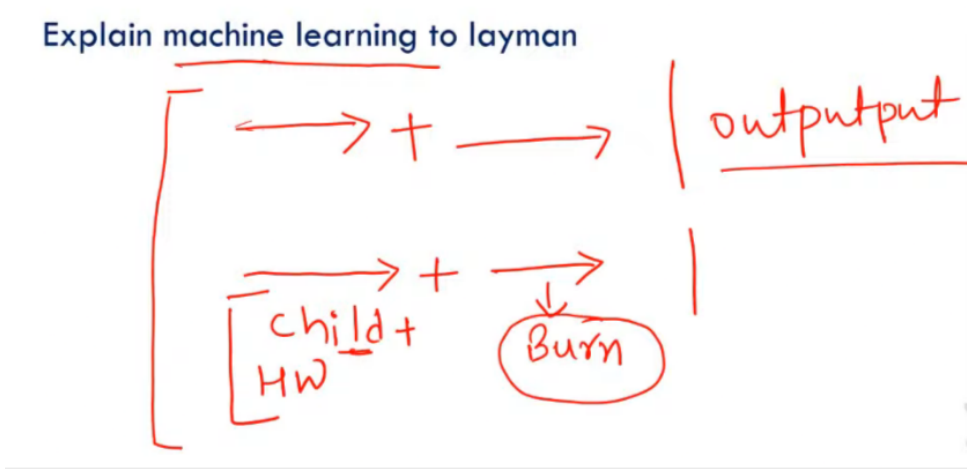
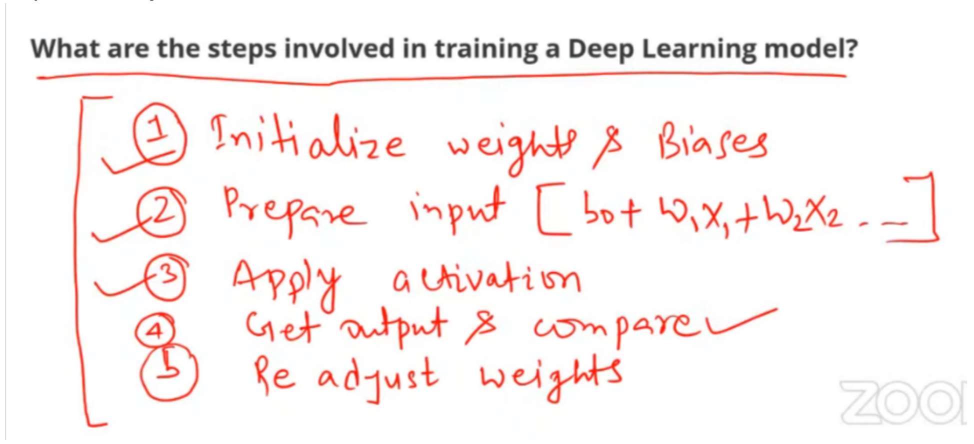
Session 13

Wednesday, March 29, 2023 5:59 PM


Explain machine learning to layman

Machine learning is a type of artificial intelligence that allows computer systems to learn and improve from experience without being explicitly programmed. In simple terms, it is a way for computers to automatically improve their performance at a task by learning from data instead of having to be programmed every step of the way.
Just like how humans learn from their experiences and improve their decision-making skills, machine learning algorithms can also learn from data and improve their accuracy and efficiency over time. This makes them very useful in a variety of applications such as image recognition, speech recognition, natural language processing, and predictive modeling.
For example, imagine you want to teach a computer how to recognize cats in pictures. Instead of telling the computer exactly what a cat looks like, you would give it a large dataset of pictures that have been labeled as either containing a cat or not. The machine learning algorithm would then analyze these images and look for patterns and features that are common to all the pictures that have cats in them. Over time, the algorithm would get better and better at recognizing cats in new pictures, and you wouldn't need to tell it explicitly what to look for.
In summary, machine learning is a powerful tool that allows computers to learn and improve from experience, just like humans do.


What are the steps involved in training a Deep Learning model?

1. Initialize weights & Biases
2. Prepare input [$b_0 + w_1x_1 + w_2x_2$ ...]
3. Apply activation
4. Get output & compare
5. Re adjust weights

Training a deep learning model typically involves the following steps:

1. **Defining the problem and preparing the data:** The first step is to define the problem you want to solve and gather data relevant to that problem. The data must be cleaned, preprocessed, and transformed into a format suitable for training the model.
2. **Building the model architecture:** The next step is to design the architecture of the neural network. This involves selecting the appropriate layers and parameters for the model based on the specific problem you are trying to solve.
3. **Compiling the model:** After the architecture is defined, the next step is to compile the model. This involves specifying the optimization algorithm to use, the loss function to optimize, and the metrics to track during training.
4. **Training the model:** This is the step where the model is actually trained on the prepared data. The training process involves feeding the data into the model, allowing it to make predictions,

and adjusting the parameters of the model to minimize the loss function.

5. **Evaluating the model:** After the model has been trained, it needs to be evaluated to determine how well it performs on new data. This involves using a separate set of data to test the model's accuracy and other metrics.
6. **Tuning the model:** If the model does not perform well, the next step is to adjust the model parameters or architecture and repeat the training process until the desired performance is achieved.
7. **Using the model for predictions:** Once the model has been trained and evaluated, it can be used to make predictions on new data that it has not seen before.

Overall, training a deep learning model is an iterative process that involves defining the problem, preparing the data, designing the model architecture, training the model, evaluating its performance, and making adjustments as necessary.

What is the use of the swish function? MCQ

Activation function

Relu
Sigmoid
tanh

Swish is an activation function for neural networks that was proposed by researchers at Google. It is similar to the widely used ReLU (Rectified Linear Unit) function but has been shown to improve the performance of certain types of neural networks.
The Swish function is defined as f(x) = x / (1 + exp(-x)). It is a smooth, non-monotonic function that has a range between 0 and 1. Like ReLU, it is computationally efficient and easy to implement, but unlike ReLU, it has a non-zero gradient for all values of x, which can help mitigate some of the issues with dead neurons that can occur with ReLU.
The main advantage of using the Swish function over other activation functions is that it has been shown to improve the accuracy and convergence speed of certain types of neural networks, particularly in tasks involving image recognition and natural language processing. Some researchers have also reported that it can lead to better generalization performance compared to ReLU or other activation functions.
However, it is important to note that the Swish function may not always perform better than other activation functions, and its effectiveness may depend on the specific architecture and dataset being used. As with any neural network design decision, the choice of activation function should be carefully considered and tested empirically.

A. Given an array containing a range of numbers from 0 to n with a missing number, find the missing number in the input array.

i/p → [13, 14, 15, 17, 18, 19, 21, 22]

o/p → [16, 20] FAANG

```
def missing(num):
return [x for x in range(num[0], num[-1]+1) if x not in num]
print(missing([1,2,3,4,5,6,7,9,11]))
output - [8,10]
```

What are variants of gradient descent and how it works?

→ N → one shot ✓

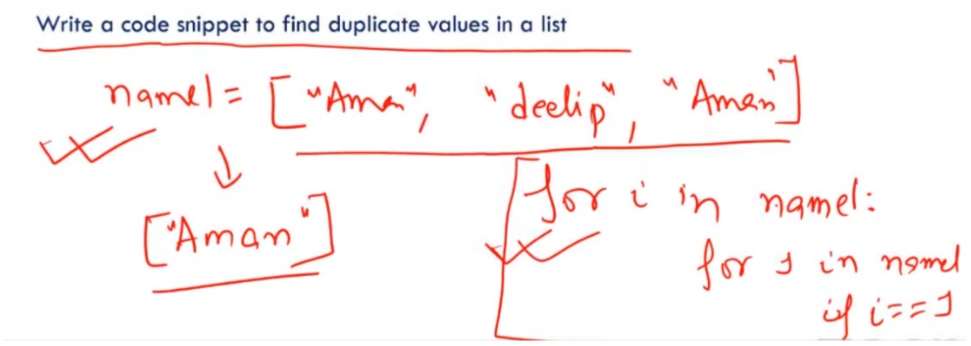
100→10→ MB → Batch wise ✓

→ SC → One example? ✓

Gradient descent is an optimization algorithm that is commonly used in machine learning and deep learning. It is used to find the minimum of a function by iteratively adjusting the parameters until a minimum is found. There are several variants of gradient descent, including:

8. Batch Gradient Descent: This variant calculates the gradients of the cost function with respect to all the training examples, and updates the parameters after iterating over all the examples. This can be computationally expensive for large datasets.
9. Stochastic Gradient Descent (SGD): This variant randomly selects one training example at a time and calculates the gradient of the cost function with respect to that example. The parameters are then updated based on this gradient. SGD is faster than batch gradient descent but can be noisy because it does not use all the training examples to update the parameters.
10. Mini-batch Gradient Descent: This variant is a compromise between batch GD and SGD. It uses a small batch of training examples to compute the gradient.
11. Momentum-based Gradient Descent: This variant takes into account the past gradients to calculate the current gradient. It adds a fraction of the previous gradient to the current gradient which helps it converge faster.
12. Adaptive Learning Rate Methods: These variants adjust the learning rate during the training process. One popular method is AdaGrad, which uses a different learning rate for each parameter based on its previous gradients.
13. Rprop: This variant uses only the sign of the gradient and adjusts the learning rate differently for positive and negative gradients.
14. Adagrad: It adapts the learning rate to the parameters, performing smaller updates (i.e., low learning rates) for parameters associated with frequently occurring features, and larger updates (i.e., high learning rates) for parameters associated with infrequent features.

All these variants have their strengths and weaknesses depending on the specific application and dataset. The choice of which variant to use depends on factors such as the size of the dataset, the number of parameters in the model, and the computational resources available.


Write a code snippet to find duplicate values in a list

name1 = ["Aman", "deelip", "Aman"]

["Aman"]

for i in name1:
    for j in name1
    if i==j

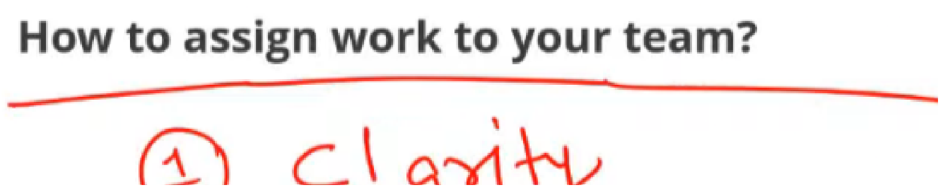
Sure, here's a Python code snippet to find duplicate values in a list:

```
Copy Code
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3]
duplicate_values = []

for i in my_list:
    if my_list.count(i) > 1:
        if i not in duplicate_values:
            duplicate_values.append(i)

print("Duplicate Values:", duplicate_values)
```

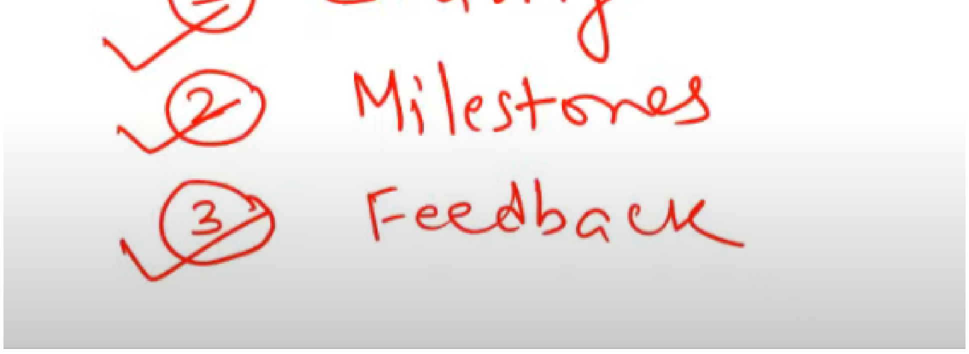
In the above code, we first define our list `my_list` which contains some duplicate values. We create an empty list called `duplicate_values` to store any duplicates that we find.

We then loop through each element in `my_list`, and use the `count()` method to check how many times that element appears in the list. If it appears more than once, we add it to our `duplicate_values` list (if it hasn't already been added).

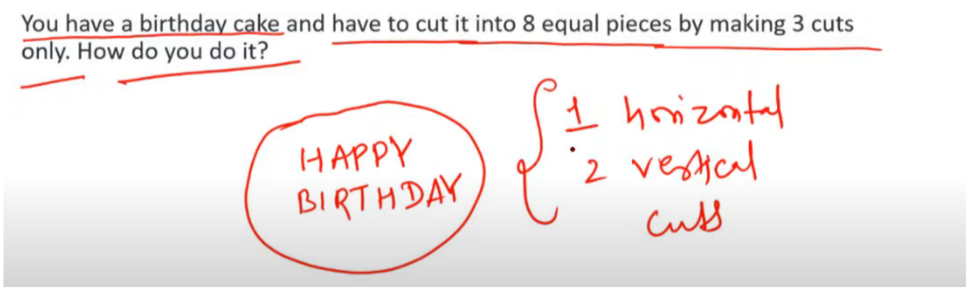
Finally, we print out the `duplicate_values` list to show which values were duplicated in the original list.


How to assign work to your team?

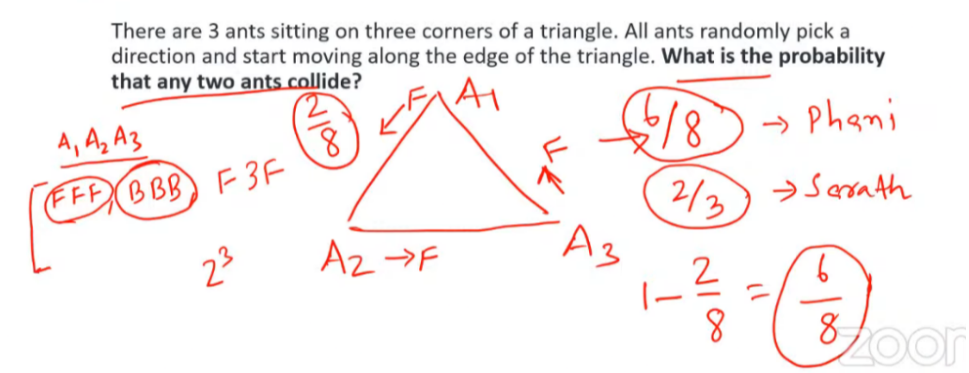
① Clarity



Assigning work to your team can be a complex process that requires careful planning and communication. Here are some steps you can follow to effectively assign work to your team:
Define the project or task: Clearly define the project or task you want your team to work on. This includes defining goals, objectives, timelines, and resources required.
Identify the right team members: Consider the skills and strengths of each team member and identify who is best suited to take on specific tasks. This will allow you to maximize efficiency and ensure that the work is completed to a high standard.
Set expectations: Clearly communicate what you expect from each team member in terms of deliverables, quality standards, deadlines, and expectations. Make sure everyone understands their role in the project and the importance of their contribution.
Provide support: Ensure that your team has access to the necessary resources, tools, and information needed to complete their tasks effectively. Encourage questions and feedback to help your team navigate any challenges that arise.
Monitor progress: Regularly check in with your team to monitor progress, provide guidance, and offer assistance where needed. This helps to ensure that the work stays on track and any issues are addressed promptly.
Recognize achievements: Acknowledge and celebrate your team's achievements along the way. This helps to build morale and motivation, and encourages your team to continue working towards success.
By following these steps, you can effectively assign work to your team, optimize performance, and achieve successful outcomes.


You have a birthday cake and have to cut it into 8 equal pieces by making 3 cuts only. How do you do it?

To cut a cake into 8 equal pieces using only 3 cuts, you can follow these steps:
First, make two cuts in the shape of a cross, dividing the cake into four equal pieces.
Next, make a third cut that is horizontal and passes through both the vertical and horizontal cuts you made earlier. This will divide each of the four equal pieces in half, resulting in 8 equal pieces.
You can visualize this by imagining a cake sitting on a plate, with a knife held vertically to cut the cake in half from top to bottom (creating two halves), then horizontally to create two more halves perpendicular to the first cut (creating four equal quarters), and finally cutting horizontally again so that the third cut goes through each quarter to create eight equal pieces.


There are 3 ants sitting on three corners of a triangle. All ants randomly pick a direction and start moving along the edge of the triangle. **What is the probability that any two ants collide?**

To see why, let's consider the possible movements of the ants along the edges of the triangle. Each ant has two choices of direction, and there are three ants. Therefore, there are a total of 2^3 = 8 possible ways in which the ants can move.

Out of these eight possibilities, there are four ways in which any two ants collide. The four cases where two ants collide are:

- Ant 1 and ant 2 collide while ant 3 moves in a different direction.
- Ant 1 and ant 3 collide while ant 2 moves in a different direction.
- Ant 2 and ant 3 collide while ant 1 moves in a different direction.
- Ant 1 and ant 2 collide, and then either ant 1 or ant 2 collides with ant 3.

Therefore, the probability that any two ants collide (and the other ant moves in a different direction) is 4/8, which simplifies to 1/2. And since any two ants can collide in three different pairs (i.e., ants 1 and 2, ants 1 and 3, or ants 2 and 3), we need to multiply 1/2 by 3 to get the overall probability of any two ants colliding, which gives us 3/6 or 1/2 again.

However, we also need to account for the case where all three ants move in the same direction, as this would result in no collisions. There is only one way for this to happen out of the eight possible movements, so the probability of no collisions is 1/8.

Therefore, the probability of any two ants colliding (and the other ant moving in a different direction) is:

(4/8) x 3 = 12/8

And the probability of no collisions is:

1/8

So, the total probability that any two ants collide is:

12/8 + 1/8 = 13/8

However, since a probability cannot exceed 1, we know that the correct answer must be 1. Therefore, the probability of any two ants colliding (and the other ant moving in a different direction) is actually 1/4 or 0.25.