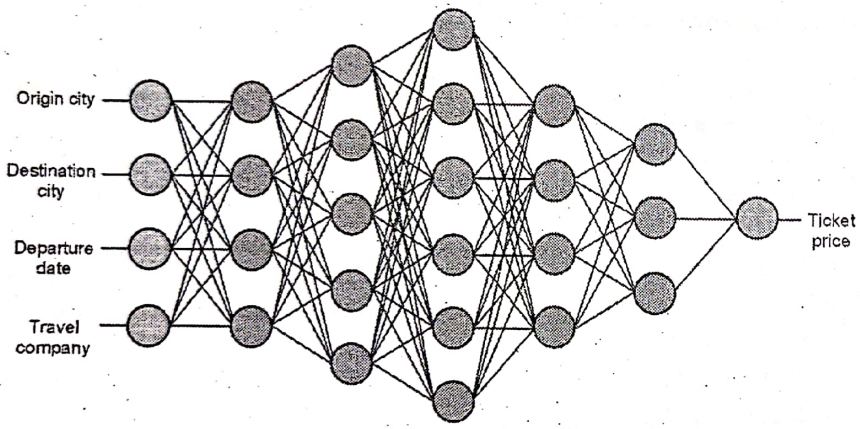


Fig. 1.9.2 Ticket reservation using deep neural network

- Each connection between neurons is associated with a weight. This weight dictates the importance of the input value. The initial weights are set randomly. When predicting the price of an travel ticket, the departure date is one of the heavier factors. Hence, the departure date neuron connections will have a big weight.
- Each neuron has an Activation Function. One of its purposes is to "standardize" the output from the neuron. Once a set of input data has passed through all the layers of the neural network, it returns the output data through the output layer.
- Training phase in deep network is complex part because it requires large data set and a large amount of computational power.

- In this example, we need to find historical data of ticket prices. And due to the large amount of possible city and departure date combinations, we need a very large list of ticket prices. To train the AI, we need to give it the inputs from our data set and compare its outputs with the outputs from the data set. Since the AI is still untrained, its outputs will be wrong.
- Once we go through the whole data set, we can create a function that shows us how wrong the AI's outputs were from the real outputs. This function is called the Cost Function. Ideally, cost function to be zero. Gradient Descent technique used for reducing cost. Fig. 1.9.3 shows weight with cost function.

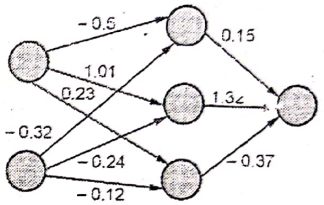


Fig. 1.9.3 Weight with cost function

- The weights of the network are assigned random values at the start. Therefore the network only implements a series of random transformations. Its output is away from what it should ideally be. The loss score is accordingly very high. The loss score decreases as the weights are adjusted a little in the correct direction. This training loop repeated a sufficient number of times.. This yields weight values that minimize the loss function.
- It works by changing the weights in small increments after each data set iteration. By computing the derivative (or gradient) of the cost function at a certain set of weight. Updating the weights using gradient descent is done automatically. This is working of deep network.

1.10 Common Architectural Principles of Deep Network

- The purposes of understanding deep networks are as follows :

  a) Parameters b) Layers c) Activation functions

  d) Loss functions e) Optimization methods f) Hyperparameters

- Parameters : Parameters in neural networks relate directly to the weights on the connections in the network. Gradient descent method is used to find good values for the parameter vector to minimize loss across training dataset.

Scanned with OKEN Scanner

- **Layers :** Layers are a fundamental architectural unit in deep networks. Sometime we need to customize a layer by changing the type of activation function.
- **Activation Functions :** To drive feature extraction, activation functions are used in specific architectures. The higher-order features learned from the data in deep networks are a nonlinear transform applied to the output of the previous layer. This allows the network to learn patterns in the data within a constrained space.
- **Loss Functions :** Loss functions quantify the agreement between the predicted output and the actual output. Various types of loss functions are squared loss, logistic loss, hinge loss and negative log likelihood.
- **Optimization Algorithms :** Training a model in machine learning involves finding the best set of values for the parameter vector of the model. Optimization algorithms are of two types : First-order and Second-order.
- First-order optimization algorithms calculate the Jacobian matrix. Second-order algorithms calculate the derivative of the Jacobian by approximating the Hessian. Second-order methods take into account interdependencies between parameters when choosing how much to modify each parameter.
- **Hyper - parameters :** Hyperparameters are parameters whose values control the learning process and determine the values of model parameters that a learning algorithm ends up learning. Various categories of hyper-parameters are as follows :

  a) Layer size

  b) Magnitude (momentum, learning rate)

  c) Regularization (dropout, drop connect, L1, L2)

  d) Activations (and activation function families)

  e) Weight initialization strategy

  f) Loss functions

  g) Settings for epochs during training (mini-batch size)

  h) Normalization scheme for input data (vectorization)

1.10.1 Optimization Algorithms

- Optimization issue in machine learning algorithms is about finding the correct set of inputs for a function that gives the maximum effectiveness in the function evaluation.

- Be it fitting in the logical regression models in machine learning or training the neural networks with varying datasets, the problem of optimization arises in all phases of a machine learning model.
- Out of hundreds of optimization algorithms available, it becomes difficult to choose a single algorithm that can give the best performance while applied in our machine learning model. One suggestable approach to execute these optimization algorithms and make the maximum use of it is to group these algorithms and execute them on the machine learning.
- Most of the times, the "continuous function optimization" problem arises in majority of the machine learning algorithms, in which most of the input given are the real numbers, and so are the outputs. These kinds of problems that take only discrete values as input are generally referred to as "Combinatorial Optimization Problems".
- The optimization algorithms state that if more information about the target function can be made available, it becomes easier to optimize that function and that information can also be utilized effectively for further data processing.
- The major point that comes across the execution of an optimization algorithm is to decide whether we can differentiate an objective function at a given point or not.
- That is, can we calculate the first derivative of a function for a given solution or not ?
- Based on this point, the optimization algorithms are further classified into two categories : One that differentiates the function and other that does not.
- Hence, in this section, we shall discuss the "differentiable" and "non differentiable" objective functions that can be used to group multiple optimization algorithms.

1.10.2 Differentiable Objective Function

- If we can calculate the derivative of a function at any given point while input is given to the system such function can be referred to as "Differential Function".
- One can define the derivative of a function as the rate at which the function changes its value at a given point of time. This is often referred to as a slope too.
- One can apply the optimization techniques on these derivative functions using simple calculus.
- Optimization techniques can sound easier if the derivatives of these "continuous functions", as mentioned above, can be calculated. Some of the algorithms that uses these gradient values of the derivatives are as follows :

Scanned with OKEN Scanner

1. **Bracketing algorithms :**
   - This technique is helpful when there are problems having only one input variable and the optima exist in the pre - defined specific criteria or range.
   - These algorithms can easily navigate this range that is known already and locate the optima. The only drawback is that the algorithm assumes that there is only one optima present in the model.
   - The advantage of using this algorithm is that it can be utilized in a model even if sometimes there is no derivative information about the variables available.
   - Some of the examples that use bracketing algorithms are Fibonacci search, Golden Section search, Bisection method, etc.
2. **Local descent algorithms :**
   - These algorithms work for the models in which there are multiple input variables with one global optima.
   - The algorithm is widely used in the line search problem.
   - This problem includes the definition of the direction to move during a search space and then it performs the bracketing type search in a line in the direction chosen.
   - The algorithm executes until no other iteration of finding the improved directions is possible.
   - These iterations make the algorithm costly as it continues its execution till an effective direction is obtained.
3. **First order algorithms :**
   - These algorithms use the first order derivatives (gradient) to decide the direction to move in the search space.
   - This algorithm works by first calculating the first derivative of the function, and then following it in the opposite direction, for example going downhill to minimum value for minimization problems, with the help of step size, also known as "learning rate".
   - This step size, or learning rate, is a hyperparameter in the algorithm, that decides the distance to cover, or how far to cover in a search space, which is opposite to the normally used local descent algorithms, which do not have this hyperparameter and performs a full line search in every directions specified.

   - These algorithms are also known as "Gradient Descent" algorithms and following the introduction to some of the minor extensions, these are also known as Momentum, Adagrad, RMSProp, Adam, etc.
   - These gradient descent algorithms are also helpful in training the artificial neural networks and implementing deep learning models in it, by providing the template for Stochastic Gradient Descent, useful for artificial neural networks.
   - Here, the gradient will be based on assumption, instead of direct calculation, using the prediction techniques on the trained data.
4. **Second order algorithms :**
   - These algorithms use the second order derivative of the input variables for choosing the direction of movement in the search space.
   - The algorithms work appropriately only for the objective functions in which the Hessian matrix needs to be calculated.
   - Some of the examples in which the second order algorithms are used.
     - Newton's method
     - Secant method
   - These algorithms are also known as Quasi Newton Methods.

1.10.3 Non - Differentiable Objective Function

- Although, optimization algorithms working on the derivatives of the objective functions are efficient and fast, there are certain objective functions whose derivatives cannot be calculated, the reason being the complexity of the function.
- Some of the reasons for the complexities in the function include.
  - Lack of analysis of the function
  - Multiple optima required.
  - Evaluation of stochastic functions
  - Objective functions are discontinuous.
- The optimization algorithms that do not make the compulsion for the first or second order derivatives for their objective functions are called as Black - box optimization algorithms. Some of these algorithms are :
  - Direct algorithms
  - Stochastic algorithms
  - Population algorithms

Scanned with OKEN Scanner

Let us have a brief of each of these :

1. **Direct algorithms :**
   - These algorithms are used when the calculation of the derivatives of the objective function is not possible.
   - The algorithms work with an assumption that the objective function contains single optima.
   - These methods are also referred to as "pattern search" algorithms, since they analyze the search space using the geometrical shapes and patterns.
   - The gradient information required to run the algorithm is calculated directly from the objective function by computing the difference between the scores obtained from the points in the search space.
   - This information estimated are then helpful in choosing a direction to commute in the search space and cover the region of the present optima.
   - Some of the examples that use these direct algorithms are Cyclic Coordinate search, Powell's method, Hooke-Jeeves method, etc.
2. **Stochastic algorithms :**
   - For the variables whose derivatives cannot be calculated, stochastic optimization algorithms use the randomness for those objective functions to commute in the search space.
   - Hence, due to the randomness involved, the stochastic algorithms involve many data sampling for the objective function.
3. **Population algorithms :**
   - These algorithms maintain a pool of solutions for a given input, often known as a population of candidate solutions, which are used to explore, sample the optima.
   - These algorithms are mostly used in the problems that are more challenging and also involves the evaluation of functions containing considerable noise in it, in addition to the presence of multiple global optima. The solutions of such algorithms are difficult to be found by other methods.
   - Genetic algorithms, differential evolution, particle swarm optimization, etc. are the examples of population algorithms.

1.11 Architecture Design

- Architecture refers to the overall structure of the network.
- Most neural networks are organized into groups of units called layers. These layers

are arranged in a chain structure, with each layer being a function of the layer that preceded it.

- In this structure, the first layer is given by

$$h^{(1)} = g^{(1)}(W^{(1)T}x + b^{(1)})$$

The second layer is given by

$$h^{(2)} = g^{(2)}(W^{(2)T}x + b^{(2)})$$

1.12 Applications of Deep Learning

1. **Aerospace and defense :** Deep learning is utilized extensively to help satellites identify specific objects or areas of interest and classify them as safe or unsafe for soldiers.
2. **Financial services :** Financial institutions regularly use predictive analytics to drive algorithmic trading of stocks, assess business risks for loan approvals, detect fraud and help manage credit and investment portfolios for clients.
3. **Medical research :** The medical research field uses deep learning extensively. For example, in ongoing cancer research, deep learning is used to detect the presence of cancer cells automatically.
4. **Industrial automation :** The heavy machinery sector is one that requires a large number of safety measures. Deep learning helps with the improvement of worker safety in such environments by detecting any person or objects that comes within the unsafe radius of a heavy machine.
5. **Facial recognition :** This feature utilizing deep learning is being used not just for a range of security purposes but will soon enable purchases at stores. Facial recognition is already being extensively used in airports to enable seamless, paperless check-ins.

1.12.1 Speech Recognition Problem

- Speech Recognition (is also known as Automatic Speech Recognition (ASR), or computer speech recognition) is the process of converting a speech signal to a sequence of words, by means of an algorithm implemented as a computer program.
- Automatic Speech Recognition (ASR) is the task of producing a text transcription of the audio signal from a speaker.
- Large Vocabulary Speech Recognition (LVSR) involves a large or open vocabulary and we will also take it to imply a speaker independent recognizer.

Scanned with OKEN Scanner

- The most general LVSR systems may also need to operate on spontaneous as well as read speech with audio data from an uncontrolled recording environment corrupted by both unstructured and structured noise
- Traditional LVSR recognizers are based on Hidden Markov Models (HMMs) with Gaussian Mixture Model (GMM) emission distributions, n-gram language models, and use beam search for decoding.
- GMM-HMM acoustic models are trained with the Expectation-Maximization (EM) algorithm using a procedure that trains progressively more sophisticated models using an initial alignment from the previous model.
- In speech recognition, the main goal of the feature extraction step is to compute a parsimonious sequence of feature vectors providing a compact representation of the given input signal.
- The feature extraction is usually performed in three stages.
  1. The first stage is called the speech analysis or the acoustic front end. It performs some kind of spectro temporal analysis of the signal and generates raw features describing the envelope of the power spectrum of short speech intervals.
  2. The second stage compiles an extended feature vector composed of static and dynamic features.
  3. Finally, the last stage transforms these extended feature vectors into more compact and robust vectors that are then supplied to the recognizer.
- In speech recognition a supervised pattern classification system is trained with labeled examples; that is, each input pattern has a class label associated with it.
- Pattern classifiers can also be trained in an unsupervised fashion. For example in a technique known as vector quantization, some representation of the input data is clustered by finding implicit groupings in the data.
- The resulting table of cluster centers is known as a codebook, which can be used to index new vectors by finding the cluster center that is closest to the new vectors.
- Once a feature selection or classification procedure finds a proper representation, a classifier can be designed using a number of possible approaches

1.12.2 Image Classification

- Image classification forms basis of computer vision problems.
  1. In self driving cars image classification can be used to detect traffic lights, trees, vehicles, peoples around etc.

  2. In healthcare it can be used to analyze medical images and depict the symptoms of illness.
  3. With the ubiquitous technologies such as AI and IOT huge amount of data in the form of images, video, speech is generated. Image and video data posted by persons can be used in recommendation system in online shopping or places to visit etc.
- Image classification is where a computer can analyses an image and identify the 'class' the image falls under. For example, input an image of a sheep. Image classification is the process of the computer analyzing the image and telling you it's a sheep.
- Early image classification relied on raw pixel data. This meant that computers would break down images into individual pixels. The problem is that two pictures of the same thing can look very different. They can have different backgrounds, angles, poses, etc. This made it quite the challenge for computers to correctly 'see' and categories images.
- Image classification with deep learning most often involves convolutional neural networks or CNNs. In CNNs, the nodes in the hidden layers don't always share their output with every node in the next layer.
- Deep learning allows machines to identify and extract features from images. This means they can learn the features to look for in images by analyzing lots of pictures.

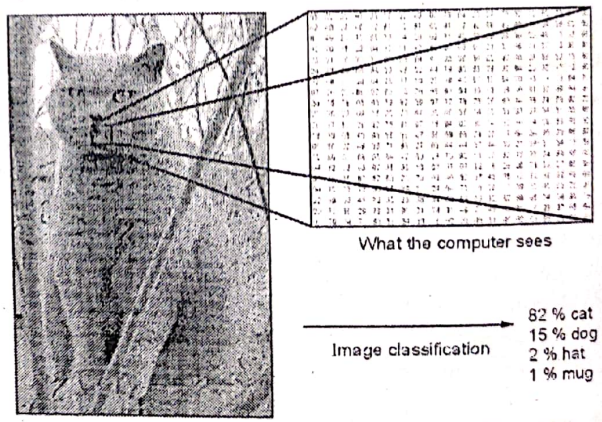


Fig. 1.12.1

- Image is analyzed in the form of pixels. Image is an array of pixels where size of the

Scanned with OKEN Scanner

matrix depends on resolution of an image. Image classification task is done by grouping pixels into specified categories referred to as classes.

- The image is segregated into its most prominent features using the algorithm giving the classifier an idea about the class of the image it may belong. Thus the feature extraction process is most important step in image classification. Also data fed to the algorithm plays important role particularly in supervised image classification technique.

1.13 Introduction and use of Popular Industry Tools

1.13.1 TensorFLow

- TensorFlow is an end-to-end open source platform for Deep learning. It was developed by the Google Brain team and released in November 2015.
- It uses Python or JavaScript to provide a convenient front-end API for building applications, while executing those applications in high-performance C++.
- TensorFlow is popular for its TensorFlow Core programs where it has two main actions.
  1. Building the computational graph in the construction phase
  2. Running the computational graph in the execution phase
- A tensor is a mathematical object and a generalization of scalars, vectors, and matrices. A tensor can be represented as a multidimensional array. Let's understand how TensorFlow works.
  1. Its programs are usually structured into a construction phase and an execution phase.
  2. The construction phase assembles a graph that has nodes (ops/operations) and edges (tensors).
  3. The execution phase uses a session to execute ops (operations) in the graph.
- Features of Tensorflow are as follows :
  1 Faster debugging with Python tools
  2 Dynamic models with Python control flow
  3 Support for custom and higher-order gradients
  4 TensorFlow offers multiple levels of abstraction, which helps to build and train models.
  5 TensorFlow provides the flexibility and control with features like the Keras functional API and model.

  6 Well-documented so easy to understand.
  7 Probably the most popular easy to use with Python.
- TensorFlow programs use a tensor data structure to represent all data, only tensors are passed between operations in the computation graph. TensorFlow allows developers to create dataflow graphs, structures that describe how data moves through a graph or a series of processing nodes.
- Each node in the graph represents a mathematical operation, and each connection or edge between nodes is a multidimensional data array, or tensor.
- TensorFlow applications can be run on most any target that's convenient : a local machine, a cluster in the cloud, iOS and Android devices, CPUs or GPUs. If you use Google's own cloud, you can run TensorFlow on Google's custom TensorFlow Processing Unit (TPU) silicon for further acceleration. The resulting models created by TensorFlow, though, can be deployed on most any device where they will be used to serve predictions.
- Tesorflow supports three main type of data types : Constants, Variables and Placeholders.
- Constant in TensorFlow : tf.constant() function is used to create constant in TensorFlow. Here, it will always create host/CPU tensor. Syntax is as follows:

  tf.constant (value, dtype=None, shape=Name, name='Constant')

  where

  a) **Value :** A constant value or list. This value is the constant value that is passed to the function.

  b) **Dtype :** It is the data type of the element. dtype is passed to the specify the datatype of the value which is passed as constant. If dtype is not mentioned, then it concludes the type from value, by default.

  c) **Shape :** It is optional dimensions given to the value.

  d) **Name :** Another optional attribute which assigns name for the tensor, which contains the constant value.

  e) **Verify_shape :** It is a boolean value that helps enabling of verification of the shape of values

1.13.2 Keras

- Keras is a high-level neural networks API, capable of running on top of Tensorflow, Theano, and CNTK. It enables fast experimentation through a high level,

Scanned with OKEN Scanner

userfriendly, modular and extensible API. Keras can also be run on both CPU and GPU.

- Keras is written in Python and supports multiple back-end neural network computation engines.
- Keras provides seven different datasets, which can be loaded in using Keras directly. These include image datasets as well as a house price and a movie review datasets.
- Salient features of Keras
  1. Keras is a high-level interface and uses Theano or Tensorflow for its backend.
  2. It runs smoothly on both CPU and GPU.
  3. Keras supports almost all the models of a neural network – fully connected, convolutional, pooling, recurrent, embedding, etc. Furthermore, these models can be combined to build more complex models.
  4. Keras, being modular in nature, is incredibly expressive, flexible, and apt for innovative research.
  5. Keras is a completely Python-based framework, which makes it easy to debug and explore.
- The model is the core Keras data structure. There are two main types of models available in Keras: the Sequential model, and the Model class used with the functional API.
- The easiest way of creating a model in Keras is by using the sequential API, which lets we stack one layer after the other. The problem with the sequential API is that it doesn't allow models to have multiple inputs or outputs, which are needed for some problems.
- The functional API allows user to create the same models but offers more flexibility at the cost of simplicity and readability. It can be used with multiple input and output layers and shared layers, which enables to build complex network structures. When using the functional API, we always need to pass the previous layer to the current layer. It also requires the use of an input layer. The functional API is also often used for transfer learning.
- Keras provides seven different datasets, which can be loaded using Keras directly. These include image datasets, a house price, and a movie review dataset. Keras is a high-level neural networks API running on top of Tensorflow.

- Demerits of Keras :
  1. Restricted to use only on smaller datasets.
  2. Slower performance since it is completely written in Python.
  3. The project repository is not as huge compared to TensorFlow.

1.13.3 Difference Between Keras and Tensorflow

Keras	Tensorflow
Keras is a high-level API which is running on top of TensorFlow.	TensorFlow is a framework that offers both high and low-level APIs.
Keras is used for small datasets as it is slower.	TensorFlow is used for high-performance models and massive datasets that require execution fast.
It can be used for low-performance models.	It is use for high-performance models.
Working with beginner-friendly projects with small datasets.	Rendering heavy projects with ease.
Readable and concise architecture	Comparatively complex architecture.
Community support is minimal but growing.	Being one of the first deep learning frameworks that were developed, it has an extensive community and a rich reserve of readable material.

1.13.4 PyTorch

- PyTorch is an open source machine learning library for Python and is completely based on Torch. It is primarily used for applications such as natural language processing. It was developed by Facebook's AI Research lab.
- PyTorch uses dynamic computation, which allows greater flexibility in building complex architectures. Pytorch uses core Python concepts like classes, structures and conditional loops.
- PyTorch has a very simple interface like Python. It provides an easy way to use API.
- It allows developers to train a neural network model in a distributed manner. It provides optimized performance in both research and production with the help of native support for peer to peer communication and asynchronous execution of collective operation from Python and C++.

Scanned with OKEN Scanner

- The PyTorch core is used to implement tensor data structure, CPU and GPU operators, basic parallel primitives and automatic differentiation calculations.
- PyTorch is supported by many major cloud platforms such as AWS. With the help of prebuilt images, large scale training on GPU's and ability to run models in a production scale environment etc.; it provides frictionless development and easy scaling.

1.13.5 Caffe

- Convolutional Architecture for Fast Feature Embedding (Caffe) is a deep learning framework made with expression, speed, and modularity in mind. It is developed by Berkeley AI Research (BAIR) and by community contributors.
- Caffe is primarily a C++ library and exposes a modular development interface. Caffe offers interfaces for daily use by way of the command line, Python and MATLAB.
- Caffe processes data in the form of Blobs which are N-dimensional arrays stored in a C - contiguous fashion. Data is stored both as data we pass along the model and as diff, which is a gradient computed by the network.
- Data layers handle how the data is processed in and out of the Caffe model. Pre-processing and transformation like random cropping, mirroring, scaling and mean subtraction can be done by configuring the data layer. Furthermore, pre-fetching and multiple-input configurations are also possible.

1.13.6 Shogun

- Shogun is an open source machine learning software library built in C++. Shogun was developed in 1999 by Soeren Sonnenburg and Gunnar Raetsch.
- It offers numerous algorithms and data structures for machine learning problems. The focus of Shogun is on kernel machines such as support vector machines for regression and classification problems. Shogun also offers a full implementation of Hidden Markov models.
- Shogun is a community learning platform in every sense, anyone can use the toolbox to learn about ML and apply it to solve problems.
- Shogun is community-based and non-commercial. It is currently released under the GPLv3. It is also moving towards BSD compatibility with optional GPL (General Public License) parts. This means companies can use the code without having to turn their code-base to GPL.

❑❑❑

Notes

Scanned with OKEN Scanner

Unit II

2 Deep Neural Networks

Syllabus

Introduction to Neural Networks : *The Biological Neuron, The Perceptron, Multilayer Feed-Forward Networks,* ***Training Neural Networks :*** *Backpropagation and Forward propagation* ***Activation Functions :*** *Linear, Sigmoid, Tannh, Hard Tanh, Softmax, Rectified Linear,* ***Loss Functions :*** *Loss Function Notation, Loss Functions for Regression, Loss Functions for Classification, Loss Functions for Reconstruction,* ***Hyperparameters :*** *Learning Rate, Regularization, Momentum, Sparsity, Deep Feedforward Networks - Example of Ex OR, Hidden Units, cost functions, error backpropagation, Gradient-Based Learning, Implementing Gradient Descent, vanishing and Exploding gradient descent, Sentiment Analysis, Deep Learning with Pytorch, Jupyter, colab.*

Contents

2.1 Introduction to Neural Networks

2.2 The Biological Neuron

2.3 The Perceptron

2.4 Architecture of Neural Networks

2.5 Training Neural Networks

2.6 Activation Functions

2.7 Loss Functions

2.8 Hyper Parameters

2.9 Regularization

2.10 Deep Feedforward Networks

2.11 Gradient-Based Learning

2.12 Deep Learning with Jupyter

2.13 Deep Learning with Colab

2.1 Introduction to Neural Networks

- Neural networks consists of many numbers of simple elements (neurons) connected between them in system. Whole system is able to solve of complex tasks and to learn for it like a natural brain.
- Neural network is a system composed of many simple processing elements operating in parallel whose function is determined by network structure, connection strengths, and the processing performed at computing elements or nodes.
- For user, NN is black box with input vector (source data) and output vector (result).
  - A Neural network is usually structured into an input layer of neurons, one or more hidden layers one output layer.
  - Neurons belonging to adjacent layers are usually fully connected and the various types and architectures are indentified both by the different topologies adopted for the connections as well as by the choice of the activation function.
  - The values of the functions associated with the connections are called "weights".
  - The whole game of using NNs is in fact that, in order for the network to yield appropriate outputs for given inputs, the weight must be set to suitable values. The way this is obtained allows a further distinction among modes of operations.
  - A neural network is a processing device, either an algorithm or actual hardware, whose design was motivated by the design and functioning of human brains and components thereof.
  - Most neural networks have some sort of "training" rule whereby the weights of connections are adjusted on the basis of presented patterns.
  - In other words, neural networks "learn" from example, just like children learn to recognize dogs from examples of dogs, and exhibit some structural capability for generalization.
  - Neural networks normally have great potential for paralleism, since the computations of the components are independent of each other.
  - Neural networks are a different paradigm for computing :
    1. Von Neumann machines are based on the processing/memory abstraction of human information processing.
    2. Neural networks are based on the paralllel architecture of animal brains.

Scanned with OKEN Scanner

- Neural networks are a form of multiprocessor computer system, with
  a. Simple processing elements
  b. A high degree of interconnections
  c. Simple scalar messages
  d. Adaptive interaction between elements
- The advantages of neural networks are due to its adaptive and generalization ability.
  a) Neural networks are adaptive methods that can learn without any prior assumption of the underlying data.
  b) Neural network, namely the feed forward multilayer perception and radial basis function network have been proven to be universal functional approximations.
  c) Neural networks are non-linear model with good generalization ability.
- **Useful properties and capabilities of neural network.**
  1. **Nonlinearity :** An artificial neuron can be linear or nonlinear. A neural network, made up of an interconnection of nonlinear neurons, is itself nonlinear.
  2. **Adaptivity :** Neural networks have a built-in capability to adapt their synaptic weights to changes in the surrounding environment.
  3. **Contextual information :** Knowledge is represented by the very structured and activation state of a neural network.
  4. **Evidential response :** In the context of pattern classification, a neural network can be designed to provide information not only about which particular pattern to select, but also about the confidence in the decision made.
  5. **Uniformity of analysis and design :** Neural networks enjoy universality as information processors.
  6. **VLSI implement-ability :** The massively parallel nature of a neural network makes it potentially fast for the computation of certain tasks.

2.1.1 Advantages of Neural Network

- The advantages of neural networks are due to its adaptive and generalization ability.
  a) Neural networks are adaptive methods that can learn without any prior assumption of the underlying data.
  b) Neural network, namely the feed forward multilayer perception and radial basis function network have been proven to be universal functional approximations.
  c) Neural networks are non-linear model with good generalization ability.

2.1.2 Application of Neural Network

- Neural network applications can be grouped in following categories :
  1. **Clustering :** A clustering algorithm explores the similarity between patterns and places similar patterns in a cluster. Best known applications include data compression and data mining.
  2. **Classification/Pattern recognition :** The task of pattern recognition is to assign an input pattern (like handwritten symbol) to one of many classes. This category includes algorithmic implementations such as associative memory.
  3. **Function approximation :** The tasks of function approximation is to find an estimate of the unknown function f() subject to noise. Various engineering and scientific disciplines require function approximation.
  4. **Prediction/Dynamical systems :** The task is to forecast some future values of a time-sequenced data. Prediciton has a significant impact on decision support systems. Prediction differs from function approximation by considering time factor.

2.1.3 Difference between Digital Computer and Neural Networks

Sr. No.	Digital Computers	Neural Networks
1.	Deductive reasoning : We apply known rules input data to produce output.	Inductuve reasoning : Given input and output data (training examples), we construct the rules.
2.	Computation is centralized, synchronous and serial.	Coputation is collectivem asynchronous and parallel.
3.	Memory is packetted, literally stored and location addressable.	Memory is distributed, internalized and content addressable.
4.	Not fault toerant. One transistor goes and it no longer works.	Fault tolerant, redundancy and sharing of responsibilities.
5.	Fast. Measured in millionths of a second.	Slow. Measured in thousands of a second.
6.	Exact.	Inexact.
7.	Static connectivity.	Dynamic connectivity.
8.	Applicable if well defined rules with precise input data.	Applicable if rules are unknown or complicated or if data is noisy or partial.

Scanned with OKEN Scanner

2.2 The Biological Neuron

- Artificial neural systems are inspired by biological neural systems. The elementary building block of biological neural systems is the neuron.
- The brain is a collection of about 10 billion interconnected neurons. Each neuron is a cell [right] that uses biochemical reactions to receive, process and transmit information. Fig. 2.2.1 shows biological neural systems.

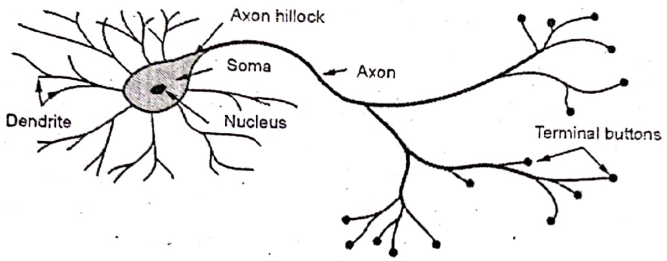


Fig. 2.2.1 Schematic of biological neuron

- The single cell neuron consists of the cell body or soma, the dendrites and the axon. The dendrites receive signals from the axons of other neurons. The small space between the axon of one neuron and the dendrite of another is the synapse. The afferent dendrites conduct impulses toward the soma. The efferent axon conducts impulses away from the soma.

Basic Components of Biological Neurons

1. The majority of neurons encode their activations or outputs as a series of brief electrical pulses (i.e. spikes or action potentials).
2. The neuron's ***cell body (soma)*** processes the incoming activations and converts them into output activations.
3. The neuron's ***nucleus*** contains the genetic material in the form of DNA. This exists in most types of cells, not just neurons.
4. ***Dendrites*** are fibres which emanate from the cell body and provide the receptive zones that receive activation from other neurons.
5. Axons are fibres acting as transmission lines that send activation to other neurons.
6. The junctions that allow signal transmission between the axons and dendrites are called ***synapses***. The process of transmission is by diffusion of chemicals called ***neurotransmitters*** across the synaptic cleft.

- Comparison between Biological NN and Artificial NN

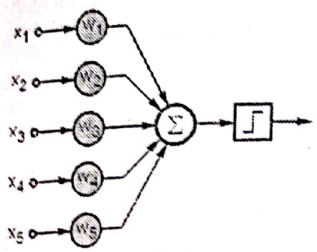
Biological NN	Artificial NN
soma	unit
Axon, dendrite	dendrite
synapse	weight
potential	weighted sum
threshold	bias weight
signal	activation

2.3 The Perceptron

- A learning algorithm is an adaptive method by which a network of computing units self-organizes to implement the desired behaviour.
- Learning algorithms can be divided into supervised and unsupervised methods. Supervised learning denotes a method in which some input vectors are collected and presented to the network. The output computed by the network is observed and the deviation from the expected answer is measured.
- Unsupervised learning is used when, for a given input, the exact numerical output a network should produce is unknown. The perceptron learning algorithm is an example of supervised learning with reinforcement
- Rosenblatt's perceptron is built around a nonlinear neuron, namely, the McCulloch-Pitts model of a neuron.
- Rosenblatt perceptron is a binary single neuron model. The inputs integration is implemented through the addition of the weighted inputs that have fixed weights obtained during the training stage. If the result of this addition is larger than a given threshold ? the neuron fires. When the neuron fires its output is set to 1, otherwise it's set to 0.

Scanned with OKEN Scanner

- The equation can be re-written as follows including what it's known as the bias term :



$$h^{(x)} = \begin{cases} 1 & \text{if } w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_d \cdot x_d \geq \theta \\ 0 & \text{if } w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_d \cdot x_d < \theta \end{cases}$$

The equation can be re-written as follows including what its known as the bias term : $x_0 = 1$, $w_0 = \theta$.

$$h^{(x)} = \begin{cases} 1 & \text{if } w_0 \cdot x_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_d \cdot x_d \geq 0 \\ 0 & \text{if } w_0 \cdot x_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_d \cdot x_d < 0 \end{cases}$$

$$h^{(x)} = \begin{cases} 1 & \text{if } w^t \cdot x \geq 0 \\ 0 & \text{if } w^t \cdot x \geq 0 \end{cases}$$

2.3.1 ADALINE Network

- ADALINE (Adaptive Linear Neuron) is an early single-layer artificial neural network.
- An important generalized of the perceptrons training algorithm was presented by Widrow and Hoff as the least mean square learning procedure also known as the delta rule.
- The learning rule was applied to the "adaptive linear element" also named Adaline.
- The perceptron learning rule uses the output of the thersold function for learning. The delta rule uses the net output without further mapping into output values –1 or + 1.
- Fig. 2.3.1 shows Adaline.

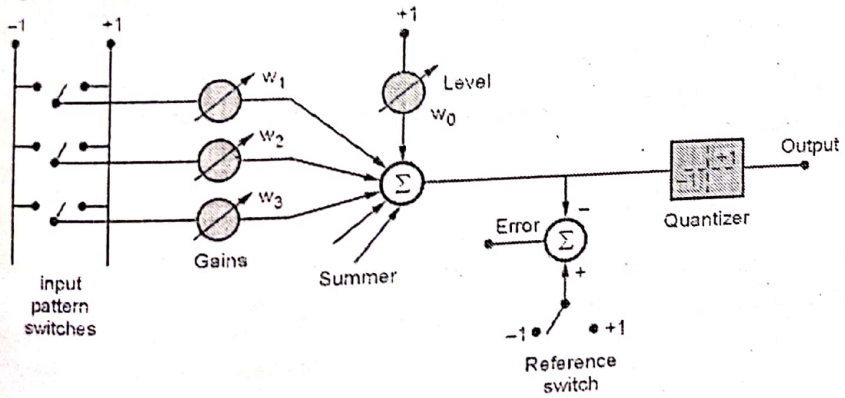


Fig. 2.3.1 Adaline

- If the input conductances are denoted by W_i where m i = 0, 1, 2, ..., n and input and output signals by x_{in} and y respectively, then the output of the central block is defined to be :

$$y = \sum_{i=1}^{x_{in}} w_i x_i + \theta$$

Where, $\theta = W_o$

- In a simple physical implementation, this device consists of a set of controllable resistors connected to a circuit which can sum up currents caused by the input voltage signals. Usually the central block, the summer is also followed by a quantizer which outputs + 1 of – 1, depending on the polarity of the sum.
- The problem is to determine the coefficients , where i = 0, 1 ..., n , in such way that the input output response is correct for a large number of arbitrarily chosen signal sets.
- If an exact mapping is not possible the average error must be minimized, for instance, in the sense of least squares.
- An adaptive operation means that there exists a mechanism by which the can be adjusted, usually iteratively to attain the correct values.
- For the Adaline, Widrow introduced the delta rule to adjust the weights.
- For the pth input-output pattern, the earor measure of a single-output Adaline can be expressed as,

$$E_P = (t_p - o_p)^2$$

Where t_p = Target output

o_p = Actual output of the Adaline E_P

- The derivation of E_P with respect to each weight W_i is

$$\frac{\partial E_P}{\partial W_i} = -2 (t_p - o_p) x_i$$

- To decrease by gradient descent, the update formula for W_i on the pth input-output pattern is

$$\Delta_P W_i = \eta (t_p - O_p) x_i$$

- The delta rule tries to minimize squared errors, it is also referred to as the least mean square learning procedure or Widrow - Hoff learning rule.

Scanned with OKEN Scanner

2.3.2 Sigmoid Neuron

- A sigmoid function produces a curve with an "S" shape. The example sigmoid function shown on the left is a special case of the logistic function, which models the growth of some set.

$$\text{sig}(t) = \frac{1}{1+e^{-t}}$$

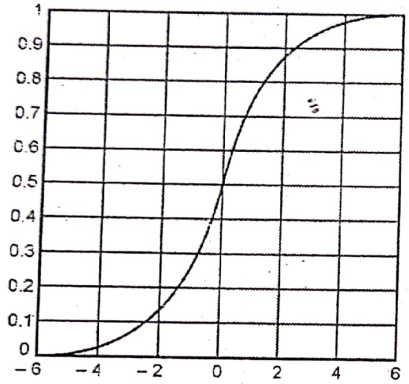


Fig. 2.3.2

- In general, a sigmoid function is real-valued and differentiable, having a non-negative or non-positive first derivative, one local minimum, and one local maximum.
- The logistic sigmoid function is related to the hyperbolic tangent as follows :

$$1 - 2\,\text{sig}(x) = 1 - 2\,\frac{1}{1+e^{-x}} = -\tan h\,\frac{x}{2}$$

- Sigmoid functions are often used in artificial neural networks to introduce nonlinearity in the model.
- A neural network element computes a linear combination of its input signals, and applies a sigmoid function to the result.
- A reason for its popularity in neural networks is because the sigmoid function satisfies a property between the derivative and itself such that it is computationally easy to perform.

$$\frac{d}{dt}\,\text{sig}(t) = \text{sig}(t)\,(1 - \text{sig}(t))$$

- Derivatives of the sigmoid function are usually employed in learning algorithms.

2.4 Architecture of Neural Networks

2.4.1 Single Layer Feed Forward Network

- The architecture of the neural network refers to the arrangement of the connection between neurons, processing element, number of layers, and the flow of signal in the neural network.
- There are mainly two category of neural network architecture :
  a. Feed-forward
  b. Feedback (recurrent) neural networks.

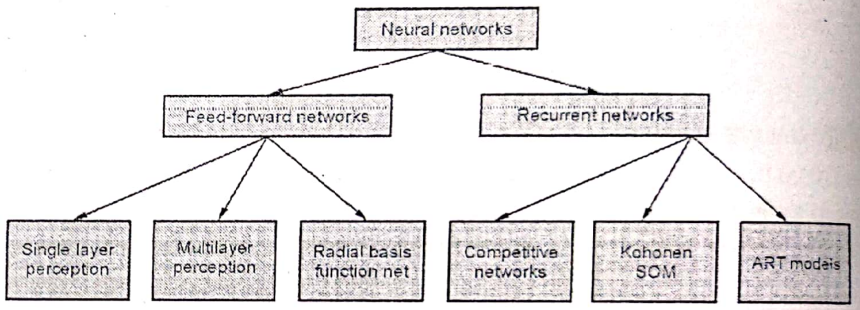


Fig. 2.4.1

1. Architecture and Learning Rule

- In late 1950s, Frank Rosenblatt introduced a network composed of the units that were enhanced version of McCulloch-Pitts Threshold Logic Unit (TLU) model.
- Rosenblatt's model of neuron, a perceptron, was the result of merger between two concepts from the 1940s, McCulloch-Pitts model of an artificial neuron and Hebbian learning rule of adjusting weights.
- In addition to the variable weight values, the perceptron model added an extra input that represents bias. Thus, the modified equation is now as follows :

$$\text{Sum} = \sum_{i=1}^{N} I_i W_i + b,$$

where b represents the bias value.

- Fig. 2.4.2 shows a typical perception setup for pattern recognition applications, in which visual patterns are represented as matrices of elements between 0 and 1.

Scanned with OKEN Scanner

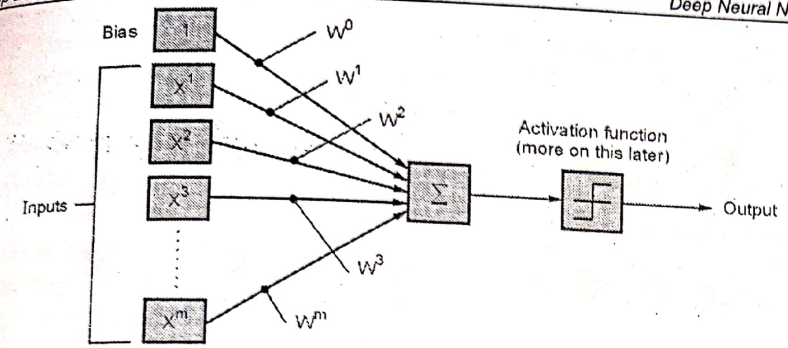


Fig. 2.4.2 Perception setup

1. First layer act as a set of feature detectors that are hardwired to the input signals to detect specific features.
2. Second layer i.e. output layer takes the outputs of the feature detectors in the first layer and classifies the given input pattern.

- Learning is initiated by making adjustments to the relevant connection strengths and a threshold value θ.
- Here we consider only two class problem. Here output layer usually has only a single node. For an n-class problem (n > 3), the output layer usually has n-nodes, each corresponding to a class and the output node with the largest value indicates which class the input vector belongs to.
- In the first stage, the linear combination of inputs is calculated. Each value of input array is associated with its weight value, which is normally between 0 and 1. Also, the summation function often takes an extra input value Theta with weight value of 1 to represent threshold or bias of a neuron.

a. The term x_i is referred to as **active or excitatory** if its value is 1.

b. If the value is 0 then it is **inactive**.

c. If the value is – 1 then it is **inhibitory**.

- The output unit is a linear threshold element with a threshold value θ :

$$0 = f\left(\sum_{i=1}^{n} w_i x_i - \theta\right)$$

$$= f\left(\sum_{i=1}^{n} w_i x_i + w_0\right),\ w_0 = -\theta$$

$$= f\left(\sum_{i=1}^{n} w_i x_i\right),\ x_0 \equiv 1$$

where w_i is a modifiable weight associated with an incoming signal x_i.

- Fig. 2.4.3 shows the bias term w_0.

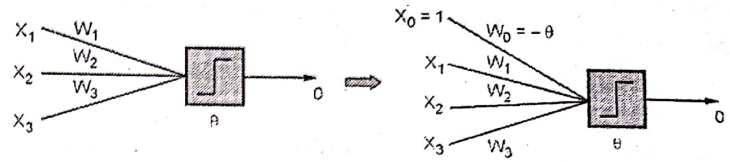


Fig. 2.4.3 Bias term W_0

- The function y = f(x) describes relationship, an input-output mapping from x to y.
- The equation (2.4.1), the f(.) is the **activation function** of the perceptron and it is typically either a **signum function** sgn(x) or **step function** step(x) :

$$\text{sgn}(x) = \begin{cases} 1 & \text{if } x > 0, \\ -1 & \text{otherwise} \end{cases}$$

$$\text{step}(x) = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{otherwise} \end{cases} \qquad \text{... (2.4.1)}$$

- The sum-of-product value is then passed into the second stage to perform the activation function which generates the output from the neuron. The activation function "squashes" the amplitude of the output in the range of [0, 1] or [– 1, 1] alternately. The behavior of the activation function will describe the characteristics of an artificial neuron model.
- The basic learning algorithm for a single layer perceptron repeats the following steps until the weights converge :
  1. Select an input vector x from the training data set.
  2. If the perceptron gives an incorrect response, modify all connection weights w_i according to

$$\Delta w_i = \eta\, t_i X_i$$

Where t_i is a target output and η is a learning state.

Perceptron Convergence Theorem

Theorem : If there is a set of weights that correctly classify the (linearly separable) training patterns, then the learning algorithm will find one such weight set, w^* in a finite number of iterations.

Assumptions :

1. At least one such set of weights, w^*, exists and

Scanned with OKEN Scanner

2. There are a finite number of training patterns.
3. The threshold function is uni-polar (output is 0 or 1).

2. Exclusive OR problem

- XOR problem is a pattern recognition problem in neural network.
- Neural networks can be used to classify boolean functions depending on their desired outputs.
- For a two input binary XOR problem, the desired output is given in the form of truth table.

	X	Y	Class
Desired I/O pair 1	0	0	0
Desired I/O pair 2	0	1	1
Desired I/O pair 3	1	0	1
Desired I/O pair 4	1	1	0

- The XOR problem is not **linearly separable**. We cannot use a single layer perceptron to construct a straight line to partition the two dimensional input space into two regions, each containing only data points of the same class.
- Let us consider following four equations :

$$0 \times w_1 + 0 \times w_2 + w_0 \le 0 \Leftrightarrow w_0 \le 0,$$

$$0 \times w_1 + 1 \times w_2 + w_0 > 0 \Leftrightarrow w_0 > -w_2,$$

$$1 \times w_1 + 0 \times w_2 + w_0 > 0 \Leftrightarrow w_0 > -w_1,$$

$$1 \times w_1 + 1 \times w_2 + w_0 \le 0 \Leftrightarrow w_0 \le -w_1 - w_2$$

2.4.2 Multi-Layer Feed Forward Network

- A multilayer feed-forward neural network is a network consisting of multiple layers of units, all of which are adaptive. The network is not allowed to have cycles from later layers back to earlier layers, hence the name "feed-forward". Let us consider a network with a single complete hidden layer. i.e., the network consists of some input nodes, some output nodes, and a set of hidden nodes. Every hidden node takes inputs from each of the input nodes and feeds into each of the output nodes.

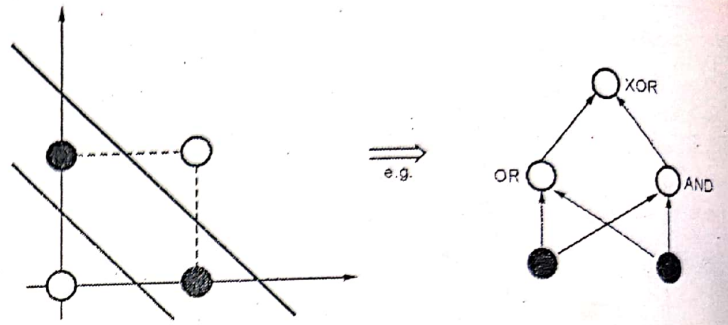


Fig. 2.4.4

- In multi-layer feed forward neural networks, the sigmoid activation function, defined by $g(x) = \frac{1}{1+e^{-x}}$ is normally used.
- A Multi-Layer Perceptron (MLP) has the same structure of a single layer perceptron with one or more hidden layers. An MLP is a network of simple neurons called perceptrons.
- A typical multilayer perceptron network consists of a set of source nodes forming the input layer, one or more hidden layers of computation nodes and an output layer of nodes.
- It is not possible to find weights which enable single layer perceptrons to deal with non-linearly separable problems like XOR :
- Multi-layer perceptrons are able to cope with non-linearly separable problems.
- Each neuron in one layer has direct connections to all the neurons of the subsequent layer. MLP can implement nonlinear discriminants (for classification) and nonlinear regression functions (for regression).
- Historically, the problem was that there were no known learning algorithms for training MLPs. Fortunately; it is now known to be quite straightforward. The procedure for finding a gradient vector in the network structure is generally referred to as **backpropagation**. Because the gradient vector is calculated in the direction opposite to the flow of the output of each node.
- Procedure of backpropagation :
  1. The output values are compared with the target to compute the value of some predefined error function.
  2. The error is then fedback through the network.

Scanned with OKEN Scanner

3. Using this information, the algorithm adjusts the weights of each connection in order to reduce the value of the error function.

- Continue this process until the connection weights in the network have been adjusted so that the network output has converged, to an acceptable level, with the desired output.
- If we use the gradient vector in a simple steepest descent method, the resulting learning paradigm is often referred to as the **backpropagation** learning rule. Backpropagation works by approximating the non-linear relationship between the input and the output by adjusting the weight values internally.
- Generally, the backpropagation network has two stages, training and testing. During the training phase, the network is "shown" sample inputs and the correct classifications. For example, the input might be an encoded picture of a face, and the output could be represented by a code that corresponds to the name of the person.
- Fig. 2.4.5 shows three most commonly used activation functions in backpropagation MLPs.

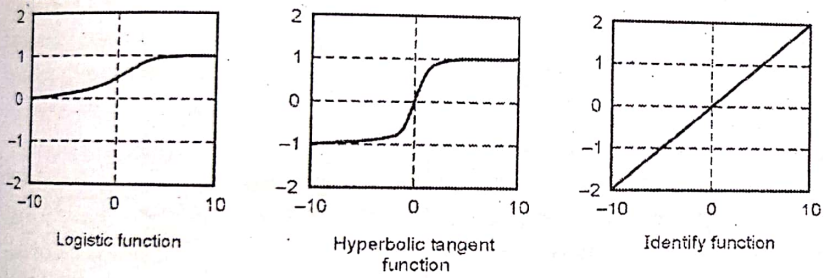


Fig. 2.4.5 Activation function

Logistic function :

$$f(x) = \frac{1}{1+e^{-x}}$$

Hyperbolic tangent function :

$$f(x) = \tanh(x/2) = \frac{1-e^{-x}}{1+e^{-x}}$$

Identity function :

$$f(x) = x$$

- Both the hyperbolic tangent function and logistic function approximate the signum and step function respectively. Sometimes these two function are referred to as **squashing functions** since the inputs to these functions are squashed to the range [0, 1] or [– 1, 1].
- These functions are also called **sigmoidal functions** because their S-shaped curves exhibits smoothness and asymptotic properties.
- A learning process is organized through a learning algorithm, which is a process of updating the weights in such a way that a machine learning tool implements a given input/output mapping with no errors or with some minimal acceptable error.
- Any learning algorithm is based on a certain learning rule, which determines how the weights shall be updated if the error occurs.

Backpropagation Learning Rule

- The **net input** of a node is defined as the weighted sum of the incoming signals plus a bias term. Fig. 2.4.6 shows the backpropagation MLP for node j. The net input and output of node j is as follows :

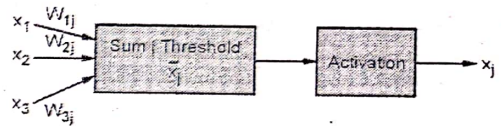


Fig. 2.4.6 Backpropagation MLP for node j

$$\bar{X}_j = \Sigma_i + W_{ij} + W_j$$

$$x_j = f(\bar{X}_j) = \frac{1}{1+\exp(-\bar{X}_j)}$$

Where

x_i is the output of node i located in any one of the pervious layers,

W_{ij} is the weight associated with the link connecting nodes i and j,

W_j is the bias of node j.

- Internal parameters associated with each node j is the weight W_{ij}. So changing the weights of the node will change the behaviour of the whole backpropagation MLP.
- Fig. 2.4.7 shows two layer backpropagation MLP.
- The above backpropagation MLP will refer to as a 3-4-3 network, corresponding to the number of nodes in each layer.
- The backward error propagation also known as the backpropagation (BP) or the

Scanned with OKEN Scanner

Generalized Delta Rule (GDR). A squared error measure for the pth input-output pair is defined as

$$E_p = \sum_k (d_k - x_k)^2$$

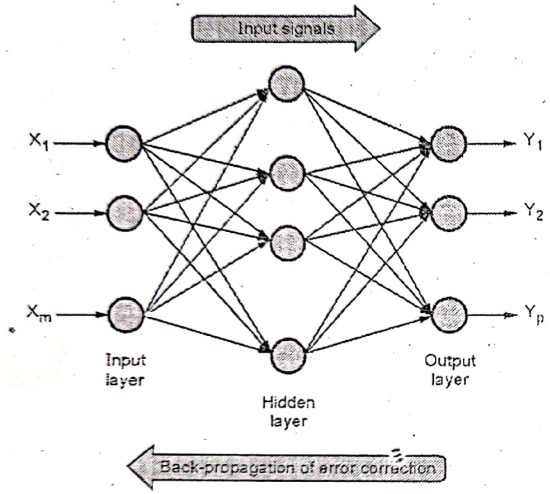


Fig. 2.4.7 Two layer backpropagation MLP

Where d_k is the desired output for node k and x_k is the actual output for node k when the input part of the pth data pair is presented.

- To find the gradient vector, an error term for node i is defined as :

$$\bar{\varepsilon}_i = \frac{\partial + E_p}{\partial \bar{X}_i}$$

- The partial derivative can be rewritten as product of two terms using chain rule for partial differentiation :

$$\frac{\partial E(t)}{\partial w_{ij}(t)} = \frac{\partial E(t)}{\partial a_i(t)} \bullet \frac{\partial a_i(t)}{w_{ij}(t)}$$

- Features of the delta rule are as follows :
  1. Simplicity
  2. Distributed learning : Learning is not reliant on central control of the network.
  3. Online learning : Weights are updated after presentation of each pattern.

Rules for Feedforward Multilayer Perceptron

- The training algorithm is called Error Back Propagation (EBP) training algorithm. If a submitted pattern provides an output far from desired value, the weights and thresholds are adjusted so that the current mean square classification error is reduced.
- The training is repeated for all patterns until the training set provide an acceptable overall error. Usually the mapping error is computed over the full training set.
- Error back propagation algorithm is working in two stages :
  1. The trained network operates feed-forward to obtain output of the network
  2. The weight adjustment propagate backward from output layer through hidden layer toward input layer.

2.4.3 Recurrent Neural Network

- A recurrent neural network is a type of neural network that contains loops, allowing information to be stored within the network.
- A RNN is particularly useful when a sequence of data is being processed to make a classification decision or regression estimate but it can also be used on non-sequential data. Recurrent neural networks are typically used to solve tasks related to time series data.
- Applications of recurrent neural networks include natural language processing, speech recognition, machine translation, character-level language modeling, image classification, image captioning, stock prediction and financial engineering.

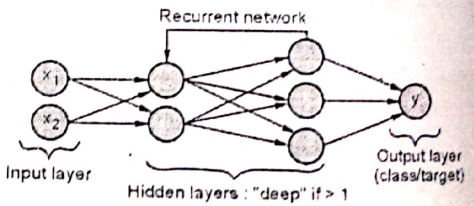


Fig. 2.4.8 Architecture of RNN

- Fig. 2.4.8 shows architecture of recurrent neural network.
- Recurrent Neural Networks can be thought of as a series of networks linked together. They often have a chain-like architecture, making them applicable for tasks such as speech recognition, language translation, etc.
- An RNN can be designed to operate across sequences of vectors in the input, output, or both. For example, a sequenced input may take a sentence as an input and output a positive or negative sentiment value. Alternatively, a sequenced output may take an image as an input and produce a sentence as an output.

Scanned with OKEN Scanner

2.5 Training Neural Networks

- Well-trained artificial neural network has weights that amplify the signal and creates the noise. A bigger weight signifies a tighter correlation between a signal and the network's outcome. Inputs paired with large weights will affect the network's interpretation of the data more than inputs paired with smaller weights.
- The process of learning for any learning algorithm using weights is the process of readjusting the weights and biases. In most datasets, certain features are strongly correlated with certain labels. Neural networks learn these relationships blindly by making a guess based on the inputs and weights.
- The loss functions in optimization algorithms, such as stochastic gradient descent (SGD), reward the network for good guesses and penalize it for bad ones.

2.5.1 Backpropagation

- Backpropagation is a training method used for a multi-layer neural network. It is also called the generalized delta rule. It is a gradient descent method which minimizes the total squared error of the output computed by the net.
- The backpropagation algorithm looks for the minimum value of the error function in weight space using a technique called the delta rule or gradient descent. The weights that minimize the error function is then considered to be a solution to the learning problem.
- Backpropagation is a systematic method for training multiple layer ANN. It is a generalization of Widrow-Hoff error correction rule. 80 % of ANN applications uses backpropagation.
- Fig. 2.5.1 shows backpropagation network.

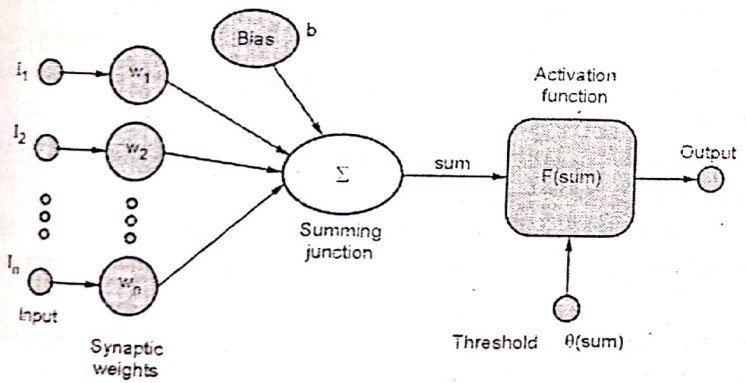


Fig. 2.5.1 Backpropagation network

- Consider a simple neuron :
  a. Neuron has a summing junction and activation function.
  b. Any non linear function which differentiable everywhere and increases everywhere with sum can be used as activation function.
  c. Examples : Logistic function, Arc tangent function, Hyperbolic tangent activation function.
- These activation function makes the multilayer network to have greater representational power than single layer network only when non-linearity is introduced.
- **Need of hidden layers :**
  1. A network with only two layers (input and output) can only represent the input with whatever representation already exists in the input data.
  2. If the data is discontinuous or non-linearly separable, the innate representation is inconsistent, and the mapping cannot be learned using two layers (Input and Output).
  3. Therefore, hidden layer(s) are used between input and output layers
- **Weights** connects unit (neuron) in one layer only to those in the next higher layer. The output of the unit is scaled by the value of the connecting weight, and it is fed forward to provide a portion of the activation for the units in the next higher layer.
- Backpropagation can be applied to an artificial neural network with any number of hidden layers. The training objective is to adjust the weights so that the application of a set of inputs produces the desired outputs.
- **Training procedure :** The network is usually trained with a large number of input - output pairs.
  1. Generate weights randomly to small random values (both positive and negative) to ensure that the network is not saturated by large values of weights.
  2. Choose a training pair from the training set.
  3. Apply the input vector to network input.
  4. Calculate the network output.
  5. Calculate the error, the difference between the network output and the desired output.
  6. Adjust the weights of the network in a way that minimizes this error.
  7. Repeat steps 2 - 6 for each pair of input-output in the training set until the error for the entire system is acceptably low.

Scanned with OKEN Scanner

Forward pass and backward pass :

- Backpropagation neural network training involves two passes.
  1. In the forward pass, the input signals moves forward from the network input to the output.
  2. In the backward pass, the calculated error signals propagate backward through the network, where they are used to adjust the weights.
  3. In the forward pass, the calculation of the output is carried out, layer by layer, in the forward direction. The output of one layer is the input to the next layer.
- In the reverse pass,
  a. The weights of the output neuron layer are adjusted first since the target value of each output neuron is available to guide the adjustment of the associated weights, using the delta rule.
  b. Next, we adjust the weights of the middle layers. As the middle layer neurons have no target values, it makes the problem complex.
- **Selection of number of hidden units :** The number of hidden units depends on the number of input units.
  1. Never choose h to be more than twice the number of input units.
  2. You can load p patterns of I elements into log2 p hidden units.
  3. Ensure that we must have at least 1/e times as many training examples.
  4. Feature extraction requires fewer hidden units than inputs.
  5. Learning many examples of disjointed inputs requires more hidden units than inputs.
  6. The number of hidden units required for a classification task increases with the number of classes in the task. Large networks require longer training times.

Factors Influencing Backpropagation training

- The training time can be reduced by using :
  1. **Bias :** Networks with biases can represent relationships between inputs and outputs more easily than networks without biases. Adding a bias to each neuron is usually desirable to offset the origin of the activation function. The weight of the bias is trainable similar to weight except that the input is always + 1.
  2. **Momentum :** The use of momentum enhances the stability of the training process. Momentum is used to keep the training process going in the same general direction analogous to the way that momentum of a moving object

behaves. In backpropagation with momentum, the weight change is a combination of the current gradient and the previous gradient.

2.5.2 Advantages and Disadvantages of Backpropagation

Advantages of backpropagation :

1. It is simple, fast and easy to program.
2. Only numbers of the input are tuned and not any other parameter.
3. No need to have prior knowledge about the network.
4. It is flexible.
5. A standard approach and works efficiently.
6. It does not require the user to learn special functions.

Disadvantages of backpropagation :

1. Backpropagation possibly be sensitive to noisy data and irregularity.
2. The performance of this is highly reliant on the input data.
3. Needs excessive time for training.
4. The need for a matrix-based method for backpropagation instead of mini - batch.

2.6 Activation Functions

- Activation functions also known as transfer function is used to map input nodes to output nodes in certain fashion.
- The activation function is the most important factor in a neural network which decided whether or not a neuron will be activated or not and transferred to the next layer.
- Activation functions help in normalizing the output between 0 to 1 or – 1 to 1. It helps in the process of backpropagation due to their differentiable property. During backpropagation, loss function gets updated, and activation function helps the gradient descent curves to achieve their local minima.
- Activation function basically decides in any neural network that given input or receiving information is relevant or it is irrelevant.
- These activation function makes the multilayer network to have greater representational power than single layer network only when non-linearity is introduced.
- The input to the activation function is sum which is defined by the following equation.

Scanned with OKEN Scanner

$$\text{sum} = I_1W_1 + I_2W_2 + \ldots + I_nW_n = \sum_{j=1}^{n} I_j W_j + b$$

- **Activation Function : Logistic Function**

$$f(\text{sum}) = \frac{1}{(1 + e^{-s * \text{sum}})}$$

$$= (1 + s^{-s \times \text{sum}}) - 1$$

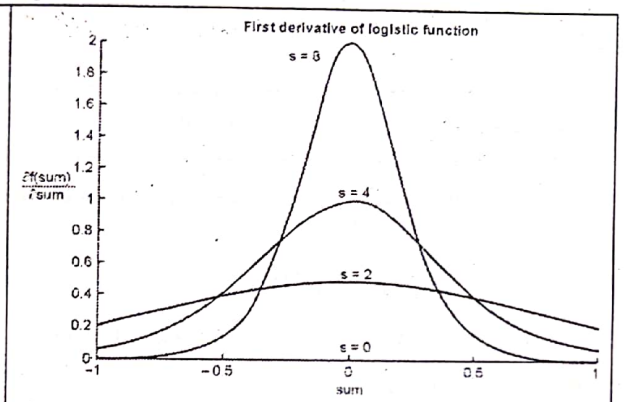


Fig. 2.6.1

- Logistic function monotonically increases from a lower limit (0 or – 1) to an upper limit (+ 1) as sum increases. In which values vary between 0 and 1, with a value of 0.5 when I is zero.
- **Activation Function : Arc Tangent**

$$f(\text{sum}) = \frac{2}{\pi} \tan^{-1} (s \times \text{sum})$$

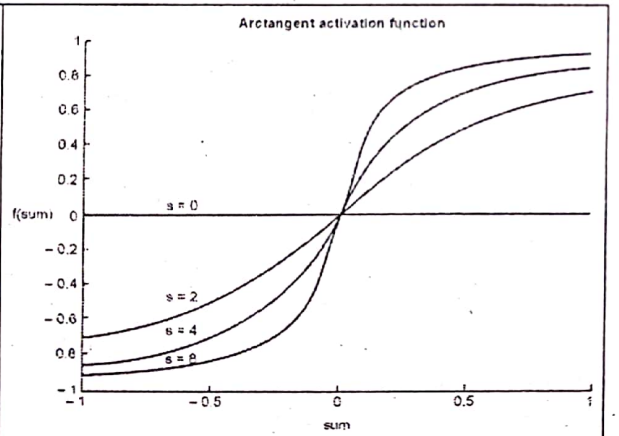


Fig. 2.6.2

- **Activation Function : Hyperbolic Tangent**

$$f(\text{sum}) = \tanh (s * I)$$

$$= \frac{e^{s \times \text{sum}} - e^{-s \times \text{sum}}}{e^{s \times \text{sum}} + e^{-s \times \text{sum}}}$$

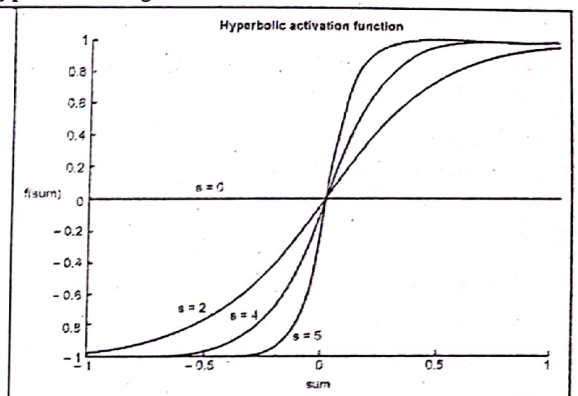


Fig. 2.6.3

2.6.1 Identity or Linear Activation Function

- A linear activation is a mathematical equation used for obtaining output vectors with specific properties.
- It is a simple straight line activation function where our function is directly proportional to the weighted sum of neurons or input.
- Linear activation functions are better in giving a wide range of activations and a line of a positive slope may increase the firing rate as the input rate increases.
- Fig. 2.6.4 shows identity function.

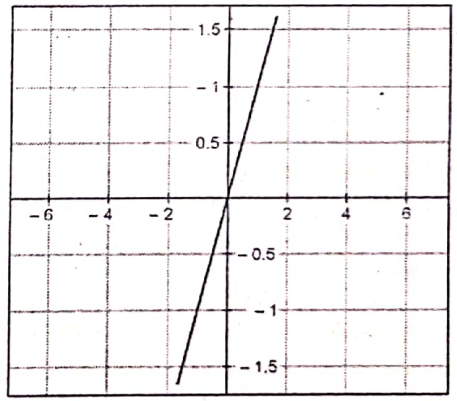


Fig. 2.6.4

Scanned with OKEN Scanner

2.7.1 Loss Function Notation

- Loss function notation are as follows :
  a) N = the number of samples collected.
  b) P = the number of input features gathered
  c) M = the number of output features that have been observed.
  d) (X, Y) to denote the input and output data collected. there will be N such pairs where the input is a collection of P values and the output Y is a collection of M values. We will denote the ith pair in the dataset as X_i and Y_i.
  e) $\hat{Y}$ = output of the neural net.
  f) $h(X_i) = \hat{Y}_i$ = neural network transforming the input X_i to give the output.
  g) Thus $y_{i,j}$ refers to the jth feature observed in the ith sample collected.
  h) Loss function = L(W, b)

2.7.2 Loss Functions for Regression

- Loss functions for regression : Regression involves predicting a specific value that is continuous in nature. Estimating the price of a house or predicting stock prices are examples of regression because one works towards building a model that would predict a real-valued quantity.

Mean Square Error :

- Mean Sqaure Error (MSE) is the most commonly used regression loss function. MSE is the sum of squared distances between our target variable and predicted values.
- Mean Squared Error is the average of the squared differences between the actual and the predicted values. For a data point Y_i and its predicted value $\hat{Y}_i$, where n is the total number of data points in the dataset, the mean squared error is defined as :

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

- **Advantages :** For small errors, MSE helps converge to the minima efficiently, as the gradient reduces gradually.

- **Drawback :**
  a) Squaring the values does increases the rate of training, but at the same time, an extremely large loss may lead to a drastic jump during backpropagation, which is not desirable.
  b) MSE is also sensitive to outliers.

2.7.3 Loss Functions for Classification

- Loss functions for classification : Classification problems involve predicting a discrete class output. It involves dividing the dataset into different and unique classes based on different parameters so that a new and unseen record can be put into one of the classes.

1. Hinge Loss

- Hinge Loss is a specific loss function used by Support Vector Machines (SVM). This loss function will help SVM to make a decision boundary with a certain margin distance.
- The equation for hinge loss when data points must be categorized as – 1 or 1 is as follows :

$$L(W,b) = \frac{1}{N}\sum_{i=1}^{N}\max(0, 1 - y_{ij} \times \hat{y}_{ij})$$

- Hinge loss is mostly used of binary classification.

Square hinge loss :

- There are many extensions of hinge loss are present to use with SVM models. One of the popular extensions is called Squared Hinge Loss. It simply calculates the square of the hinge loss value.
- Squared hinge loss has the effect of the smoothing the surface of the error function and making it numerically easier to work with.
- When the hinge loss requires better performance on a given binary classification problem it is mostly observed that a squared hinge loss may be appropriate to use. As using the hinge loss function, the target variable must be modified to have values in the set {– 1, 1}.
- It is simple to implement using python only we have to change the loss function name to "squared_hinge" in compile () function when building the model.
- A typical application can be classifying email into 'spam' and 'not spam' and we are only interested in the classification accuracy. Let us see how squared Hinge can be

Scanned with OKEN Scanner

used with Keras. It just involves specifying it as the used loss function during the model compilation step :

```
#Compile the model
model.compile(loss=squared_hinge,optimizer=tensorflow.keras.optimizers.Adam(lr=0.03), metrics=['accuracy'])
```

2.8 Hyper Parameters

- Hyperparameters are parameters whose values control the learning process and determine the values of model parameters that a learning algorithm ends up learning.
- While designing a machine learning model, one always has multiple choices for the architectural design for the model. This creates a confusion on which design to choose for the model based on its optimality. And due to this, there are always trials for defining a perfect machine learning model.
- The parameters that are used to define these machine learning models are known as the hyperparameters and the rigorous search for these parameters to build an optimized model is known as hyperparameter tuning.
- Hyperparameters are not model parameters, which can be directly trained from data. Model parameters usually specify the way to transform the input into the required output, whereas hyperparameters define the actual structure of the model that gives the required data.

2.8.1 Layer Size

- Layer size is defined by the number of neurons in a given layer. Input and output layers are relatively easy to figure out because they correspond directly to how our modeling problem handles input and output.
- For the input layer, this will match up to the number of features in the input vector. For the output layer, this will either be a single output neuron or a number of neurons matching the number of classes we are trying to predict.
- It is obvious that a neural network with 3 layers will give better performance than that of 2 layers. Increasing more than 3 doesn't help that much in neural networks. In the case of CNN, an increasing number of layers makes the model better.

2.8.2 Magnitude : Learning Rate

- The amount that the weights are updated during training is referred to as the step size or the learning rate. Specifically, the learning rate is a configurable hyper-parameter used in the training of neural networks that has a small positive value, often in the range between 0.0 and 1.0.
- For example, if learning rate is 0.1, then the weights in the network are updated 0.1^* (estimated weight error) or 10 % of the estimated weight error each time the weights are updated. The learning rate hyper-parameter controls the rate or speed at which the model learns.
- Learning rates are tricky because they end up being specific to the dataset and even to other hyper-parameters. This creates a lot of overhead for finding the right setting for hyper-parameters.
- Large learning rates () make the model learn faster but at the same time it may cause us to miss the minimum loss function and only reach the surrounding of it. In cases where the learning rate is too large, the optimizer overshoots the minimum and the loss updates will lead to divergent behaviours.
- On the other hand, choosing lower learning rate values gives a better chance of finding the local minima with the trade-off of needing larger number of epochs and more time.
- **Momentum** can accelerate learning on those problems where the high-dimensional weight space that is being navigated by the optimization process has structures that mislead the gradient descent algorithm, such as flat regions or steep curvature.

2.9 Regularization

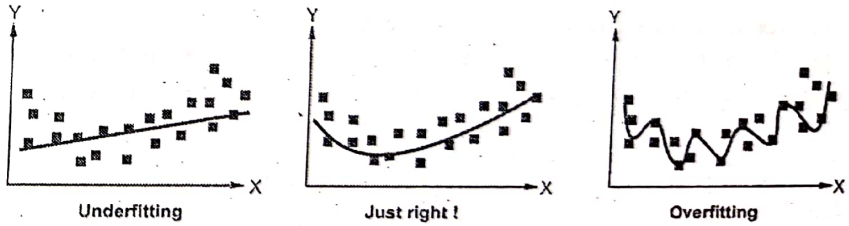


Fig. 2.9.1

- Just have a look at the above figure, and we can immediately predict that once we try to cover every minutest feature of the input data, there can be irregularities in

Scanned with OKEN Scanner

the extracted features, which can introduce noise in the output. This is referred to as "Overfitting".

- This may also happen with the lesser number of features extracted as some of the important details might be missed out. This will leave an effect on the accuracy of the outputs produced. This is referred to as "Underfitting".
- This also shows that the complexity for processing the input elements increases with overfitting. Also, neural networks being a complex interconnection of nodes, the issue of overfitting may arise frequently.
- To eliminate this, regularization is used, in which we have to make the slightest modification in the design of the neural network, and we can get better outcomes.

2.9.1 Regularization in Machine Learning

- One of the most important factors that affect the machine learning model is overfitting.
- The machine learning model may perform poorly if it tries to capture even the noise present in the dataset applied for training the system, which ultimately results in overfitting. In this context, noise doesn't mean the ambiguous or false data, but those inputs which do not acquire the required features to execute the machine learning model.
- Analyzing these data inputs may surely make the model flexible, but the risk of overfitting will also increase accordingly.
- One of the ways to avoid this is to cross validate the training dataset and decide accordingly the parameters to include that can increase the efficiency and performance of the model.
- Let this be the simple relation for linear regression :

$$Y \approx b_0 + b_1X_1 + b_2X_2 + \dots b_pX_p$$

Where Y = Learned relation

β = Co-efficient estimators for different variables and/or predictors (X)

- Now, we shall introduce a loss function, that implements the fitting procedure, which is referred to as "Residual Sum of Squares" or RSS.
- The co-efficient in the function is chosen in such a way that it can minimize the loss function easily.

Hence,

$$RSS = \sum_{i=1}^{n}\left(Y_i - \beta_0 - \sum_{j=1}^{p}\beta_j X_{ij}\right)^2$$

- Above equation will help in adjusting the co-efficient function depending on the training dataset.
- In case noise is present in the training dataset, then the adjusted co-efficient won't be generalized when the future datasets will be introduced. Hence, at this point, regularization comes into picture and makes this adjusted co-efficient shrink towards zero.
- One of the methods to implement this is the ridge regression, also known as L2 regularization. Lets have a quick overview on this.

2.9.2 Ridge Regression (L2 Regularization)

- Ridge regression, also known as L2 regularization, is a technique of regularization to avoid the overfitting in training data set, which introduces a small bias in the training model, through which one can get long term predictions for that input.
- In this method, a penalty term is added to the cost function. This amount of bias altered to the cost function in the model is also known as ridge regression penalty.
- Hence, the equation for the cost function, after introducing the ridge regression penalty is as follows :

$$\sum_{i=1}^{m}(y_i - y'_i)^2 = \sum_{i=1}^{m}\left(Y_i - \sum_{j=1}^{n}\beta_j \times X_{ij}\right)^2 + \lambda\sum_{j=0}^{n}\beta_j^2$$

Here, λ is multiplied by the square of the weight set for the individual feature of the input data. This term is ridge regression penalty.

- It regularizes the co-efficient set for the model and hence the ridge regression term deduces the values of the coefficient, which ultimately helps in deducing the complexity of the machine learning model.
- From the above equation, we can observe that if the value of tends to zero, the last term on the right - hand side will tend to zero, thus making the above equation a representation of a simple linear regression model.
- Hence, lower the value of , the model will tend to linear regression.
- This model is important to execute the neural networks for machine learning, as there would be risks of failure for generalized linear regression models, if there are dependencies found between its variables. Hence, ridge regression is used here.

Scanned with OKEN Scanner

2.9.3 Lasso Regression (L1 Regularization)

- One more technique to reduce the overfitting and thus the complexity of the model is the lasso regression.
- Lasso regression stands for Least Absolute and Selection Operator and is also sometimes known as L1 regularization.
- The equation for the lasso regression is almost same as that of the ridge regression, except for a change that the value of the penalty term is taken as the absolute weights.
- The advantage of taking the absolute values is that its slope can shrink to 0, as compared to the ridge regression, where the slope will shrink it near to 0.
- The following equation gives the cost function defined in the Lasso regression :

$$\sum_{i=1}^{m}(y_i - y'_i)^2 = \sum_{i=1}^{m}\left(Y_i - \sum_{j=0}^{n}\beta_j \times X_{ij}\right)^2 + \lambda \sum_{j=0}^{n}|\beta_j|^2$$

- Due to the acceptance of absolute values for the cost function, some of the features of the input dataset can be ignored completely while evaluating the machine learning model and hence the feature selection and overfitting can be reduced to much extent.
- On the other hand, ridge regression does not ignore any feature in the model and includes it all for model evaluation. The complexity of the model can be reduced using the shrinking of co-efficient in the ridge regression model.

2.9.4 Dropout

- Dropout was introduced by "Hinton et al"and this method is now very popular. It consists of setting to zero the output of each hidden neuron in chosen layer with some probability and is proven to be very effective in reducing overfitting.
- Fig. 2.9.2 shows dropout regulations.
- To achieve dropout regularization, some neurons in the artificial neural network are randomly disabled. That prevents them from being too dependent on one another as they learn the correlations. Thus, the neurons work more independently, and the artificial neural network learns multiple independent correlations in the data based on different configurations of the neurons.
- It is used to improve the training of neural networks by omitting a hidden unit. It also speeds training.

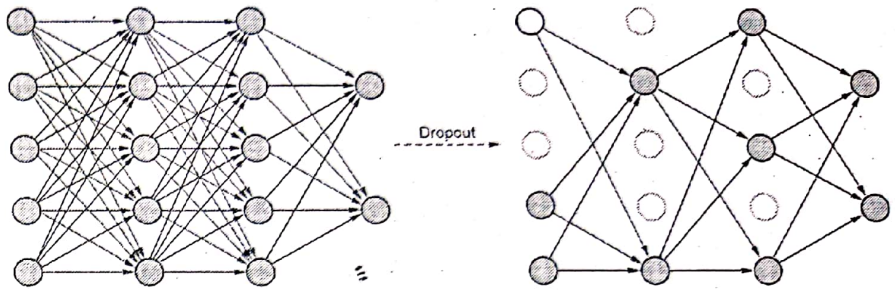


Fig. 2.9.2 Dropout regulation

- Dropout is driven by randomly dropping a neuron so that it will not contribute to the forward pass and back-propagation.
- Dropout is an inexpensive but powerful method of regularizing a broad family of models.

2.9.5 DropConnect

- DropConnect, known as the generalized version of Dropout, is the method used for regularizing deep neural networks. Fig. 2.9.3 shows dropconnect.

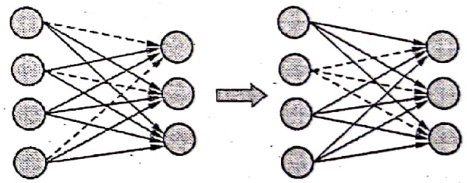


Fig. 2.9.3 Dropconnect

- DropConnect has been proposed to add more noise to the network. The primary difference is that instead of randomly dropping the output of the neurons, we randomly drop the connection between neurons.
- In other words, the fully connected layer with DropConnect becomes a sparsely connected layer in which the connections are chosen at random during the training stage.

Scanned with OKEN Scanner

2.9.6 Difference between L1 and L2 Regularization

L1 regulation	L2 regulation
Calculate the sum of the absolute values of the weights, called L1.	Calculate the sum of the squared values of the weights, called L2.
L1 has a sparse solution	L2 has a non sparse solution
It has built in feature selection	There is no feature selection
L1 is robust to outliers	L2 is not robust to outliers
L1 generates model that are simple and interpretable but cannot learn complex patterns	L2 regularization is able to learn complex data patterns
L1 has multiple solutions	L2 has one solution
It is equivalent to MAP Bayesian estimation with Laplace prior	It is equivalent to MAP Bayesian estimation with Gaussian prior.
A regression model that uses L1 regularization technique is called Lasso Regression	model which uses L2 is called Ridge Regression

2.10 Deep Feedforward Networks

- Deep feedforward networks is also called feedforward neural networks or multilayer perceptrons.
- Feedforward neural networks are called networks because they are typically represented by composing together many different functions. The model is associated with a directed acyclic graph describing how the functions are composed together.
- For example : consider 3 functions $f^{(1)}$, $f^{(2)}$ and $f^{(3)}$ connected in a chain, to form $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$.
- These chain structures are the most commonly used structures of neural networks. In this case, $f^{(1)}$ is called the **first layer** of the network, $f^{(2)}$ is called the **second layer** and so on.
- The overall length of the chain gives the depth of the model. It is from this terminology that the name "deep learning" arises.

- The final layer of a feedforward network is called the **output layer**. During neural network training, we drive f(x) to match $f^*(x)$.
- The training data provides us with noisy, approximate examples of $f^*(x)$ evaluated at different training points.
- The training examples specify directly what the output layer must do at each point x; it must produce a value that is close to y. The behavior of the other layers is not directly specified by the training data.
- The learning algorithm must decide how to use those layers to produce the desired output, but the training data does not say what each individual layer should do.
- Instead, the learning algorithm must decide how to use these layers to best implement an approximation of f^*.
- Because the training data does not show the desired output for each of these layers, these layers are called **hidden layers**.
- Feedforward networks have introduced the concept of a hidden layer and this requires us to choose the activation functions that will be used to compute the hidden layer values.
- Finally, these networks are called neural because they are loosely inspired by neuroscience. Each hidden layer of the network is typically vector-valued. The dimensionality of these hidden layers determines the **width** of the model

2.10.1 Learning Concept of XOR

- XOR problem is a pattern recognition problem in neural network.
- The XOR function is an operation on two binary values, x_1 and x_2. When exactly one of these binary values is equal to 1, the XOR function returns 1. Otherwise, it returns 0.
- The XOR function provides the target function $y = f^*(x)$ that we want to learn. Our model provides a function $y = f(x; \theta)$ and our learning algorithm will adapt the parameters θ to make f as similar as possible to f^*.
- Neural networks can be used to classify Boolean functions depending on their desired outputs.
- The XOR problem is not **linearly separable**. We cannot use a single layer perceptron to construct a straight line to partition the two dimensional input space into two regions, each containing only data points of the same class.

Scanned with OKEN Scanner

2.11 Gradient-Based Learning

- Much of machine learning can be written as an optimization problem.
- Example loss functions : Logistic regression, linear regression, principle component analysis, neural network loss.
- A very efficient way to train logistic models is with **Stochastic Gradient Descent (SGD).**
- One challenge with training on power law data (i.e. most data) is that the terms in the gradient can have very different strengths
- The idea behind stochastic gradient descent is iterating a weight update based on the gradient of loss function :

$$\bar{w}(k+1) = \bar{w}(k) - \gamma \nabla L(\bar{w})$$

- Logistic regression is designed as a **binary classifer** (output say {0, 1}) but actually **outputs the probability** that the input instance is in the "1" class.
- A logistic classifier has the form :

$$p(X) = \frac{1}{1 + \exp(-X\beta)}$$

where $X = (X_1, \ldots, X_n)$ is a vector of features.

- Stochastic gradient has some serious limitations however, especially if the gradients vary widely in magnitude. Some coefficients change very fast, others very slowly.
- This happens for text, user activity and social media data (and other power-law data), because gradient magnitudes scale with feature frequency, i.e. over several orders of magnitude.
- It is not possible to set a single learning rate that trains the frequent and infrequent features at the same time.
- An example of stochastic gradient descent with perceptron loss is shown as follows :

from sklearn.linear_model import SGDClassifier

2.11.1 Finding the Optimal Hyper-parameters through Grid Search

- In statistics, hyperparameter is a parameter from a prior distribution; it captures the prior belief before data is observed.
- In any machine learning algorithm, these parameters need to be initialized before training a model.

- **Model hyperparameters** *are the properties that govern the entire training process.*
- Hyperparameters are important because they directly control the behaviour of the training algorithm and have a significant impact on the performance of the model is being trained.
- Choosing appropriate hyperparameters plays a crucial role in the success of our neural network architecture. Since it makes a huge impact on the learned model.
- For example, if the learning rate is too low, the model will miss the important patterns in the data. If it is high, it may have collisions.
- Choosing good hyperparameters gives two benefits :
  1. Efficiently search the space of possible hyperparameters.
  2. Easy to manage a large set of experiments for hyperparameter tuning.
- **The process of finding most optimal hyperparameters in machine learning is called hyperparameter optimisation.**
- Grid search is a very traditional technique for implementing hyperparameters. It brute force all combinations. Grid search requires to create two set of hyperparameters.

  1. Learning rate 2. Number of layers
- Grid search trains the algorithm for all combinations by using the two set of hyperparameters and measures the performance using "Cross Validation" technique.
- This validation technique gives assurance that our trained model got most of the patterns from the dataset.
- One of the best methods to do validation by using "K-Fold Cross Validation" which helps to provide ample data for training the model and ample data for validations.
- With this technique, we simply build a model for each possible combination of all of the hyperparameter values provided, evaluating each model and selecting the architecture which produces the best results.
- For example, say you have two continuous parameters α and β, where manually selected values for the parameters are the following :

$$\alpha \in \{0, 1, 2\}.$$
$$\beta \in \{.25, .50, .75\}$$

- Then the pairing of the selected hyperparametric values, H, can take on any of the following :

$$H \in \{(0, .25), (0, .50), (0, .75), (1, .25), (1, .50), (1, .75), (2, .25), (2, .50), (2, .75)\}$$

Scanned with OKEN Scanner

- Grid search will examine each pairing of and to determine the best performing combination. The resulting pairs, H, are simply each output that results from taking the Cartesian product of α and β.
- While straightforward, this "brute force" approach for hyperparameter optimization has some drawbacks. Higher-dimensional hyperparametric spaces are far more time consuming to test than the simple two-dimensional problem presented here.
- Also, because there will always be a fixed number of training samples for any given model, the model's predictive power will decrease as the number of dimensions increases. This is known as Hughes phenomenon.

2.11.2 Vanishing Gradient Problem

- When back-propagation is used, the earlier layers will receive very small updates compared to the later layers. This problem is referred to as the vanishing gradient problem.
- The vanishing gradient problem is essentially a situation in which a deep multilayer feed-forward network or a Recurrent Neural Network (RNN) does not have the ability to propagate useful gradient information from the output end of the model back to the layers near the input end of the model.
- Weight initialization is one technique that can be used to solve the vanishing gradient problem. It involves artificially creating an initial value for weights in a neural network to prevent the backpropagation algorithm from assigning weights that are unrealistically small.
- The most important solution to the vanishing gradient problem is a specific type of neural network called Long Short-Term Memory Networks (LSTMs).
- Indication of vanishing gradient problem :
  a) The parameters of the higher layers change to a great extent, while the parameters of lower layers barely change.
  b) The model weights could become 0 during training.
  c) The model learns at a particularly slow pace and the training could stagnate at a very early phase after only a few iterations.
- Some methods that are proposed to overcome the vanishing gradient problem:
  a) Residual neural networks (ResNets)
  b) Multi-level hierarchy
  c) Long short term memory (LSTM)
  d) Faster hardware

  e) ReLU
  f) Batch normalization

2.12 Deep Learning with Jupyter

- Jupyter is an open source project. The Jupyter console is a terminal frontend for kernels using the Jupyter protocol. The console can be installed with: pip install jupyter-console
- Jupyter is a web application perfect for this task. Jupyter works with Notebooks, documents that mix rich text including beautifully rendered math formulas, blocks of code and code output, including graphics.
- Jupyter main features are :
  1. inline code execution
  2. easy idea structuring
  3. nice displays of pictures and dataframe
- Jupyter Notebook is an open source web interface that enables to include text, video, audio, images.
- The main difference between Jupyter console and Jupyter notebooks is that the console functions in interactive mode. Whenever you type a line of code, it is immediately executed and you can see the results.
- If you want to write medium-length pieces of code, do a deep exploration of a dataset to tell a story, the notebook is better. If you want to test out code you're writing, or run quick commands, the console is better.
- A kernel is a program that runs and introspects the user's code: it provides computation and communication with the frontend interfaces, such as notebooks.
- The Jupyter Notebook Application has three main kernels : the IPython, IRkernel and IJulia kernels.
- Anaconda is the most widely used Python distribution for data science and comes pre-loaded with all the most popular libraries and tools.
- There are quite a number of packages that will help you to deploy your notebooks and that are part of the Jupyter ecosystem.
  1. docker-stacks will come in handy when user need stacks of Jupyter applications and kernels as Docker containers.
  2. ipywidgets provides interactive HTML and JavaScript widgets for the Jupyter architecture that combine front-end controls coupled to a Jupyter kernel.

Scanned with OKEN Scanner

3. jupyter-drive allows IPython to use Google Drive for file management.
4. jupyter-sphinx-theme to add a Jupyter Sphinx theme to your notebook. It will make it easier to create intelligent and beautiful documentation.
5. kernel_gateway is a web server that supports different mechanisms for spawning and communicating with Jupyter kernels. Look here to see some use cases in which this package can be come in handy.

- To create a new notebook, click on the New button, and select Notebook. A new browser tab opens and shows the Notebook interface as follows :

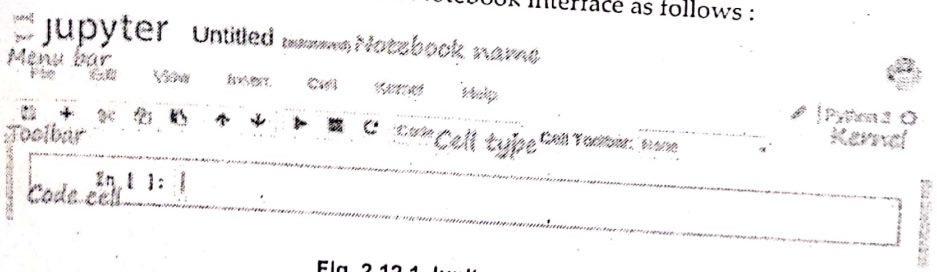


Fig. 2.12.1 Jupiter notebook

- Notebook consists of two cells : Markdown cells and code cell.
- Markdown cell is used for text and code cell for executing code by the kernel.
- Code cell contains various section like prompt number, input area, widget area and output area.
- Notebook model interface support edit mode and command mode.
- Try typing something like print("Hello World") into the cell. To run the code in the cell and see the output, click the Run button on the toolbar, or type Shift-Enter :

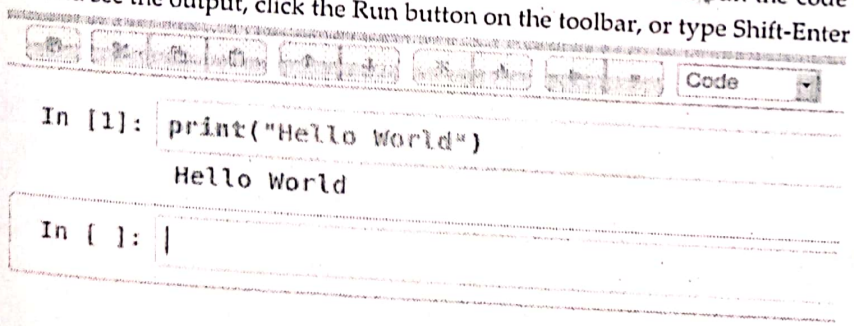


Fig. 2.12.2

- A Jupyter notebook is always organized as a sequence of so called 'cells' with each cell either containing some code or rich text created using the Markdown notation approach.

- When you click into the corresponding text field to add or modify the content of the cell, the bar color will change to green indicating that you are now in 'Edit mode'. Clicking anywhere outside of the text area of a cell will change back to 'Command mode'.

2.12.1 Magic Function

- Jupyter provides a number of so-called magic commands that can be used in code cells to simplify common tasks. Magic commands are interpreted by Jupyter and, for instance, transformed into Python code before the content is passed on to the kernel for execution
- Jupyter has a set of predefined 'magic functions' that you can call with a command line style syntax. There are two kinds of magics, line-oriented and cell-oriented.
- Line magics are prefixed with "%" and work like a command typed in your terminal. Line magics can be used as expression and their return value can be assigned to variable.
- Example of line magics are %autocall, %cd, %automagic, %dhist, %edit, %matplotlib and %env.
- Cell magics work on a block of code, not on a single line. They are prefixed with "%%". To close a block of code, when you are inside a cell magic function, hit Enter twice.

2.12.2 Using Jupyter Notebook

- The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text.
- Jupyter Notebook is often used for exploratory data analysis and visualization. The Jupyter Notebook has several menus that you can use to interact with your Notebook. Here is a list of some current menus:

  a) Edit b) View C) Insert D) Cell E) Kernel f) Widgets G) Help H) File
- File menu is used for creating new notebook and opening existing one.
- Edit menu perform operation like cut, copy, and paste cells.
- View menu is useful for toggling the visibility of the header and toolbar.
- Insert menu is for inserting cells above or below the currently selected cell.
- Cell menu allows you to run one cell, a group of cells, or all the cells.
- The Widgets menu is for saving and clearing widget state.
- The Kernel cell is for working with the kernel that is running in the background

Scanned with OKEN Scanner

2.12.3 Restarting the Kernel

- Juypter has a autocomplete feature that helps with the code writing and can save a lot of typing : While editing code in a code cell, you can press the TAB key and Jupyter will either automatically complement the name or keyword you are writing or provide you with a dropdown list of choices that you can pick from.
- Fig. 2.12.3 shows restarting the kernel.

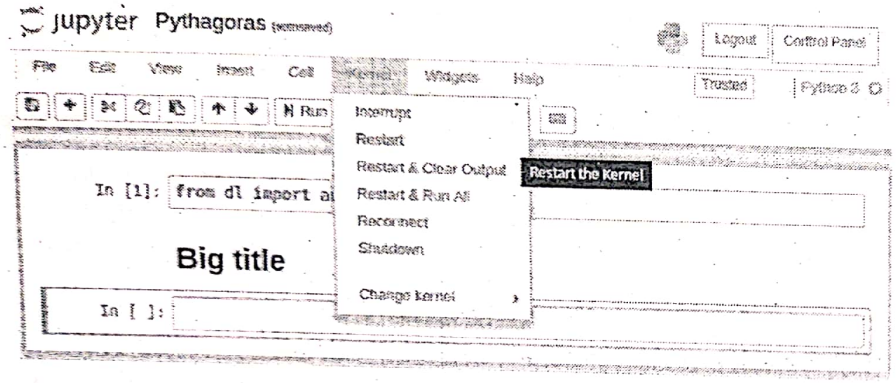


Fig. 2.12.3 Restarting the kernel

- Save your notebook locally to store your current progress
- In the notebook toolbar, click Kernel, then Restart
- Try testing your kernel by running a print statement in one of your notebook cells. If this is successful, you can continue to save and proceed with your work.
- If your notebook kernel is still timed out, try closing your browser and relaunching the notebook. When the notebook reopens, you will need to do "Cell -> Run All" or "Cell -> Run All Above" to regenerate the execution state.

2.12.4 Restoring a Checkpoint

- In Jupyter Notebook Workspaces, it's a good idea to save your checkpoints before you close the tab.
- To restore a checkpoint, choose the entry found on the File → Revert to Checkpoint menu.
- This menu makes it appear that you can have more than one checkpoint file, but the menu never has more than one entry in it.
- Fig. 2.12.4 shows menu for restoring a checkpoint.

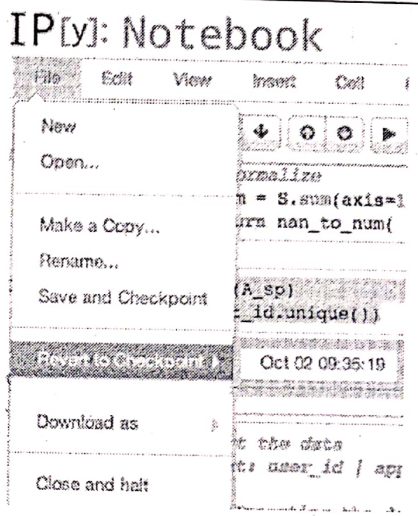


Fig. 2.12.4 Restoring a checkpoint

2.12.5 Performing Multimedia and Graphic Integration

- IPython Notebook support both a coding platform and a presentation platform.

Embedding plots and other images

- When you run %matplotlib inline, any plots you create appear as part of the document. Use %matplotlib inline - this only draws the images, not interactive / zoom-able but it works well.
- Following Fig. 2.12.5 shows example of embedding plot.

```
In [3]: %matplotlib inline
        import matplotlib.pyplot as plt
        plt.plot([0,0],[0,1]), linewidth=0, label='random diagonal')
        plt.show()
```

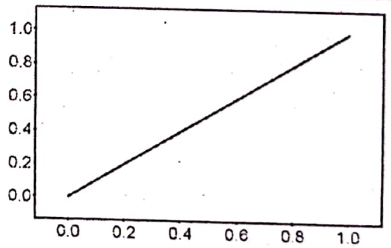


Fig. 2.12.5 Example of %matplotlib inline

Scanned with OKEN Scanner

- The display function is a general purpose tool for displaying different representations of objects. Think of it as print for these rich representations.

```
In [1]: from IPython.display import display
```

A few points :

a) Calling display on an object will send all possible representations to the Notebook.

b) These representations are stored in the Notebook document.

c) In general the Notebook will use the richest available representation.

- To work with images, use the Image class.

```
In [3]: from IPython.display import Image
In [4]: i = Image(url='img/python-logo.gif')
```

2.13 Deep Learning with Colab

- Colab (Colaboratory) is a product from Google Research, which is technically a hosted Jupyter Notebook service that requires no setup to use. It is cloud-based framework. With Google's Colab, one can access its GPUs and TPUs almost for free.
- Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning and data analysis.
- Colab has several useful features that make it helpful for the data science community. The following are some of the advantages :

a) Pre-installed data science libraries

b) Easy sharing and collaboration

c) Seamless integration with GitHub

d) Working with data from various sources

e) Automatic storage and version control

f) Access to hardware accelerators such as GPUs and TPUs

□□□

Notes

Scanned with OKEN Scanner