AIDS - ANN - Unit 3 - Associative Learning

Unit III

Unit III	Associative Learning	07(Hours)
Introduction, Associative Learning, Hopfield network, Error Performance in Hopfield networks, simulated annealing, Boltzmann machine and Boltzmann learning, State transition diagram and false minima problem, stochastic update, simulated annealing. Basic functional units of ANN for pattern recognition tasks: Pattern association, pattern classification and pattern mapping tasks.		
#Exemplar/Case Studies	Understanding catastrophic, Interference in neural nets	

Introduction, Associative Learning, Hopfield network, Error Performance in Hopfield networks, simulated annealing, Boltzmann machine and Boltzmann learning, State transition diagram and false minima problem, stochastic update. Basic functional units of ANN for pattern recognition tasks: Pattern association, pattern classification and pattern mapping tasks.

Introduction

What is Association?

A connection or cooperative link between similar groups of function which are made to achieve same goal.

Example:

Human identify objects based on their characteristics like shape, color, weight which they have seen earlier. When we say an apple, we imagine red color spherical object with shape we can hold in our hand.

But how machine will memorize an apple? Here is explanation from table.

Diameter	Weight	Red	Green	Blue	Name
2.8	10.1	172	85	3	**Grape**
3.9	6.2	166	78	1	**Grape**
11.2	108.5	157	98	2	**Apple**
10.2	120.3	161	78	2	**Orange**

For any object, machine will have their characteristics saved with reference to their size, weight, color combination. As per table, if weight, diameter and color combinations are matched then that object is what mentioned in last column.

Associative Learning

Associative learning is when a person or animal learns to connect two things.

Classical Conditioning: Learning happens when a neutral thing (like a sound) is repeatedly linked to something that naturally causes a reaction (like food). Over time, the neutral thing alone causes the reaction.

Example: A dog hears a bell every time it gets food. Eventually, just hearing the bell makes the dog drool.

Operant Conditioning: Learning happens through rewards or punishments. A behavior becomes more common if it’s rewarded and less common if it’s punished.

Example: A child gets a cookie for cleaning their room, so they clean more often. If they lose playtime for making a mess, they stop making messes.

Hopfield Network

An **Artificial Neural Network (ANN)** is inspired by the way the human brain recognizes patterns and processes information. Just like our brain handles emotions, thoughts, and memories through millions of signals, ANNs learn by adjusting connections between artificial neurons.

A **Hopfield Network** is a special type of **Recurrent Neural Network (RNN)**, introduced by **John Hopfield in 1982**, that is inspired by physics and magnetism.

Types of Hopfield Networks:

1. **Discrete Hopfield Network**

Every neuron is connected to *every* other neuron. The output is either **Binary (0/1)** or **Bipolar (-1/1)**.

Example: Used in **pattern recognition**, like recognizing handwritten digits.

2. **Continuous Hopfield Network**

Unlike discrete networks, it allows values **between 0 and 1** instead of fixed binary outputs.

Example: Used in **optimization problems**, like scheduling

These networks help in tasks like **memory recall, optimization, and associative pattern storage**, just like how the human brain remembers and associates information.

Each neuron in the network has three qualities to consider:

connections to other neurons — each neuron in the network is connected to all other neurons, and each connection has a unique strength, or weight, analogous to the strength of a synapse. These connection strengths are stored inside a matrix of weights.

activation — this is computed via the net input from other neurons and their respective connection weights, loosely analogous to the membrane potential of a neuron.

bipolar state — this is the output of the neuron, computed using the neuron's activation and a thresholding function, analogous to a neuron's 'firing state.' In this case, -1 and +1.

Structure & Architecture

- Each neuron has an inverting and a non-inverting output.
- Being fully connected, the output of each neuron is an input to all other neurons but not self.

[x1 , x2 , ... , xn] -> Input to the n given neurons.
[y1 , y2 , ... , yn] -> Output obtained from the n given neurons
Wij -> weight associated with the connection between the i th and the jth neuron.

Weights assigned can be of two types: **binary** (0 and 1) or **bipolar** (weights have no self-connection).

Steps for the Algorithm

1. **Initialize weights** w_{ij} to store patterns (using a training algorithm).
2. For each input vector y_i, perform steps 3-7.
3. Make initial activators of the network equal to the external input vector x.

$$Y_i = X_i \quad \text{for } i = 1 \text{ to } N$$

4. For each vector y_i, perform steps 5-7.
5. Calculate the total input of the network Y_{in} using the equation:

$$Y_{\text{input}}(i) = X_i + \sum_j (Y_j W_{ji})$$

6. Apply activation over the total input to calculate the output as per the equation:

$$Y_i = \begin{cases} 1, & \text{if } Y_{\text{in}} > \theta_i \\ Y_i, & \text{if } Y_{\text{in}} = \theta_i \\ 0, & \text{if } Y_{\text{in}} < \theta_i \end{cases}$$

where θ_i (threshold) is normally taken as 0.

7. Now, feedback the obtained output y_i to all other units. Thus, the activation vectors are updated.
8. Test the network for convergence.

Simulated Annealing

Simulated annealing is a technique used in AI to find solutions to optimization problems. In other words, simulated annealing can be used to find solutions to optimization problems by slowly changing the values of the variables in the problem until a solution is found.

It allows for small changes to be made to the solution, which means that it can escape from local minima and find the global optimum. Simulated annealing is not a guaranteed method of finding the best solution to an optimization problem, but it is a powerful tool that can be used to find good solutions in many cases.

Simulated annealing works by starting with a random solution and then slowly improving it over time. The key is to not get stuck in a local optimum, which can happen if the search moves too slowly.

How does it work?

We need to provide an initial solution so the algorithm knows where to start. This can be done in two ways:
(1) using prior knowledge about the problem to input a good starting point and
(2) generating a random solution.

1. Move all points 0 or 1 units in a random direction
2. Shift input elements randomly
3. Swap random elements in input sequence
4. Permute input sequence
5. Partition input sequence into a random number of segments and permute segments

Benefits of using Simulated Annealing?

1. The ability to find global optima.
2. The ability to escape from local optima.
3. The ability to handle constraints.
4. The ability to handle noisy data.
5. The ability to handle discontinuities.
6. The ability to find solutions in a fraction of the time required by other methods.
7. The ability to find solutions to problems that are difficult or impossible to solve using other methods.

Drawbacks of using Simulated Annealing?

It can be slow and may not always find the best solution. Additionally, it can be difficult to tune the parameters of the algorithm, which can lead to sub-optimal results.

Applications of Simulated Annealing

travelling salesman problem, the knapsack problem, and the satisfiability problem. It has also been used in image recognition

Boltzmann machine and Boltzmann learning

What is a Boltzmann Machine?

A Boltzmann machine is a neural network of symmetrically connected nodes that make their own decisions whether to activate. It uses a straightforward stochastic learning algorithm to discover “interesting” features that represent complex patterns in the database.
This is categorized under “unsupervised deep learning”.

Note: stochastic means having a random probability distribution or pattern that may be analyzed statistically but may not be predicted precisely.

The main purpose of Boltzmann Machine is to optimize the solution of a problem.

Types of Boltzmann Machines:

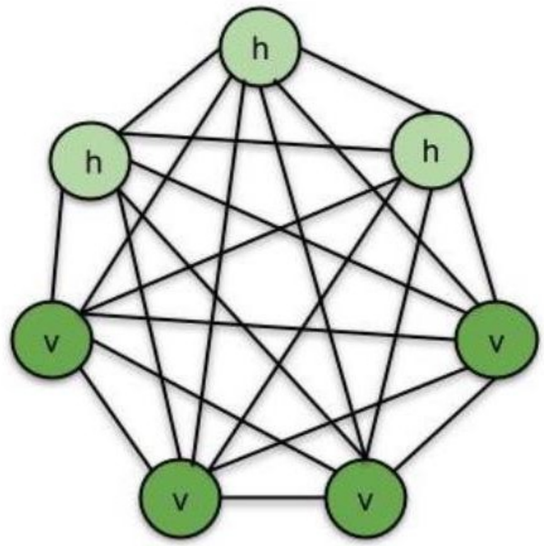
1. Restricted Boltzmann Machines (RBMs)

   RBMs are a simplified version of Boltzmann Machines with a two-layer architecture: a visible layer and a hidden layer, with no connections between neurons in the same layer. They are primarily used for feature learning, dimensionality reduction, and collaborative filtering.

2. Deep Belief Networks (DBNs)
   DBNs are a stack of multiple RBMs where each RBM is trained layer by layer in an unsupervised manner. After pre-training, the network can be fine-tuned using supervised learning. DBNs are widely used in speech recognition, image classification, and generative modeling.

3. Deep Boltzmann Machines (DBMs)
   DBMs are similar to DBNs but consist of multiple layers of hidden units that are interconnected. Unlike DBNs, where training is done sequentially, DBMs learn all layers simultaneously, making them more powerful but computationally expensive. They are useful in unsupervised learning and feature representation.

There are two types of nodes in the Boltzmann Machine
Visible nodes: those nodes which we can represent and do measure
Hidden nodes: those nodes which we cannot represent or do not measure.
Irrespective of these different types, Boltzmann machine consider them same as machine work on single system.



Best example we can give is recommendations of YouTube videos.
Considering architecture of ANN, we are aware about input layer where input nodes are known.
Here Visible nodes are nodes can be considered as recommendation seen on screen.
For example; first video is related to sports, second can be news, third can be a video of song, a next can be any interview (these can be consider as Visible nodes).
Here in this scenario, we keep scrolling until we find something interesting.
Once we get it play, let's say video of song from your favorite artist. Then now onwards, machine will recommend you other videos of same artist. In this situation video from your favorite can be considered as hidden node.

Boltzmann learning

Boltzmann learning is similar to error-correction learning and is used during supervised training.
In this algorithm, the state of each individual neuron, in addition to the system output, are taken into account. In this respect, the Boltzmann learning rule is significantly slower than the error-correction learning rule.

Note: Neural networks that use Boltzmann learning are called Boltzmann machines.

Boltzmann learning is similar to an error-correction learning rule, in that an error signal is used to train the system in each iteration. However, instead of a direct difference between the result value and the desired value, we take the difference between the probability distributions of the system.

The Boltzmann learning rule updates the weights by minimizing the difference between observed and expected correlations:

$$\Delta w_{ij} = \eta(\langle v_i v_j \rangle_{data} - \langle v_i v_j \rangle_{model}), \tag{3}$$

where:

- η is the learning rate.
- $\langle v_i v_j \rangle_{data}$ is the average correlation in the presence of training data.
- $\langle v_i v_j \rangle_{model}$ is the correlation in the absence of training data (free-running network).

State transition diagram and false minima problem

What is State Transition?

A **State** can be consider as current condition or situation of system; this can be active, inactive, in progress, waiting, stopped and so many states.

Transition is the process or a period of changing state from one condition to another.

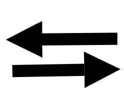
A **State Transition** diagram is a type of diagram used to describe the behavior of systems. State diagrams require that the system described is composed of a finite number of states.

<u>Four Parts of State Transition Diagram</u>

1) **States** that the software might get

1st Try

2) **Transition** from one state to another state



3) **Events** that origin from / to transition

Incorrect Pin

4) **Action** can be result from Transition

Access Granted

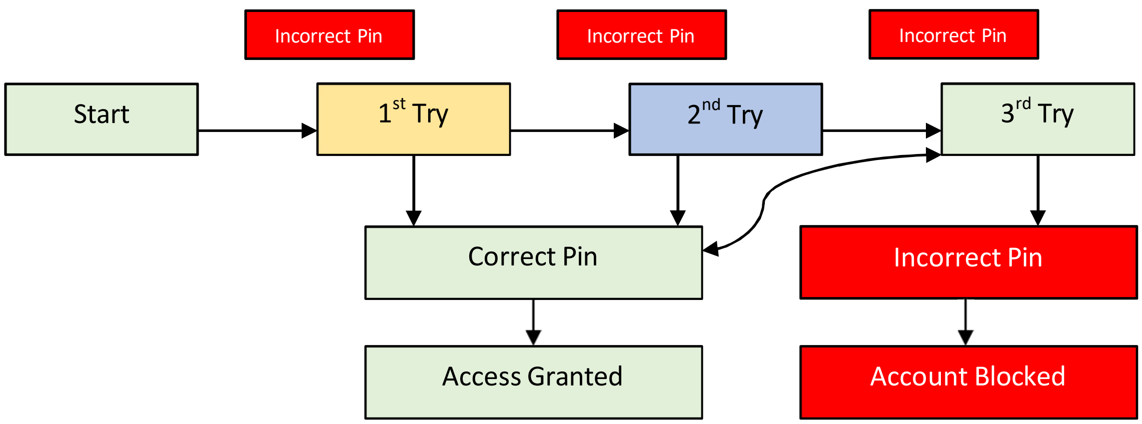
State Transition Table

In state transition diagram the states are shown in boxed texts, and the transition is represented by arrows. It is also called State Chart or Graph.

In state transition table all the states are listed on the left side, and the events are described on the top. Each cell in the table represents the state of the system after the event has occurred. It is also called State Table.

Example of State Transition Diagram

Let's consider an ATM system function where if the user enters the invalid password three times the account will be locked.



In this system, if the user enters a valid password in any of the first three attempts the user will be logged in successfully. If the user enters the invalid password in the first or second try, the user will be asked to re-enter the password. And finally, if the user enters incorrect password 3rd time, the account will be blocked.

In the diagram whenever the user enters the correct PIN he is moved to Access granted state, and if he enters the wrong password he is moved to next try and if he does the same for the 3rd time the account blocked state is reached.

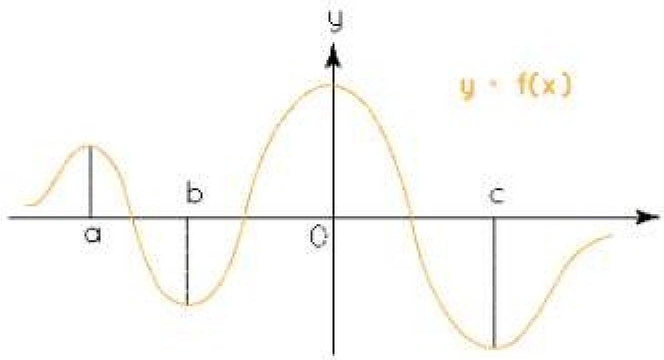
State Transition Table

States	Correct PIN	Incorrect PIN
S1) Start	S5	S2
S2) 1st attempt	S5	S3
S3) 2nd attempt	S5	S4
S4) 3rd attempt	S5	S6
S5) Access Granted	–	–
S6) Account blocked	–	–

False Minima problem

Minima is the minimum value of function within the given set of ranges.

Note: Maxima and minima are the peaks and valleys in the curve of a function respectively. Maxima will be the highest point on the curve within the given range and minima would be the lowest point on the curve.



In the image given below, we can see various peaks and valleys in the graph.
At x = a and at x = 0, we get maximum values of the function,
and at x = b and x = c, we get minimum values of the function.
All the peaks are the maxima and the valleys are the minima.

What is False Minima Problem? How to address it in ANN.

False minima, also known as local minima, are a common problem that can occur in optimization algorithms used in artificial intelligence. In optimization, the goal is to find the minimum or maximum value of a function. However, some functions can have multiple local minima, which are points where the function has a lower value than all of its neighboring points, but are not the absolute minimum of the function.

The false minima problem occurs when an optimization algorithm gets stuck in one of these local minima instead of finding the global minimum. This can happen when the algorithm follows a path that leads to a local minimum, but is unable to escape and find the global minimum. This can result in suboptimal solutions and reduce the effectiveness of the algorithm.
There are several techniques used in artificial intelligence to address the false minima problem, including

1. **Initialization**: The initial values of the parameters can have a significant impact on the optimization process. By selecting initial values that are more likely to lead to the global minimum, the chances of getting stuck in a local minimum can be reduced.
2. **Exploration vs. exploitation**: Some optimization algorithms balance exploration, which involves searching for new solutions, and exploitation, which involves refining known solutions. By balancing exploration and exploitation, the algorithm can be more effective at finding the global minimum.
3. **Stochastic optimization**: Stochastic optimization involves introducing randomness into the optimization process. By adding randomness, the algorithm can explore different regions of the search space and reduce the chances of getting stuck in a local minimum.

Stochastic Update

Stochastic refers to a variable process where the outcome involves some randomness and has some uncertainty.

Stochastic update is a technique used in machine learning and artificial intelligence to update the parameters of a model based on a random subset of the training data, rather than the entire dataset. Stochastic update is a form of stochastic gradient descent, which is a widely used optimization technique in machine learning.

In stochastic update, the model parameters are updated based on the error or loss computed on a randomly selected subset of the training data, also known as a mini-batch. The mini-batch size is typically much smaller than the full training set, and the random selection of data points helps to introduce randomness and prevent the optimization process from getting stuck in local minima.

Stochastic update has several advantages over batch update, which involves updating the model parameters based on the full training set. These advantages include:

1. Faster convergence: Stochastic update can converge faster than batch update, especially when the dataset is large, as the model parameters are updated more frequently.
2. More robustness: Stochastic update can be more robust to noisy or irrelevant data points, as they are less likely to have a significant impact on the model parameters.
3. Reduced memory requirements: Stochastic update requires less memory than batch update, as it only needs to store the mini-batch instead of the entire training set.

However, stochastic update can also have some disadvantages, including:

1. Higher variance: Stochastic update can introduce higher variance in the optimization process, as the model parameters are updated based on a random subset of the training data.
2. Slower convergence at the end: At the end of the training process, stochastic update can converge slower than batch update, as the updates become less frequent and more noisy.

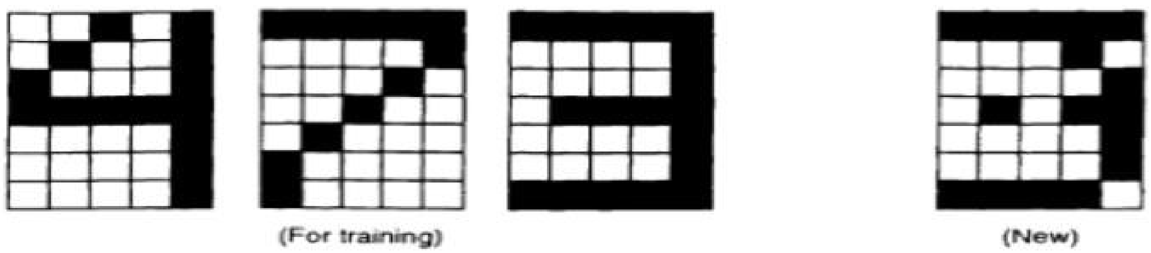
Popular examples of stochastic optimization algorithms are: Simulated Annealing, Genetic Algorithm, Particle Swarm Optimization

Basic functional units of ANN for pattern recognition tasks:

Pattern association

Pattern association is the process of memorizing input-output patterns in hetero associative network architecture, or input patterns only in an auto associative network, in order to recall the patterns when a new input pattern is presented. It is not required that the new input pattern be exactly the same as one that is memorized. It can be different, but similar to it. Three exemplar patterns are shown in figure.

After memorizing them in a system, a new pattern is presented, a corrupted variant of the pattern 3. An associative memory system should associate this pattern with one of the memorized patterns. This is a task for auto associative network architecture.



Pattern classification and pattern mapping tasks

Before searching for a pattern there are some certain steps and the first one is to collect the data from the real world.
The collected data needs to be filtered and pre-processed so that system can extract the features from the data. This data can be divided into general category like audio, number, image, text which can be in the form of traffic signal images, human faces, handwritten letters, and the DNA sequences. These need to be converted to machine understandable language like 0 and 1.

In classification, the algorithm assigns labels to data based on the predefined features. This is an example of supervised learning.

What is Pattern Classification?

Pattern classification is a fundamental problem in artificial intelligence that involves assigning objects or data points to predefined categories or classes. The goal of pattern classification is to develop an algorithm or model that can accurately predict the class of a new data point based on its features or attributes.

In pattern classification, the algorithm or model is typically trained on a labeled dataset, where each data point is associated with a known class label. The model learns to recognize patterns in the data that are characteristic of each class and uses these patterns to classify new, unseen data points.
There are several techniques used in artificial intelligence for pattern classification, including:

1. Supervised learning: This involves training a model on labeled data, where the class labels are known. The model learns to recognize patterns in the data that are characteristic of each class and uses these patterns to classify new, unseen data points.

2. Unsupervised learning: This involves training a model on unlabeled data, where the class labels are not known. The model learns to identify patterns in the data without any prior knowledge of the class labels.

3. Semi-supervised learning: This involves training a model on a combination of labeled and unlabeled data. The model learns to identify patterns in the labeled data and uses this knowledge to classify the unlabeled data.

Pattern mapping tasks

Pattern mapping is a common task in artificial intelligence (AI) and involves identifying and mapping patterns within data. There are several techniques used in AI to perform pattern mapping tasks, including:

1. Machine learning: This involves training an AI model on a large dataset and then using the model to identify patterns in new data.

2. Deep learning: This is a subset of machine learning that uses neural networks to identify patterns in data. Deep learning is particularly useful for tasks that involve image recognition and natural language processing.

3. Clustering: This involves grouping data points based on their similarities. Clustering algorithms can be used to identify patterns in large datasets.

4. Association rule mining: This involves identifying relationships between items in a dataset. Association rule mining is commonly used in market basket analysis to identify products that are frequently purchased together.

5. Decision trees: This involves breaking down a dataset into smaller subsets based on specific criteria. Decision trees are commonly used in classification tasks to identify patterns that distinguish between different classes of data.

Overall, pattern mapping tasks in AI involve using algorithms and techniques to identify and extract meaningful patterns from large datasets. These patterns can then be used to make predictions or inform decision-making processes in various domains.