**IT19980782**

**question 001**

1D convolution is a type of convolution that is used for processing one-dimensional data, such as time-series data.In 1D convolution, a small set of learnable filters are applied to a one-dimensional input sequence to produce a set of output features. The filters are typically small in size and are moved over the input sequence, one element at a time. At each position, the filter is multiplied with the corresponding elements of the input sequence, and the result is summed to produce a single output value. This process is repeated for each position in the input sequence, producing a set of output features.

In the context of deep learning, convolution is used to extract features from input data. In the case of images, convolution can be used to extract features such as edges, corners, and textures. In the case of time-series data, convolution can be used to identify patterns and trends.

**question 003**

The purpose of a validation set is to give us an idea of how our model behaves on data on which it has not been trained. Therefore, the epoch when the validation error starts to increase is precisely when the model is overfitting to the training set and does not generalize new data correctly.

(The model trains for too long on a single sample set of data.)

Overfitting examples

Consider a use case where a machine learning model has to analyze photos and identify the ones that contain dogs in them. If the machine learning model was trained on a data set that contained majority photos showing dogs outside in parks , it may may learn to use grass as a feature forclassification, and may not recognize a dog inside a room.Another overfitting example is a machine learning algorithm that predicts a university student's academic performance and graduation outcome by analyzing several factors like family income, past academic performance, and academic qualifications of parents. However, the test data only includes candidates from a specific gender or ethnic group. In this case, overfitting causes the algorithm's prediction accuracy to drop for candidates with gender or ethnicity outside of the test dataset.

**Ways to reduce overfitting?**

1.Early stopping

Early stopping pauses the training phase before the machine learning model learns the noise in the data. However, getting the timing right is important; else the model will still not give accurate results.

2.Pruning

You might identify several features or parameters that impact the final prediction when you build a model. Feature selection—or pruning—identifies the most important features within the training set and eliminates irrelevant ones. For example, to predict if an image is an animal or human, you can look at various input parameters like face shape, ear position, body structure, etc. You may prioritize face shape and ignore the shape of the eyes.

3.Regularization

Regularization is a collection of training/optimization techniques that seek to reduce overfitting. These methods try to eliminate those factors that do not impact the prediction outcomes by grading features based on importance. For example, mathematical calculations apply apenalty value to features with minimal impact. Consider a statistical model attempting to predict the housing prices of a city in 20 years.

Regularization would give a lower penalty value to features like population growth and average annual income but a higher penalty value to the average annual temperature of the city.

4.Ensembling

Ensembling combines predictions from several separate machine learning algorithms. Some models are called weak learners because their results are often inaccurate. Ensemble methods combine all the weak learners to get more accurate results. They use multiple models to analyze sample data and pick the most accurate outcomes. The two main ensemble methods are bagging and boosting. Boosting trains different machine learning models one after another to get the final result, while bagging trains them in parallel.

5.Data augmentation

Data augmentation is a machine learning technique that changes the sample data slightly every time the model processes it. You can do this by changing the input data in small ways. When done in moderation, data augmentation makes the training sets appear unique to the model and prevents the model from learning their characteristics. For example, applying transformations such as translation, flipping, and rotation to input images.

**how can the mini batch Stochastic Gradient Descent converge faster than the batch gradient descent?**

Batch gradient descent takes longer to converge since it computes the gradient using the entire training dataset in each iteration. Stochastic gradient descent, on the other hand, can converge faster since it updates the model parameters after processing each example, which can lead to faster convergence.

In Batch Gradient Descent, all the training data is taken into consideration to take a single step. We take the average of the gradients of all the training examples and then use that mean gradient to update our parameters. So that’s just one step of gradient descent in one epoch.

In Batch Gradient Descent we were considering all the examples for every step of Gradient Descent. But what if our dataset is very huge. Deep learning models crave for data. The more the data the more chances of a model to be good. Suppose our dataset has 5 million examples, then just to take one step the model will have to calculate the gradients of all the 5 million examples. This does not seem an efficient way. To tackle this problem we have Stochastic Gradient Descent. In Stochastic Gradient Descent (SGD), we consider just one example at a time to take a single step.

In batch gradient descent,to take a single step all datasets examples are considered.In SGD, only one example is taken at a time to take one step.Hence SGD is faster.