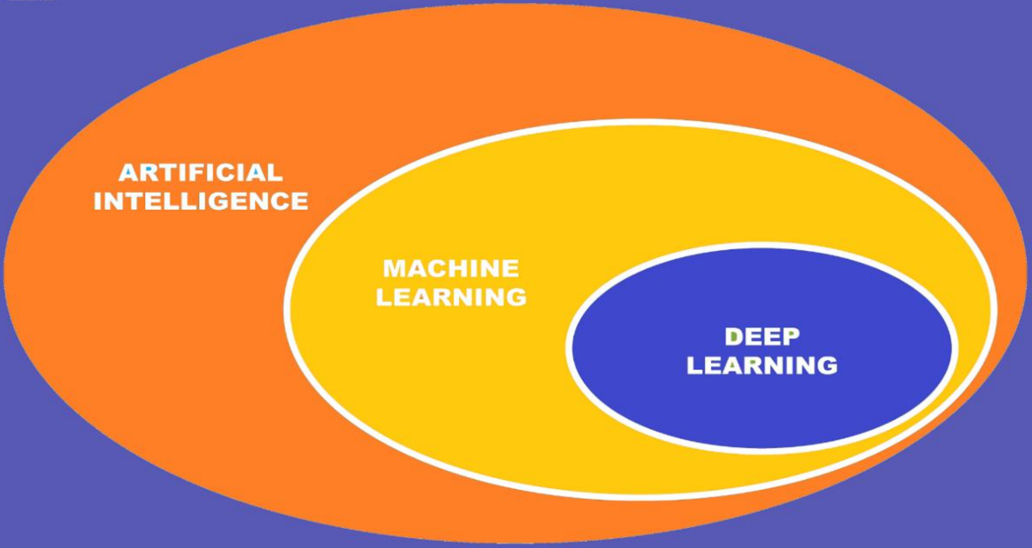
Objective Type Questions and Answers in Deep Learning

Deep Learning



NARESH KUMAR

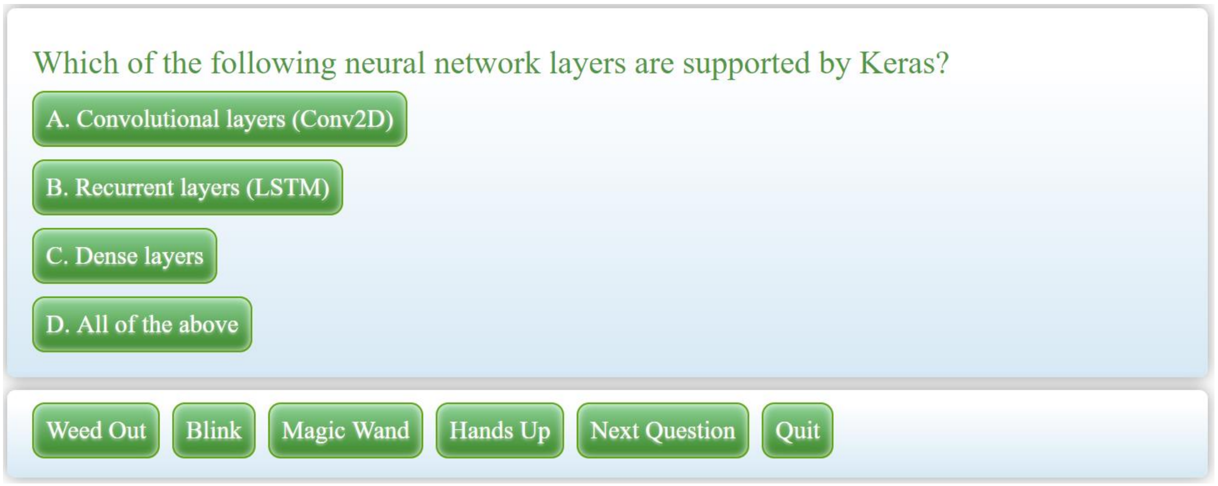
A NOTE TO READERS

This book demonstrates very basic concepts of deep learning. It contains **20 chapters** and **205 objective type questions**. Each chapter contains a short description of a concept and objective type questions from that concept.

This book is just a short summary of my online contents on:

- The Professionals Point (http://theprofessionalspoint.blogspot.com/)
- Online ML Quiz (http://onlinemlquiz.com/)

Contents of this book are available on my blog (The Professionals Point) and objective type questions are available in the form of quiz on my website (Online ML Quiz).



Which of the following neural network layers are supported by Keras?

A. Convolutional layers (Conv2D)

B. Recurrent layers (LSTM)

C. Dense layers

D. All of the above

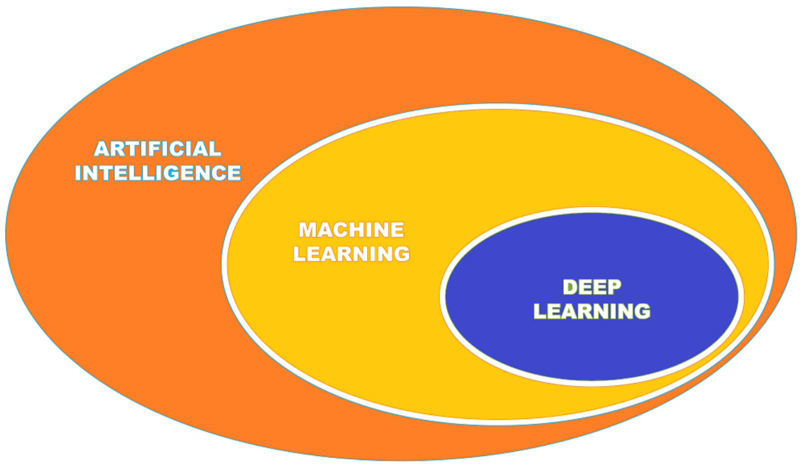
Assumption: This should not be your first book on deep learning as I have not covered deep learning concepts in detail, just given a short description to revise your concepts. So, I assume, you have some basic understanding of deep learning concepts before reading this book.

Continuously upgrading this book: I am continuously adding more and more objective type questions in this book and in my online deep learning quiz.

TABLE OF CONTENTS

CHAPTER 1

MACHINE LEARNING vs DEEP LEARNING



Deep Learning is considered as a subset of Machine Learning. Both have a lot of similarities and differences. Let's see some of the differences between them.

1. Scale of data: Deep Learning algorithms work efficiently on high amount of data (both structured and unstructured). If there is less amount of data, deep learning algorithms may not perform well as compared to machine learning algorithms. Deep learning algorithms are best suited for unstructured data like images, videos, voice, natural language processing etc. Machine Learning algorithms are not capable of dealing with unstructured data.

2. Scale of computation: Deep Learning algorithms require high computational power. Deep Learning algorithms need high-end machines like GPUs as these heavily perform complex operations like matrix multiplications.

3. Feature Extraction: In machine learning, we need to manually identify the features from the dataset based on our domain knowledge and expertise. This takes a huge amount of time and effort. Also, there are a lot of chances that we can miss some of the important features which are crucial for prediction.

For example, while image processing in machine learning, you need to extract the feature manually in the image like eyes, nose, lips, pixel values, shape, textures, position, orientation and so on. These extracted features are then fed to the machine learning model. The performance of most of the machine learning algorithms depend on how accurately the features are identified and extracted.

Deep learning solves this issue automatically using convoluted layers in CNN. Initial layers of CNN model learn small details from the picture like edges, lines etc. and then deeper layers combine the previous knowledge to make a more complex information. We will discuss CNN in more detail later in this book.

4. Training Data: Deep Learning algorithms usually require more training data as compared to machine learning algorithms.

5. Data Augmentation: Creating new data by making reasonable modifications to the existing data is called data augmentation. Let's take an example of a well-known MNIST dataset (hand written digits). We can easily generate thousands of new similar images by rotating, flipping, scaling, shifting, zooming in and out, cropping, changing or varying the color of the existing images.

We can use data augmentation technique when our model is overfitting due to less data.

In many cases in deep learning, increasing the amount of data is not a difficult task as we discussed above the case of MNIST dataset. In machine learning, this task is not that easy as we need labelled data which is not easily available.

6. Training Time: Deep Learning algorithms usually take a longer time to train as compared to machine learning algorithms as there are a lot of computations involved inside the hidden layers.

7. Testing Time: Deep Learning algorithms take much less testing time as compared to machine learning algorithms.

8. Interpretability: Machine Learning algorithms are more interpretable as compared to deep learning algorithms. Deep learning models mostly act as black box.

For example, decision tree in machine learning can be easily interpreted by human beings and they can easily get to know how the final values are computed. On the other hand, it is very hard to know what calculations happened inside the hidden layers of the neural networks, how convoluted layers in CNN identified the various portions of the images etc.

9. Dimensionality: As the dimension of the data increases, efficiency of machine learning algorithms starts degrading. Although we have some dimensionality reduction techniques in machine learning like PCA, t-SNE, SVD, MDS etc. but deep learning takes care of dimensionality very well.

10. Computer Vision: Deep Learning algorithms help in solving a lot of computer vision problems like:

A) Image Classification

B) Image Classification with Localization

C) Object Detection

D) Object Segmentation

E) Image Style Transfer

F) Image Colorization

G) Image Reconstruction

H) Image Super-Resolution

I) Image Synthesis

Machine Learning algorithms have limited capacity and efficiency in resolving these computer vision problems while convolutional neural networks are very efficient in handling these tasks.

QUESTIONS

1. Which of the following is FALSE about Deep Learning and Machine Learning algorithms?

A. Deep Learning algorithms work efficiently on high amount of data

B. Feature Extraction needs to be done manually in both ML and DL algorithms

C. Deep Learning algorithms are best suited for unstructured data

D. Deep Learning algorithms require high computational power

2. Which of the following is FALSE about Deep Learning and Machine Learning algorithms?

A. Data augmentation can be done easily in Deep Learning as compared to Machine Learning

B. Deep Learning algorithms efficiently solve computer vision problems

C. Deep Learning algorithms are more interpretable as compared to Machine Learning algorithms

D. None of the above

ANSWERS

Answer 1: B

Answer 2: C

CHAPTER 2

PERCEPTRONS AND NEURAL NETWORKS

Perceptron

Perceptron is an artificial neuron and is the fundamental unit of a neural network in deep learning. It is also called single layer neural network or single layer binary linear classifier.

Perceptron takes inputs which can be real or boolean, assigns random weights to the inputs along with a bias, takes their weighted sum, pass it through a threshold function which will decide whether to take any action on it or not depending upon some threshold value, and finally perform linear binary classifications. This threshold function is usually a step function.

Mathematical representation of perceptron looks like an if-else condition. If the weighted sum of the inputs is greater than a threshold value, output will be 1 else output will be 0.

Applications

Perceptrons can be used to solve any problem which contains linearly separable set of inputs. For example, we can implement logic gates like OR and AND because these are linearly separable.

Limitation

Perceptron can only learn linearly separable functions. It cannot handle non-linear inputs. For example, it cannot implement XOR gate as it can't be classified by a linear separator.

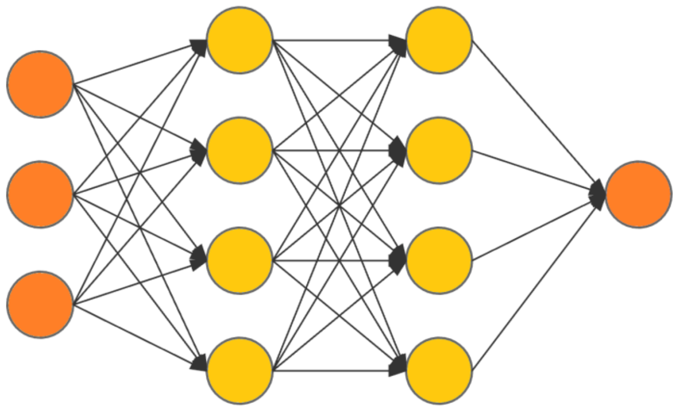
Neural Network

To address above limitation of Perceptrons, we'll need to use a multi-layer perceptron, also known as feed-forward neural network. A neural network is a composition of perceptrons, connected in different ways and operating on different activation functions.

1. All the layers (input layer, hidden layers and output layer) are interconnected.

2. Weight is added to each input and bias is added per neuron, and then it is passed to the activation function which decides whether to activate the neuron or not.

3. Forward propagate the weighted sum, calculate the error, backward propagate and update the weights using gradient descent algorithm. Keep doing the same until a satisfactory result is achieved.



Types of Neural Networks

1. Feed Forward Neural Network: This is the simplest type of neural network. Data flows only in forward direction from input layer to hidden layers to the output layer. It may contain one or more hidden layers. All the nodes are fully connected. Back propagation method is used to train these kind of neural networks. If there are a lot of hidden layers, it may be referred as deep neural network.

2. CNN (Convolutional Neural Network)

3. Capsule Neural Networks

4. RNN (Recurrent Neural Network)

5. LSTM (Long Short-Term Memory Networks)

6. Autoencoders

7. Radial Basis Function Neural Network: RBF neural networks are a type of feed forward neural networks that use radial basis function as activation function instead of logistic function. Instead of just outputting 0 or 1 (as in logistic function), radial basis functions consider the distance of a point with respect to the center.

We will discuss all these neural networks in detail later in this book.

Hidden Layers

The hidden layer is where the network stores its internal abstract representation of the training data, similar to the way that a human brain has an internal representation of the real world.

Feature extraction happens at hidden layers. We can keep increasing the number of hidden layers to obtain higher accuracy. It should also be noted that increasing the number of layers above a certain point may lead the model to overfit.

Comparison of Deep and Shallow Neural Networks

Shallow neural networks have only one hidden layer as opposed to deep neural networks which have several hidden layers.

Advantages of Deep Neural Networks

1. Deep neural networks are better in learning and extracting features at various levels of abstraction as compared to shallow neural networks.

2. Deep neural networks have better generalization capabilities.

Disadvantages of Deep Neural Networks

1. Vanishing Gradients: As we add more and more hidden layers, back-propagation becomes less and less useful in passing information to the lower layers. As information is passed back, the gradients begin to vanish and become small relative to the weights of the networks.

2. Overfitting: As we keep on adding more and more layers to a neural network, chances of overfitting increase. So, we should maintain reasonable number of hidden layers in deep neural networks.

3. Computational Complexity: As we keep on adding more and more layers to a neural network, computational complexity increases. So, again, we should maintain reasonable number of hidden layers in deep neural networks.

Training Neural Networks using Back Propagation

The most common deep learning algorithm for supervised training of the multi-layer perceptrons is known as back-propagation. Following are the basic steps:

1. A training sample is presented and propagated forward through the network.

2. The output error is calculated, typically the mean squared error or root mean square error in case of regression problem.

3. Weights are updated using gradient descent algorithm.

4. Steps 1 to 3 are repeated again and again until a satisfied result or accuracy is obtained.

QUESTIONS

1. Which of the following is TRUE about Perceptrons?

A. It is an artificial neuron and is the fundamental unit of a neural network

B. It is also called single layer neural network or single layer binary linear classifier

C. A neural network is a composition of perceptrons connected in different ways

D. All of the above

2. Which of the following is FALSE about Perceptrons?

A. Perceptron can learn both linearly and non-linearly separable functions

B. It cannot implement XOR gate as it cannot be classified by a linear separator

C. It can easily implement OR and AND gates as these are linearly separable

D. None of the above

3. Which of the following is the structure of the input to an artificial neuron?

A. Weighted sum of inputs + bias

B. Sum of inputs + bias

C. Weighted sum of bias + inputs

D. Sum of bias + inputs

4. Which of the following is FALSE about Neural Networks?

A. During backward propagation, we update the weights using gradient descent algorithm

B. We can use different activation functions in different layers

C. We can use different gradient descent algorithms in different epochs

D. None of the above

5. Which of the following is FALSE about Hidden Layers in Neural Networks?

A. Abstract representation of the training data is stored in the hidden layers

B. Feature extraction happens at the hidden layers

C. Increasing the number of hidden layers always leads to higher accuracy

D. Increasing the number of hidden layers above a certain point may lead to overfitting

6. Which of the following is FALSE about Deep Neural Networks?

A. These are computationally more complex as compared to shallow networks

B. These have less generalization capabilities as compared to shallow networks

C. These may suffer from overfitting problem

D. These may suffer from vanishing gradients problem

7. Which of the following is TRUE about Shallow Neural Networks?

A. These are computationally more complex as compared to deep neural networks

B. These have better generalization capabilities as compared to deep neural networks

C. These are better in learning and extracting features as compared to deep neural networks

D. None of the above

8. Which of the following is a type of neural network?

A. CNN (Convolutional Neural Network)

B. Capsule Neural Networks

C. Autoencoders

D. All of the above

9. Which of the following is a type of neural network?

A. RNN (Recurrent Neural Network)

B. LSTM (Long Short-Term Memory Networks)

C. Restricted Boltzmann Machines (RBM)

D. All of the above

10. Which of the following is FALSE about Radial Basis Function Neural Network?

A. It resembles to RNNs which have feedback loops

B. It uses radial basis function as activation function

C. While outputting, it considers the distance of a point with respect to the center

D. None of the above

ANSWERS

Answer 1: D

Answer 2: A

Answer 3: A

Answer 4: C

Answer 5: C

Answer 6: B

Answer 7: D

Answer 8: D

Answer 9: D

Answer 10: A

CHAPTER 3

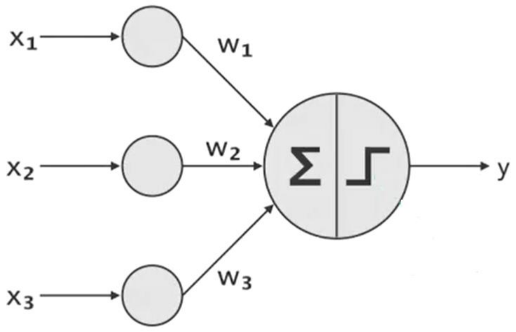
ACTIVATION FUNCTIONS

Activation functions bring non-linearity in neural networks. Commonly used activation functions in neural networks are step, sigmoid, tanh, ReLU and softmax. These are also called squashing functions as these functions squash the output under certain range.

Importance of Activation Functions

These activation functions help in achieving non-linearity in deep learning models. If we don't use these non-linear activation functions, neural network would not be able to solve complex real life problems like image, video, audio, voice and text processing, natural language processing etc. because our neural network would still be linear and linear models cannot solve real life complex problems.

Although linear models are simple but are computationally weak and are not able to handle complex problems. So, if you don't use activation functions, no matter how many hidden layers you use in your neural network, it will still be linear and inefficient.



= Activation Function. It will decide whether to fire this neuron or not.

Step (Threshold) Activation Function

It either outputs 0 or 1 (Yes or No).

In this activation function, there is a sudden change in the decision (from 0 to 1) when input value crosses the threshold value. For most real-world applications, we would expect a smoother decision function which gradually changes from 0 to 1.

As step function either outputs 0 or 1 (Yes or No), it is a non-differentiable activation function and therefore its derivative will always be zero.

Sigmoid Activation Function

It is also called logistic activation function. Its output ranges from 0 to 1. It has "S" shaped curve. Sigmoid function is much smoother than the step function which seems logical and obvious in real life.

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

Advantages

1. Sigmoid is a non-linear activation function.

2. Instead of just outputting 0 and 1, it can output any value between 0 and 1 like 0.62, 0.85, 0.98 etc. So, instead of just Yes or No, it outputs a probability value. So, the output of sigmoid function is smooth, continuous and differentiable.

3. As the range of output remains between 0 and 1, it cannot blow up the activations unlike ReLU activation function.

Disadvantages

1. Vanishing and exploding gradients problem.

2. Computing the exponential may be expensive sometimes.

Tanh (Hyperbolic Tangent) Activation Function

It is similar to Sigmoid Activation Function, the only difference is that it outputs the values in the range of -1 to 1 instead of 0 and 1 (unlike sigmoid function). So, we can say that tanh function is zero centered (unlike sigmoid function) as its values range from -1 to 1 instead of 0 to 1.

Advantages and Disadvantages of Tanh activation function are same as that of sigmoid activation function.

$$\tanh\,(\mathrm{x}) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

ReLU (Rectified Linear Unit)

ReLU outperforms both sigmoid and tanh activation functions and is computationally more efficient compared to both. Given an input value, the ReLU will generate 0, if the input is less than 0, otherwise the output will be the same as the input.

$$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$$

Mathematically, relu(z) = max(0, z)

Advantages

1. It does not require exponent calculation unlike sigmoid and tanh activation functions.

2. It does not encounter vanishing gradient problem.

Disadvantages

1. Unbounded output range: Unbounded output values generated by ReLU could make the computation within the RNN likely to blow up to infinity without reasonable weights. As a result, the learning can be remarkably unstable because a slight shift in the weights in the wrong direction during back-propagation can blow up the activations during the forward pass.

2. **Dying ReLU:** The dying ReLU refers to the problem when ReLU neurons become inactive and only output 0 for any input. So, once a neuron gets negative input, it will always output zero and is unlikely for it to recover. It will become inactive forever. Such neurons will not play any role in discriminating the input and become useless in the neural network. If this process continues, over the time you may end up with a large part of your neural network doing nothing.

Causes of Dying ReLU

1. Too high learning rate

2. Large negative bias

Consider the following statement which is used to calculate the new weights during back-propagation:

*New Weight = Old Weight - (Derivative of Loss Function * Learning Rate) + Bias*

So, if the learning rate is too high, we may end up with a new weight which is negative. Also, if the bias is too negative, we may again end up in negative weight.

Once it becomes negative, ReLU activation function of that neuron will never be activated which will lead that neuron to die forever.

Solution of Dying ReLU

Leaky ReLU is the most common and effective method to alleviate a dying ReLU. It adds a slight slope in the negative range to prevent the dying ReLU issue.

Leaky ReLU has a small slope for negative values, instead of altogether zero. For example, leaky ReLU may have $y = 0.0001x$ when $x < 0$.

Parametric ReLU (PReLU) is a type of leaky ReLU which instead of having a predetermined slope like 0.0001, makes it a parameter for the neural network to figure out itself: $y = \alpha x$ when $x < 0$.

Lower learning rates often mitigates the problem.

Difference between Sigmoid and Softmax

Softmax function can be understood as a generalized version of a sigmoid function or an extension of a sigmoid function. Softmax function is usually used in the output layers of neural networks.

Following are some of the differences between Sigmoid and Softmax function:

1. The sigmoid function is used for the two-class (binary) classification problem, whereas the softmax function is used for the multi-class classification problem.

2. Sum of all softmax units are supposed to be 1. In sigmoid, it’s not really necessary. Sigmoid just makes output between 0 and 1. The softmax enforces that the sum of the probabilities of all the output classes are equal to one, so in order to increase the probability of a particular class, softmax must correspondingly decrease the probability of at least one of the other classes.

When you use a softmax, basically you get a probability of each class (join distribution and a multinomial likelihood) whose sum is bound to be one. In case, you use sigmoid for multi class classification, it'd be like a marginal distribution and a Bernoulli likelihood.

3. Sigmoid is usually used as an activation function in hidden layers (but we use ReLU nowadays) while Softmax is used in output layers.

A general rule of thumb is to use ReLU as an activation function in hidden layers and softmax in output layer in a neural networks.

QUESTIONS

1. Which of the following components make a neural network non-linear in nature?

A. Hidden Layers

B. Activation Functions

C. Weights and Bias

D. Regularization and Dropout

2. What is the purpose of an activation function?

A. To decide whether a neuron will fire or not

B. To increase the depth of a neural network

C. To create connectivity among hidden layers

D. To normalize the inputs

3. Which of the following is FALSE about activation functions?

A. Activation functions help in achieving non-linearity in deep neural networks

B. Activation functions help in reducing overfitting problem

C. These are also called squashing functions as these squash the output under a certain range

D. Commonly used activation functions are step, sigmoid, tanh, ReLU and softmax

4. Which of the following is FALSE about step activation function?

A. It either outputs 0 or 1

B. It is linear in nature

C. It is also called Threshold activation function

D. None of the above

5. Which of the following is FALSE about step activation function?

A. It is a differentiable activation function

B. Derivative of a step function is always zero

C. It is non-linear in nature

D. None of the above

6. Which of the following is FALSE about sigmoid and tanh activation function?

A. Both are non-linear activation functions

B. Output of sigmoid ranges from -1 to 1 while output of tanh ranges from 0 to 1

C. Output of both sigmoid and tanh is smooth, continuous and differentiable

D. None of the above

7. Which of the following is FALSE about sigmoid and tanh activation function?

A. These cannot blow up the activations unlike ReLU

B. Both functions output a probability value instead of discrete values like 0 and 1

C. Sigmoid is zero centered as its values range from -1 to 1

D. None of the above

8. Which of the following is FALSE about sigmoid and tanh activation function?

A. These do not suffer from vanishing and exploding gradient problems unlike ReLU

B. These involve computing the exponential (which may be expensive) unlike ReLU

C. These are non-linear in nature like ReLU

D. None of the above

9. Output of step (threshold) activation function ranges from:

A. Either 0 or 1

B. 0 to 1

C. -1 to 1

D. Either -1 or 1

10. Output of sigmoid activation function ranges from:

A. 0 to 1

B. -1 to 1

C. -1 to 0

D. 0 to 9

11. Output of tanh activation function ranges from:

A. 0 to 1

B. -1 to 1

C. -1 to 0

D. 0 to 9

12. Output of which of the following activation functions is zero centered?

A. Hyperbolic Tangent

B. Sigmoid

C. Softmax

D. ReLU

13. ReLU activation function outputs zero when:

A. Input is zero

B. Input is less than or equal to zero

C. Input is greater than or equal to zero

D. Input is zero or one

14. ReLU outputs same value as input when:

A. Input is greater than zero

B. Input is greater than or equal to zero

C. Input lies between zero and one

D. Input is zero or one

15. Which of the following is FALSE about ReLU activation function?

A. It does not require exponent calculation unlike sigmoid and tanh

B. It outputs 0, if the input is less than or equal to 0, otherwise output will be same as the input

C. ReLU is preferred over sigmoid and tanh for output layer in a neural network

D. None of the above

16. Which of the following is FALSE about Dying ReLU?

A. In this situation, neurons become inactive and only output 0 for any input

B. Dying ReLU neurons do not play any role in discriminating the input and become useless

C. Dying ReLU problem occurs when learning rate is too low

D. None of the above

17. Dying ReLU issue occurs when

A. Learning rate is too high

B. There is a large negative bias

C. Any of the above occurs

D. None of the above

18. Which of the following is NOT a solution for Dying ReLU?

A. Batch normalization

B. Dropout

C. Leaking ReLU

D. Low learning rate

19. Which of the following is a solution for Dying ReLU?

A. Leaking ReLU

B. Max ReLU

C. Parametric ReLU (PReLU)

D. All of the above

20. Which of the following is TRUE about Leaking ReLU?

A. Leaky ReLU is the most common method to alleviate a dying ReLU

B. Leaking ReLU adds a slight slope in the negative range to prevent dying ReLU issue

C. Parametric ReLU (PReLU) is a type of leaky ReLU that uses parameterized slope value

D. All of the above

21. Which of the following is FALSE about Softmax function?

A. Softmax function is a generalized version of a sigmoid function

B. Softmax function is usually used in the hidden layers of neural networks

C. Softmax function is usually used for multi-class classification problem

D. None of the above

22. Which of the following is TRUE about Softmax and Sigmoid function?

A. Sum of probabilities of all softmax units are supposed to be 1

B. Sum of probabilities of all sigmoid units are supposed to be 1

C. Sum of probabilities of both sigmoid and softmax units are supposed to be 1

D. All of the above

23. Which of the following is TRUE about Softmax and Sigmoid function?

A. Softmax is usually used for hidden layers and sigmoid for outer layers

B. Sigmoid is usually used for hidden layers and softmax for outer layers

C. Softmax function is usually used for binary classification problem

D. All of the above

ANSWERS

Answer 1: B

Answer 2: A

Answer 3: B

Answer 4: B

Answer 5: A

Answer 6: B

Answer 7: C

Answer 8: A

Answer 9: A

Answer 10: A

Answer 11: B

Answer 12: A

Answer 13: B

Answer 14: A

Answer 15: C

Answer 16: C

Answer 17: C

Answer 18: B

Answer 19: D

Answer 20: D

Answer 21: B

Answer 22: A

Answer 23: B

CHAPTER 4

WEIGHT AND BIAS

Weight initialization is the most important step while training a neural network. If weights are high, it may lead to exploding gradient. If weights are low, it may lead to vanishing gradient. Due to these issues, our model may take a longer time to converge to global minima or sometimes it may never converge at all. So, weight initialization should be done with immense care.

Normally, weights are randomly initialized at the beginning. We use Gaussian distribution to randomly distribute these weights such that the mean of the distribution is zero and standard deviation is one. But the problem with this approach is that the variance or standard deviation tend to change in next layers which lead to explode or vanish the gradients.

Xavier Weight Initialization Technique

With each passing layer, we want the variance or standard deviation to remain the same. This helps us keep the signal from exploding to a high value or vanishing to zero. In other words, we need to initialize the weights in such a way that the variance remains the same with each passing layer. This initialization process is known as Xavier initialization.

In Xavier initialization technique, we need to pick the weights from a Gaussian distribution with zero mean and a variance of 1/N (instead of 1), where N specifies the number of input neurons.

Basic points about Xavier technique

1. Initially, it was suggested to take variance of 1/(Nin + Nout) instead of 1/N. Nin is the number of weights coming into the neuron and Nout is the number of weights going out of the neuron. But it was computationally complex, so it was discarded and we take only 1/N as variance.

2. In Keras, Xavier technique is used by default to initialize the weights in the neural network.

For ReLU activation function

If we are using ReLU as activation function in hidden layers, we need to go through following steps to implement Xavier initialization technique:

1. Generate random weights from a Gaussian distribution having mean 0 and a standard deviation of 1.

2. Multiply those random weights with the square root of (2/n). Here n is number of input units for that layer.

For other activation functions like Sigmoid or Hyperbolic Tangent

If we are using Sigmoid or Tanh as activation function in hidden layers, we need to go through following steps to implement Xavier initialization technique:

1. Generate random weights from a Gaussian distribution having mean 0 and a standard deviation of 1.

2. Multiply those random weights with the square root of (1/n). Here n is number of input units for that layer.

QUESTIONS

1. Which of the following is FALSE about Weights and Bias?

A. Biases are usually initialized to 0 (or close to 0)

B. Weights are usually initialized using Xavier technique

C. Both weight and bias are hyperparameters

D. None of the above

2. Which of the following is TRUE about Weight Initialization?

A. If weights are too high, it may lead to vanishing gradient

B. If weights are too low, it may lead to exploding gradient

C. Model may never converge due to wrong weight initialization

D. All of the above

3. Which of the following is FALSE about Weight Initialization?

A. Training period may increase due to wrong weight initialization

B. Vanishing and exploding gradient issues may arise due to wrong weight initialization

C. Initially, we should set all weights to zero while training

D. None of the above

4. Which of the following is FALSE about Xavier Weight Initialization with ReLU function?

A. We generate random weights from a Gaussian distribution (mean = 0, std dev = 1)

B. Multiply above random weights with the square root of (1/n) where n is number of input units

C. Multiply above random weights with the square root of (2/n) where n is number of input units

D. None of the above

5. Which of the following is FALSE about Xavier Weight Initialization with Sigmoid function?

A. We generate random weights from a Gaussian distribution (mean = 0, std dev = 1)

B. Multiply above random weights with the square root of (1/n) where n is number of input units

C. Multiply above random weights with the square root of (2/n) where n is number of input units

D. None of the above

ANSWERS

Answer 1: C

Answer 2: C

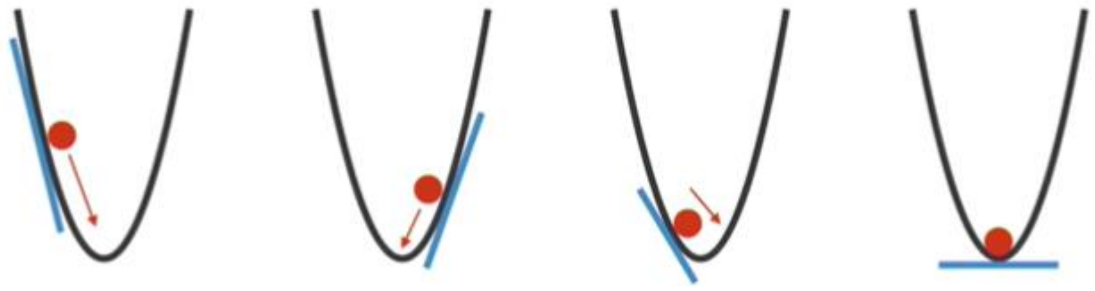
Answer 3: C

Answer 4: B

Answer 5: C

CHAPTER 5
GRADIENT DESCENT AND LEARNING RATE

Gradient Descent is the widely used algorithm to optimize our neural networks in deep learning.



Gradient Descent algorithm is used to minimize the loss function. We take gradient of the loss function w.r.t the weights. Our main aim is to get into the global minima (as shown in figure 4). In figure 1, we are on the left of the curve, so gradient of the slope of the line is negative. In figure 2, gradient is positive. We need to consider two main things in it.

1. Which direction to descend (consider first two diagrams)?

2. How much to descend? What should be the step size?

Initially, we take large steps and gradually we start taking baby steps to reach the global minima. If we don't change step size with time, we may overshoot the global minima and our model may never converge. So, instead of using constant learning rate, we need to use adaptive learning rate so that we may be able to change the step size as per the conditions.

There are a lot of flavors of gradient descent, so let’s discuss few of them.

Batch Gradient Descent (BGD)

In Batch Gradient Descent, we process the entire training dataset in one iteration. Weights are updated once an epoch is completed.

Stochastic Gradient Descent (SGD)

In Stochastic Gradient Descent, we process a single observation (instead of entire dataset) from the training dataset in each iteration. We calculate the error, gradient and new weights and keep updating the model for each observation in the training dataset.

Mini Batch Gradient Descent (MBGD)

In Mini Batch Gradient Descent, we process a small subset of the training dataset in each iteration. In other words, we can say that it is a compromise between BGD and SGD.

Note: Batch size is a very important hyper-parameter. It can vary depending on the dataset. So, deciding the batch size is a very crucial step.

Variants of SGD

1. Momentum based (Nesterov Momentum)

2. Based on adaptive learning rate (Adagrad, AdaDelta, RMSprop)

3. Combination of momentum and adaptive learning rate (Adam)

Momentum

Momentum helps in accelerating SGD in a relevant direction. So, it's a good idea to also consider momentum for every parameter. It has following advantages:

1. Avoids local minima: As momentum adds up speed and hence increases the step size, optimizer will not get trapped in local minima.

2. Faster convergence: Momentum makes the convergence faster as it increases the step size due to the gained speed.

Nesterov Momentum

It finds out the current momentum and based upon that approximates the next position. And then, it calculates the gradient w.r.t next approximated position instead of calculating gradient w.r.t current position. This thing prevents us from going too fast and results in increased responsiveness, which significantly increases the performance of SGD.

Adagrad

It mainly focuses on adaptive learning rate instead of momentum.

In standard SGD, learning rate is always constant. It means, we have to go with same speed irrespective of the slope. This seems impractical in real life. What happen if we know that we should slow down or speed up? What happen if we know that we should accelerate more in this direction and decelerate in that direction? It's not possible using the standard SGD.

Adagrad keeps updating the learning rate instead of using constant learning rate. It accumulates the sum of squares of all of the gradients, and use that to normalize the learning rate, so that now the learning rate could be smaller or larger depending on how the past gradients behaved.

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. For this reason, it is well-suited for dealing with sparse data.

AdaDelta and RMSprop

AdaDelta and RMSprop are an extension of Adagrad.

As discussed in Adagrad section, Adagrad accumulates the sum of squares of all of the gradient, and use that to normalize the learning rate. Due to this, Adagrad encounters an issue. The issue is that learning rate in Adagrad keeps on decreasing due to which at a point learning almost stops.

To handle this issue AdaDelta and RMSprop decay the past accumulated gradient, so only a portion of past gradients are considered. Now, instead of considering all of the past gradients, we consider the moving average.

Adam

Adam is the finest Gradient Descent Optimizer and is widely used. It uses powers of both momentum and adaptive learning. In other words, Adam is RMSprop or AdaDelta with momentum.

Conclusion: Most of the above Gradient Descent methods are already implemented in the popular Deep Learning frameworks like TensorFlow, Keras, Theano, Caffe etc. However, Adam is currently the default recommended algorithm to be used as it utilizes both momentum and adaptive learning features.

Local and Global Minima

As discussed above, task of a Gradient Descent optimizer is to find out optimal weights for the parameters. But sometimes, it may end up in finding weights which are less than the optimal value which leads to inaccuracy of the model. Ideally, our SGD should reach till global minima but sometimes it gets stuck in the local minima and it becomes very hard to know that whether our SGD is in global minima or stuck in local minima.

How to avoid local minima?

Local minima is a major issue with gradient descent. Hyper-parameter tuning plays a vital role in avoiding local minima. There is no universal solution to this problem, but there are some methods which we can use to avoid local minima.

1. Increasing the learning rate: If the learning rate of the algorithm is too small, then it is more likely that SGD will get stuck in a local minima.

2. Add some noise while updating weights: Adding random noise to weights also sometimes helps in finding out the global minima.

3. Assign random weights: Repeated training with random starting weights is among the popular methods to avoid this problem, but it requires extensive computational time.

4. Use large number of hidden layers: Each hidden node in a layer starts out in a different random starting state. This allows each hidden node to converge to different patterns in the network. Parameterizing this size allows the neural network user to potentially try thousands (or tens of billions) of different local minima in a single neural network.

5. MOST EFFECTIVE ONE: Using momentum and adaptive learning based SGD: As discussed above, instead of using conventional gradient descent optimizers, try using optimizers like Adagrad, AdaDelta, RMSprop and Adam. Adam uses momentum and adaptive learning rate to reach the global minima.

Sometimes local minimas are as good as global minimas

Usually, it is not always necessary to reach the true global minimum. It is generally agreed upon that most of the local minimas have values which are close to the global minimum. There are a lot of papers and research which shows sometimes reaching to global minima is not easy. So, in these cases, if we manage to find an optimal local minima which is as good as global minima, we should use that.

QUESTIONS

1. What is the purpose of the Gradient Descent algorithm?

A. To normalize the inputs

B. To minimize the weights and bias

C. To minimize the loss function

D. To prevent model from overfitting

2. Gradient Descent computes derivative of loss function w.r.t:

A. input

B. activation value

C. weight

D. bias

3. Which of the following is TRUE about Batch Gradient Descent?

A. Performs model updates at the end of each training epoch

B. Requires a lot of memory while gradient computation

C. Training speed usually becomes very slow for large datasets

D. All of the above

4. Which of the following is FALSE about Stochastic Gradient Descent?

A. Performs model updates for each single observation

B. Frequent model updates can also result in a noisy gradient signal

C. Learning is slower than batch gradient descent

D. None of the above

5. Which of the following is FALSE about Mini Batch Gradient Descent?

A. Performs model updates per batch of training data

B. Batch size is a hyper-parameter which needs to be fine-tuned

C. It maintains a balance between BGD and SGD

D. None of the above

6. Which of the following is TRUE about Momentum?

A. It helps in accelerating SGD in a relevant direction

B. It helps to prevent unwanted oscillations by adding up the speed

C. It helps to know the direction of the next step with the knowledge of the previous steps

D. All of the above

7. Which of the following is TRUE about Momentum?

A. It helps in accelerating SGD in a relevant direction

B. It helps SGD in avoiding local minima

C. It helps in faster convergence

D. All of the above

8. Which of the following SGD variants is momentum based?

A. Nesterov

B. Adagrad

C. RMSprop

D. All of the above

9. Which of the following SGD variants is NOT momentum based?

A. Nesterov

B. AdaDelta

C. Adam

D. All of the above are momentum based

10. Which of the following SGD variants is NOT based on adaptive learning?

A. Nesterov

B. Adagrad

C. AdaDelta

D. RMSprop

11. Which of the following SGD variants is based on adaptive learning?

A. Adam

B. Adagrad

C. AdaDelta

D. All of the above

12. Which of the following SGD variants is based on both momentum and adaptive learning?

A. RMSprop

B. Adagrad

C. Adam

D. Nesterov

13. Which of the following SGD variants is based on both momentum and adaptive learning?

A. RMSprop

B. Adagrad

C. AdaDelta

D. None of the above

14. Which of the following is FALSE about Adagrad, AdaDelta, RMSprop and Adam?

A. AdaDelta and RMSprop are an extension of Adagrad

B. RMSprop considers both momentum and adaptive learning

C. Adagrad mainly focuses on adaptive learning rate instead of momentum

D. None of the above

15. Which of the following is TRUE about Nesterov, RMSprop and Adam?

A. Nesterov and RMSprop are an extension of Adam

B. Nesterov considers both momentum and adaptive learning

C. RMSprop mainly focuses on adaptive learning rate instead of momentum

D. All of the above

16. Which of the following is FALSE about Nesterov, RMSprop and Adam?

A. Nesterov is based on momentum only

B. RMSprop is based on adaptive learning only

C. Adam is based on both momentum and adaptive learning

D. None of the above

17. Which of the following is FALSE about Learning Rate?

A. It determines how quickly weights and bias are updated in a neural network

B. There are high chances that SGD may end up in global minima if learning rate is very small

C. Low learning rate may slow down the training process

D. None of the above

18. Which of the following is FALSE about Learning Rate?

A. Model may never converge if learning rate is too high

B. High learning rate speeds up the training process

C. Instead of adaptive learning rate, we should prefer constant learning rate

D. None of the above

19. Which of the following is a way to avoid local minima?

A. Increase the learning rate

B. Use momentum and adaptive learning

C. Add some noise while updating weights

D. All of the above

20. Which of the following is TRUE about local and global minima?

A. Sometimes local minimas are as good as global minimas

B. Hyper-parameter tuning plays a vital role in avoiding global minima

C. Ideally, SGD should reach till local minima and should not stuck in global minima

D. All of the above

21. Which of the following is FALSE about local and global minima?

A. Using constant learning rate is a good way to avoid local minima

B. Using random weights and adding noise sometimes help in getting global minima

C. We can avoid local minima by proper tuning of hyper-parameters

D. We will not get optimal weights if SGD is stuck in local minima

22. Which of the following is FALSE about loss functions?

A. Cross entropy can be used for both binary and multi-class classification problems

B. One hot encoding of the target variable is not required with sparse cross entropy

C. Mean square error is commonly used for regression problems

D. None of the above

ANSWERS

Answer 1: C

Answer 2: C

Answer 3: D

Answer 4: C

Answer 5: D

Answer 6: D

Answer 7: D

Answer 8: A

Answer 9: B

Answer 10: A

Answer 11: D

Answer 12: C

Answer 13: D

Answer 14: B

Answer 15: C

Answer 16: D

Answer 17: B

Answer 18: C

Answer 19: D

Answer 20: A

Answer 21: A

Answer 22: D

CHAPTER 6

CNN AND CAPSNETS

CNN (Convolutional Neural Network) is a feed-forward neural network as the information moves from one layer to the next. CNN is also called ConvNets. It consists of hidden layers having convolution and pooling functions in addition to the activation function for introducing non-linearity.

CNN is mainly used for image recognition. CNN first learns to recognize the components of an image (e.g. lines, corners, curves, shapes, texture etc.) and then learns to combine these components (pooling) to recognize larger structures (e.g. faces, objects etc.).

Layers in CNN

1. Convolutional Layer
2. ReLU Layer
3. Pooling Layer
4. Normalization Layer
5. Fully connected Layer

Computers see an input image as an array of pixels. Numerical representation of the pixels is processed through many layers of a CNN. Each input image passes through a series of hidden layers like convolutional layers with filters (kernels), ReLU layers, pooling layers and fully connected layers. These hidden layers perform feature extraction from the image.

Convolutional Layer

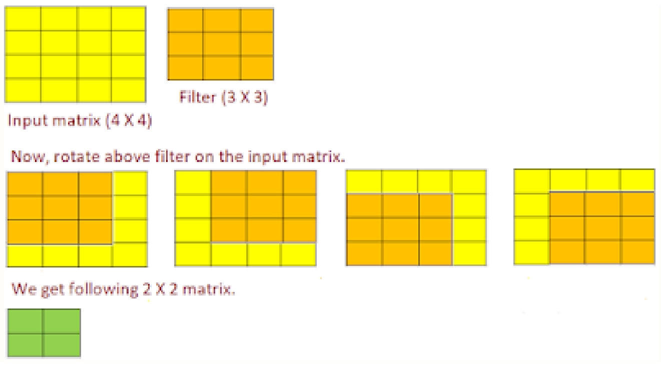
Convolution is the first layer to extract features from an input image. This layer uses a matrix filter and performs convolution operation to detect patterns in the image. Convolution of an image with different filters can perform operations such as edge detection, blur and sharpen by applying filters.

Convolution is a mathematical operation that happens between two matrices (image matrix and a filter or kernel) to form a third matrix as an output (convoluted matrix). This output is also called feature map matrix.

Filters (Kernels)

Filters act as pattern detectors. Filters help in finding out edges, curves, corners, textures, colors, dark and light areas in the image and many other details like height, width, and depth etc. Kernels keep sliding over an entire image to extract different components or patterns of an image. First filters learn to extract simple features in initial convoluted layers, and later on these filters get more sophisticated in deeper layers and find out complex patterns.

We rotate this filter over an input matrix and get an output which is of less dimension.


Input matrix (4 X 4)
Filter (3 X 3)
Now, rotate above filter on the input matrix.
We get following 2 X 2 matrix.

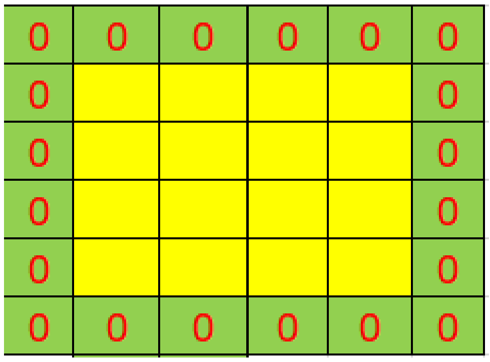
Formula: Consider that our input matrix dimension is **n X n**. Filter size is **f X f**. Then our output matrix would be **(n - f + 1) X (n - f + 1)**. Just replace **n** with **4**, **f** with **3** and observe that the output matrix comes out to be **2 X 2**.

Padding

We can observe that the input size is reduced from 4 X 4 to 2 X 2 after one convolution using 3 X 3 filter. This may lead to a problem. We may lose some information about edges and corners in the image. So, in order to preserve this information, we should use padding.

Type of Padding: We have two types of padding: Zero Padding and Valid Padding (no padding).

1. Zero Padding: Pad the image with zeros so that we don't lose any information about edges and corners.



In the above image, we have padded the input with zero. Now, if we use 3 X 3 filter over this, we get 4 X 4 output matrix (no reduction in dimensions) instead of 2 X 2.

2. Valid Padding: Drop the part of the image where the filter does not fit. This is called valid padding which keeps only valid part of the image. In this case, we compromise to lose some edge information in the image. We will get only 2 X 2 matrix in above example. We can go with this approach if we know that the information at edges is not that much useful and we can safely ignore that.

ReLU Layer

ReLU stands for Rectified Linear Unit for a non-linear operation. The output is ƒ(x) = max(0,x). ReLU’s main purpose is to introduce non-linearity in the ConvNets. It performs element-wise operation and set negative pixels to zero.

ReLU function is applied to the output matrix (feature map matrix) in the convolutional layer and converts it into a rectified feature map matrix.

We can also use other activation functions like tanh and sigmoid but generally ReLU performs better than other activation functions in many scenarios. So, by default, we consider ReLU over other activation functions.

Stride

Stride is the number of pixels shifts over the input matrix. Alternatively, stride can be thought as by how many pixels we want our filter to move as it slides across the image.

When the stride is 1, then we move the filters to 1 pixel at a time. When the stride is 2, then we move the filters to 2 pixels at a time and so on. We will use this concept in pooling layer.

Pooling Layer

Pooling layer is added after convolutional layer. Output of convolutional layer acts as an input to the pooling layer. Pooling layer does down-sampling of the image which reduces dimensionality by retaining important information. In this way, memory requirements are also reduced.

It does further feature extraction and detects multiple components of the image like edges, corners etc.

It converts the rectified feature map matrix to pooled feature map matrix.

Pooling Types

1. Max Pooling: It takes the maximum value from the rectified feature map.

2. Min Pooling: It takes the minimum value from the rectified feature map.

3. Average Pooling: It takes the average of all the elements from the rectified feature map.

4. Sum Pooling: It takes the sum of all the elements from the rectified feature map.

We can also specify padding parameter in pooling layer just like in convolutional layer.

Advantages of Pooling Layer

1. Reduces the resolution and dimensions and hence reduces computational complexity.

2. It also helps in reducing overfitting.

Normalization Layer

Normalization is a technique used to improve the performance and stability of the neural networks. It converts all inputs such that mean is zero and standard deviation is one.

Fully Connected Layer

Fully connected layers are used to connect every neuron in one layer to all the neurons in another layer. We flatten our pooled feature map matrix into vector and then feed that vector into a fully connected layer.

Hyperparameters in CNN

1. Number of convoluted layers

2. Number of kernels / filters in a convoluted layer

3. Kernel / Filter size in a convoluted layer

4. Padding in a convoluted layer (zero or valid padding)

Summary

1. Provide input image into convolution layer.

2. Choose parameters, apply filters with strides, padding if required.

3. Perform convolution on the image. Output from this layer is called feature map matrix.

4. Apply ReLU activation to the matrix. Output from this layer is called rectified feature map matrix.

5. Perform pooling to reduce dimensionality. Output from this layer is called pooled feature map matrix.

6. Add as many convolutional and pooling layers and keep repeating above steps.

7. Flatten the output (convert pooled feature map matrix to vector) and feed into a fully connected layer.

8. Output the class using an activation function (example: Logistic Regression with cost functions).

CapsNets (Capsule Neural Networks)

Capsule Neural Networks can be seen as an enhancement of Convolutional Neural Networks. In CNN, initial layers detect simple features like edges, curves, color gradients etc. Deeper convolutional layers start combining the simple features into comparatively complex features and so on. But in doing so, CNN does not take care of orientational and relative spatial relationships between the features or components. So, sometimes, CNN can be easily tricked.

For example, in face recognition, CNN does not take care of placements of eyes, nose, mouth, lips etc. Even if lips are near to eyes or eyes are below the mouth, it will still consider it a face. If all the features or components of face are available, it will consider it as a face without taking care of the orientation and placement of those components. Capsule networks take care of this.

Pooling layer problem in CNN

Pooling layer is used to perform down-sampling the data due to which a lot of information is lost. These layers reduce the spatial resolution, so their outputs are invariant to small changes in the inputs. This is a problem when detailed information must be preserved throughout the network. With CapsNets, detailed pose information (such as precise object position, rotation, thickness, skew, size, and so on) is preserved throughout the network. Small changes to the inputs result in small changes to the outputs—information is preserved. This is called "equivariance."

Capsule

Human brain is organized into modules called capsules. Considering this fact, concept of capsule was put forward by Hilton. A capsule can be considered as a group of neurons. We can add as many neurons to a capsule to capture different dimensions of an image like scale thickness, stroke thickness, width, skew, translation etc. It can maintain information such as equivariance, hue, pose, albedo, texture, deformation, speed, and location of the object.

Dynamic Routing Algorithm

Human brain has a mechanism to route information among capsules. On similar mechanism, dynamic routing algorithm was suggested by Hilton. This algorithm allows capsules to communicate with each other.

Squashing Function

Instead of ReLU, a new squashing function was suggested by Hilton known as novel squashing function. It is used to normalize the magnitude of vectors so that it falls between 0 and 1. The outputs from these squash functions tell us how to route data through various capsules that are trained to learn different concepts.

Limitations of Capsule Neural Networks

1. As compared to the CNN, the training time for the capsule network is slower because of its computational complexity.

2. It has been tested over MNIST dataset, but how will it behave on complex dataset, is still unknown.

3. This concept is still under research. So, it has a lot of scope for improvement.

QUESTIONS

1. CNN is best suited for:

A. Image Classification

B. Natural Language Processing

C. Image Captioning

D. All of the above

2. Which of the following layers is NOT a part of CNN?

A. Convolutional Layer

B. Pooling Layer

C. Code Layer

D. Fully connected Layer

3. Which of the following terms is NOT associated with CNN?

A. Filters (Kernels)

B. Forget Gates

C. Zero and Valid Padding

D. Strides

4. Which of the following is FALSE about CNN?

A. We must flatten the output before feeding it to a fully connected layer

B. There can be only one fully connected layer in CNN

C. We can use as many convolutional and pooling layers in CNN

D. None of the above

5. Which of the following is TRUE about Convolutional Layer in CNN?

A. It is used to extract features from an input image

B. It uses a matrix filter and performs convolution operation to detect patterns in the image

C. We may lose some information about edges and corners of an image during convolution

D. All of the above

6. Which of the following is FALSE about Kernels in CNN?

A. Kernels can be used in convolutional as well as in fully connected layers

B. Kernels act as pattern detectors

C. Kernels lead to dimensionality reduction

D. We may lose some information about edges and corners of an image using Kernels

7. **Which of the following is FALSE about Kernels in CNN?**

A. Kernels can be used in convolutional as well as in pooling layers

B. Kernels keep sliding over an image to extract different components or patterns of an image

C. Kernels extract simple features in initial layers and complex features in deeper layers

D. None of the above

8. Which of the following is FALSE about Padding in CNN?

A. Padding is used to prevent the loss of information about edges and corners during convolution

B. There are two types of padding: Zero Padding and Valid Padding (no padding)

C. There is no reduction in dimension when we use valid padding

D. In zero padding, we pad the image with zeros so that we do not lose any edge information

9. Which of the following is FALSE about Padding in CNN?

A. We should use valid padding if we know that information at edges is not that much useful

B. We compromise to lose some edge information of the image in zero padding

C. There is no reduction in dimension when we use zero padding

D. In valid padding, we drop the part of the image where the filter does not fit

10. Which of the following is FALSE about zero padding?

A. It is used to preserve the spatial size of the input volume

B. It is used to preserve edge information of the image

C. It is used to preserve resolution of the image

D. None of the above

11. Which of the following is TRUE about Padding in CNN?

A. Padding is used in convolution layer as well as in pooling layer

B. Padding is used in convolution layer as well as in fully connected layer

C. Padding is used in fully connected layer as well as in pooling layer

D. Padding is used only in convolution layer

12. Filter of size 3X3 is rotated over input matrix of size 4X4 (stride=1). What will be the size of output matrix after applying zero padding?

A. 4X4

B. 3X3

C. 2X2

D. 1X1

13. Filter of size 3X3 is rotated over input matrix of size 4X4 (stride=1). What will be the size of output matrix after applying valid padding?

A. 4X4

B. 3X3

C. 2X2

D. 1X1

14. Which of the following is FALSE about Pooling Layer in CNN?

A. Pooling layer must be added after each convolutional layer

B. Output of convolutional layer acts as an input to the pooling layer

C. It does down-sampling of an image which reduces dimensions by retaining vital information

D. It does feature extraction and detects components of the image like edges, corners etc.

15. Which of the following is TRUE about Pooling Layer in CNN?

A. We can use Max, Min, Average or Sum pooling in CNN

B. It helps in retaining the most useful information and throwing away useless information

C. It reduces resolution and dimension and hence reduces computational complexity

D. All of the above

16. Which of the following is FALSE about Pooling Layer in CNN?

A. It helps in reducing overfitting

B. It reduces computational complexity

C. It increases image resolution

D. None of the above

17. Which of the following is NOT a hyper-parameter in CNN?

A. Code size for compression

B. Number and size of kernels in a convolutional layer

C. Padding in a convolutional layer (zero or valid padding)

D. Number of convolutional layers

18. Which of the following is FALSE about ConvNets and CapsNets?

A. CapsNets take care of orientational and relative spatial relationships while ConvNets do not

B. CapsNets can be seen as an enhancement of ConvNets

C. Training period of ConvNets is larger as compared to CapsNets

D. None of the above

19. Which of the following is TRUE about CapsNets?

A. It is composed of capsules which can be considered as a group of neurons

B. We can add as many neurons to a capsule to capture different dimensions of an image

C. Capsules communicate with each other using Dynamic Routing algorithm

D. All of the above

20. Which of the following is TRUE about CapsNets?

A. Instead of ReLU, we use novel squashing function in CapsNets

B. CapsNets use pooling layer to maintain equivariance

C. ConvNets are computationally more complex as compared to CapsNets

D. All of the above

21. Which of the following is FALSE about using low resolution images in CNN and CapsNets?

A. Computational complexity of the model is reduced using low resolution images

B. Low resolution images lead to higher accuracy as compared to high resolution images

C. We need less number of neuron to process low resolution images which prevents the model from becoming too complex

D. Performance of the model is increased using low resolution images

22. Which of the following is a valid reason for not using fully connected networks for image recognition?

A. It creates a lot more parameters for computation as compared to CNN

B. It may overfit easily as compared to CNN

C. CNN is far efficient in terms of performance and accuracy for image recognition

D. All of the above

23. How many input nodes are required to process a grayscale image of 28X28?

A. 28 X 1

B. 28 X 28

C. 56 X 56

D. 56 X 1

24. How many input nodes are required to process a colored image of 28X28?

A. 28 X 28 X 3

B. 28 X 28 X 1

C. 56 X 56 X 1

D. 56 X 56 X 3

ANSWERS

Answer 1: A

Answer 2: C

Answer 3: B

Answer 4: B

Answer 5: D

Answer 6: A

Answer 7: D

Answer 8: C

Answer 9: B

Answer 10: C

Answer 11: A

Answer 12: A

Answer 13: C

Answer 14: A

Answer 15: D

Answer 16: C

Answer 17: A

Answer 18: C

Answer 19: D

Answer 20: A

Answer 21: B

Answer 22: D

Answer 23: B

Answer 24: A

CHAPTER 7

GOOGLE CLOUD VISION API

Google Cloud Vision API helps in label detection, face detection, logo detection, landmark detection and text detection. Let's see how can we use this API for OCR (Optical Character Recognition)?

Step 1: Setup a Google Cloud Account

A) Go to: https://console.cloud.google.com

B) Login with your google credentials

C) You will see a dashboard. Create a project if not already created.

Step 2: Enable Cloud Vision API

A) Go to Google cloud console

B) Click on Navigation Menu

C) Click on API & Services >> Library

D) Search "cloud vision" and you will get the "Cloud Vision API". Enable this API if not already enabled.

Step 3: Download credentials file

A) Go to Google cloud console

B) Click on Navigation Menu

C) Click on API & Services >> Credentials

D) Click on "Create Credentials" dropdown >> Service account key >> New service account

E) Enter Service account name whatever you want

F) Select any role. For example, select Project >> Viewer.

G) Save the file as JSON on your hard drive. Rename it to "credentials.json" or any name your like.

Step 4: Add billing information

A) Go to Google cloud console

B) Click on Navigation Menu

C) Click on Billing and add credit card information

Now open the Jupyter notebook and try using this API for text detection or OCR.

Step 5: Import required libraries

```
from googleapiclient.discovery import build
from oauth2client.client import GoogleCredentials
from base64 import b64encode
```

You may get import error "no module name..." if you have not already installed Google API Python client. Use following command to install it.

```
pip install --upgrade google-api-python-client
```

If you also get import error for oauth2client, you must install it using following command:

```
pip install --upgrade oauth2client
```

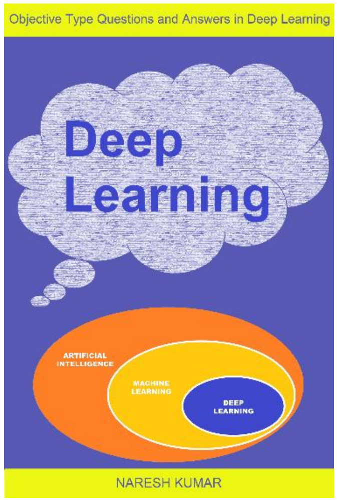
Step 6: Load credentials file

Load the credentials file (which we created in step 3) and create a service object using it.

```
CREDENTIAL_FILE = 'credentials.json'
credentials = GoogleCredentials.from_stream(CREDENTIAL_FILE)
service = build('vision', 'v1', credentials=credentials)
```

Step 7: Load image file (from which we need to extract the text)

I will load an image of cover page of this book and encode it so that it becomes compatible with the cloud vision API.



```
IMAGE_FILE = ‘book_cover_page.jpg'
```

```
with open(IMAGE_FILE, 'rb') as file:
    image_data = file.read()
    encoded_image_data = b64encode(image_data).decode('UTF-8')
```

Step 8: Create a batch request

We will create a batch request which we will send to the cloud vision API. In the batch request, we will include the above encoded image and the instruction as **TEXT_DETECTION**.

```
batch_request = [{
    'image':{'content':encoded_image_data},
    'features':[{'type':'TEXT_DETECTION'}],
}]
```

Step 9: Create a request

```
request = service.images().annotate(body={'requests':batch_request})
```

Step 10: Execute the request

```
response = request.execute()
```

This step will throw an error if you have not enabled billing (as mentioned in step 4). So, you must enable the billing in order to use Google Cloud Vision API. The charges are very reasonable. So, don't think too much and provide credit card details. For me, Google charged INR 1 and then refunded it back.

Step 11: Process the response

For error handling, include this code:

```
if 'error' in response:
    raise RuntimeError(response['error'])
```

We are interested in text annotations here. So, fetch it from the response and display the results.

```
labels = response['responses'][0]['textAnnotations']
extracted_text = extracted_texts[0]
print(extracted_text['description'], extracted_text['boundingPoly'])
```

Output

```
Objective Type Questions and Answers in Deep Learning
Deep
Learning
ARTIFICIAL
INTELLIGENCE
MACHINE
LEARNING
DEEP
LEARNING
NARESH KUMAR
 {'vertices': [{'x': 42, 'y': 77}, {'x': 2365, 'y': 77}, {'x': 2365, 'y': 3523}, {'x': 42, 'y': 3523}]}
```

You can test the above code using different images and check the accuracy of the API.

QUESTIONS

1. Which of the following is FALSE about Google Cloud Vision API?

A. You must create credentials before using the API

B. You must setup billing information before using the API as it is a paid service

C. You must have a google account to use this API

D. None of the above

2. Which of the following is FALSE about Google Cloud Vision API?

A. We can use it when we do not have that much training data

B. We can use it when we do not have time and money to create a model from scratch

C. We can use it when we have sensitive customer data

D. None of the above

3. Google Cloud Vision API can be used for:

A. Label detection and Landmark detection

B. Face detection and Logo detection

C. Optical Character Recognition

D. All of the above

ANSWERS

Answer 1: D

Answer 2: C

Answer 3: D

CHAPTER 8

RNN AND LSTM

RNN stands for Recurrent Neural Network. It is a type of neural network which contains memory and best suited for sequential data. RNN is used by Apples Siri and Googles Voice Search. Let's discuss some basic concepts of RNN.

Best suited for sequential data

RNN is best suited for sequential data. It can handle arbitrary input / output lengths. RNN uses its internal memory to process arbitrary sequences of inputs.

This makes RNNs best suited for predicting what comes next in a sequence of words. Like a human brain, particularly in conversations, more weight is given to recency of information to anticipate sentences.

RNN that is trained to translate text might learn that "dog" should be translated differently if preceded by the word "hot".

RNN has internal memory

RNN has memory capabilities. It memorizes previous data. While making a decision, it takes into consideration the current input and also what it has learned from the inputs it received previously. Output from previous step is fed as input to the current step creating a feedback loop.

So, it calculates its current state using set of current input and the previous state. In this way, the information cycles through a loop.

In nutshell, we can say that RNN has two inputs, the present and the recent past. This is important because the sequence of data contains crucial information about what is coming next, which is why a RNN can do things other algorithms can't.

Types of RNN

1. One to One: It maps one input to one output. It is also known as Vanilla Neural Network. It is used to solve regular machine learning problems.

2. One to Many: It maps one input to many outputs. Example: Image Captioning. An image is fetched into the RNN system and it provides the caption by considering various objects in the image.

Caption: "A dog catching a ball in mid-air"

3. Many to One: It maps sequence of inputs to one output. Example: Sentiment Analysis. In sentiment analysis, a sequence of words are provided as input, and RNN decides whether the sentiment is positive or negative.

4. Many to Many: It maps sequence of inputs to sequence of outputs. Example: Machine Translation. A sentence in a particular language is translated into other languages.

Forward and Backward Propagation

Forward Propagation: We do forward propagation to get the output of the model and check its accuracy and get the error.

Backward Propagation: Once the forward propagation is completed, we calculate the error. This error is then back-propagated to the network to update the weights.

We go backward through the neural network to find the partial derivatives of the error (loss function) with respect to the weights. This partial derivative is now multiplied with learning rate to calculate step size. This step size is added to the original weights to calculate new weights. That is how a neural network learns during the training process.

Vanishing and Exploding Gradients

Let's first understand what is gradient?

Gradient: As discussed above in back-propagation section, a gradient is a partial derivative with respect to its inputs. A gradient measures how much the output of a function changes, if you change the inputs a little bit.

You can also think of a gradient as the slope of a function. Higher the gradient, steeper the slope and the faster a model can learn. If the slope is almost zero, the model stops to learn. A gradient simply measures the change in all weights with regard to the change in error.

Gradient issues in RNN

While training an RNN algorithm, sometimes gradient can become too small or too large. So, the training of an RNN algorithm becomes very difficult in this situation. Due to this, following issues occur:

1. Poor Performance
2. Low Accuracy
3. Long Training Period

Exploding Gradient: When we assign high importance to the weights, exploding gradient issue occurs. In this case, values of a gradient become too large and slope tends to grow exponentially. This can be solved using following methods:

1. Identity Initialization
2. Truncated Back-propagation
3. Gradient Clipping

Vanishing Gradient: This issue occurs when the values of a gradient are too small and the model stops learning or takes way too long because of that. This can be solved using following methods:

1. Weight Initialization
2. Choosing the right Activation Function
3. LSTM (Long Short-Term Memory)

Best way to solve the vanishing gradient issue is the use of LSTM (Long Short-Term Memory).

LSTM

A usual RNN has a short-term memory. So, it is not able to handle long term dependencies. Using LSTM, it can also have a long-term memory. LSTM is an extension for RNA, which extends its memory. LSTM's enable RNN's to remember their inputs over a long period of time so that RNN become capable of learning long-term dependencies.

In this way, LSTM solves the vanishing gradients issue in RNN. It keeps the gradients steep enough and therefore make training relatively short and the accuracy high.

Gated Cells in LSTM

LSTM is comprised of different memory blocks called cells and manipulations in these cells are done using gates. LSTMs store information in these gated cells. The data can be stored, deleted and read from these gated cells much like the data in a computer's memory. Gates of these cells open and close based on some decisions.

These gates are analog gates (instead of digital gates) and their outputs range from 0 to 1. Analog has the advantage over digital of being differentiable, and therefore suitable for back-propagation.

We have following types of gates in LSTM:

1. Forget Gate: It decides what information it needs to forget or throw away. It outputs a number between 0 and 1. A 1 represents "completely keep this" while a 0 represents "completely forget this."

2. Input Gate: The input gate is responsible for the addition of information to the cell state. It ensures that only that information is added to the cell state that is important and is not redundant.

3. Output Gate: Its job is to select useful information from the current cell state and showing it out as an output.

Squashing / Activation Functions in LSTM

1. Logistic (sigmoid): Outputs range from 0 to 1.

2. Hyperbolic Tangent (tanh): Outputs range from -1 to 1.

Bidirectional RNN

Bidirectional RNNs take an input vector and train it on two RNNs. One of them gets trained on the regular RNN input sequence while the other on a reversed sequence. Outputs from both RNNs are next concatenated, or combined.

Applications of RNN

1. Natural Language Processing (Text mining, Sentiment analysis, Text and Speech analysis, Audio and Video analysis)

2. Machine Translation (Translate a language to other languages)

3. Time Series Prediction (Stock market prediction, Algorithmic trading, Weather prediction,

Understanding DNA sequence etc.)

4. Image Captioning

QUESTIONS

1. Which of the following model is best suited for sequential data?

A. Convolutional Neural Networks (ConvNets)

B. Capsule Neural Networks (CapsNets)

C. RNN (Recurrent Neural Network)

D. Autoencoders

2. Which of the following model contains internal memory?

A. Convolutional Neural Networks (ConvNets)

B. Capsule Neural Networks (CapsNets)

C. RNN (Recurrent Neural Network)

D. Autoencoders

3. Which of the following is FALSE about LSTM?

A. LSTM is an extension for RNA which extends its memory

B. LSTM enables RNN to learn long-term dependencies

C. LSTM solves the exploding gradients issue in RNN

D. None of the above

4. Which of the following is TRUE about LSTM?

A. It uses forget gates, input gates and output gates

B. These gates are analog in nature

C. It uses feedback loop to remember the data

D. All of the above

5. Which of the following is FALSE about gated cells in LSTM?

A. LSTMs store information in these gated cells

B. These gates are digital in nature

C. Gates of these cells open and close based on some decisions

D. Data can be stored, deleted and read from these gated cells like computer storage

6. Which of the following is FALSE about Forget Gate in LSTM?

A. It decides what information it needs to forget or throw away

B. It outputs a number between 0 and 1

C. 0 represents completely keep this info while 1 represents completely forget this info

D. None of the above

7. Which of the following types of RNN is also called Vanilla Neural Network?

A. One to One

B. One to Many

C. Many to One

D. Many to Many

8. Which of the following types of RNN is best suited for image captioning?

A. One to One

B. One to Many

C. Many to One

D. Many to Many

9. Which of the following types of RNN is best suited for sentiment analysis?

A. One to One

B. One to Many

C. Many to One

D. Many to Many

10. Which of the following types of RNN is best suited for machine translation?

A. One to One

B. One to Many

C. Many to One

D. Many to Many

11. Which of the following is NOT an application of RNN?

A. Time series prediction

B. Anomaly detection

C. Weather prediction

D. Stock market prediction

12. Which of the following is NOT an application of RNN?

A. Algorithmic trading

B. Understanding DNA sequence

C. Image compression

D. Image captioning

13. Which of the following is an application of RNN?

A. Text mining

B. Sentiment analysis

C. Text and Speech analysis

D. All of the above

14. Which of the following is an application of RNN?

A. Natural Language Processing

B. Audio and Video analysis

C. Stock market prediction

D. All of the above

ANSWERS

Answer 1: C

Answer 2: C

Answer 3: C

Answer 4: D

Answer 5: B

Answer 6: C

Answer 7: A

Answer 8: B

Answer 9: C

Answer 10: D

Answer 11: B

Answer 12: C

Answer 13: D

Answer 14: D

CHAPTER 9

REGULARIZATION AND DROPOUT

Ideally, the neural networks should never underfit and overfit and maintain good generalization capabilities. For this purpose, we use various regularization techniques in our neural networks. Below is the list of some of the regularization techniques which are commonly used to improve the performance and accuracy of the neural networks in deep learning.

L1 and L2 Regularization

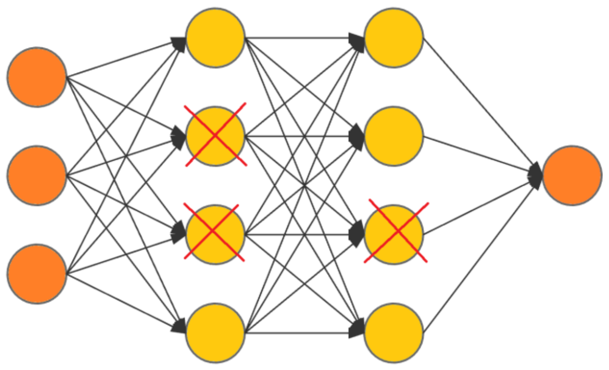
L1 and L2 are the most common types of regularization techniques used in machine learning as well as in deep learning algorithms. These update the general cost function by adding another term known as the regularization penalty.

Dropout

Dropout is an effective regularization technique used in neural networks which increases generalization capabilities of a deep learning model and prevent it from overfitting.

Dropout can be seen as temporarily deactivating or ignoring neurons in the hidden layers of a network. Probabilistically dropping out nodes in the network is a simple and effective regularization method. We can switch off some neurons in a layer so that they do not contribute any information or learn any information and the responsibility falls on other active neurons to learn harder and reduce the error.

You can notice in below diagram that we have dropped (crossed) two neurons in first hidden layer and one neuron in second hidden layer.



Basic points about dropout

1. Dropout is implemented per-layer in a neural network. Dropout can be implemented in hidden and input layers, but not in output layers.

We can use different probabilities for dropout on each layer. As mentioned previously, dropout should not be implemented on output layer, so the output layer would always have keep_prob = 1 and the input layer has high keep_prob such as 0.9 or 1.

If a hidden layer has keep_prob = 0.8, this means that on each iteration, each unit has 80% probability of being included and 20% probability of being dropped out.

This probability acts as a hyper-parameter and we should carefully decide how many neurons we want to deactivate in a given hidden layer.

2. Dropout can be used with many types of layers, such as dense fully connected layers, convolutional layers, and recurrent layers such as the long short-term memory network (LSTM) layers.

3. Dropout should be implemented only during training phase, not in testing phase.

4. Dropout can be compared to bagging technique in machine learning. In bagging, all trees are not trained on all the features. Similarly, using dropout, all the hidden layers are not trained on all the features.

Advantages

1. Reduces overfitting and hence increases the accuracy of the model

2. Improves the performance of neural networks on supervised learning tasks in vision, speech recognition, document classification and computational biology, obtaining state-of-the-art results on many benchmark datasets.

3. Computationally cheap as compared to other regularization methods.

Disadvantages

1. Introduces sparsity: If we use dropout to a large extent, activations inside the hidden layers may become sparse. You can correlate it with sparse autoencoders.

2. Dropout makes training process noisy as it forces nodes within a layer to probabilistically take on more or less responsibility for the inputs.

Data Augmentation

Creating new data by making reasonable modifications to the existing data is called data augmentation. Let’s take an example of our MNIST dataset (hand written digits). We can easily generate thousands of new similar images by rotating, flipping, scaling, shifting, zooming in and out, cropping, changing or varying the color of the existing images.

We can use data augmentation technique when our model is overfitting due to less data.

In many cases in deep learning, increasing the amount of data is not a difficult task as we discussed above the case of MNIST dataset. In machine learning, this task is not that easy as we need labeled data which is not easily available.

QUESTIONS

1. Which of the following is NOT a generalization technique in neural networks?

A. Xavier Weight Initialization

B. Dropout

C. L1 and L2 Regularization

D. Data Augmentation

2. Which of the following is FALSE about Dropout?

A. Dropout is a regularization technique

B. Dropout reduces overfitting

C. Dropout solves vanishing gradient problem

D. None of the above

3. Which of the following is TRUE about Dropout?

A. Dropout randomly switches off some neurons in the network

B. Dropout solves overfitting and underfitting problem

C. Dropout solves vanishing and exploding gradient problem

D. All of the above

4. Which of the following is FALSE about Dropout?

A. Dropout is a learnable parameter in the network

B. Dropout increases the accuracy and performance of the model

C. Dropout introduces sparsity in the network

D. Dropout makes training process noisy

5. Which of the following is FALSE about Dropout?

A. Dropout is implemented per layer in a network

B. Dropout is a hyper-parameter

C. Dropout can be used in input, hidden and output layers

D. None of the above

6. Which of the following is TRUE about Dropout?

A. Dropout can be implemented only in hidden layers

B. Dropout can be implemented in hidden and input layer

C. Dropout can be implemented in hidden and output layer

D. Dropout can be implemented in input, hidden and output layers

7. Which of the following is TRUE about Dropout?

A. Dropout should be implemented only during training phase, not in testing phase

B. Dropout should be implemented during training phase as well as during testing phase

C. Dropout can be compared to boosting technique in machine learning

D. Dropout is computationally complex as compared to L1 and L2 regularization methods

8. Which of the following is FALSE about Dropout?

A. Dropout can be compared to bagging technique in machine learning

B. Dropout is implemented only in hidden layers

C. Dropout is implemented only during training phase of the model

D. All of the above

9. Data Augmentation helps in:

A. Reducing overfitting

B. Increasing generalization capacity of the network

C. Generating data from data

D. All of the above

ANSWERS

Answer 1: A

Answer 2: C

Answer 3: A

Answer 4: A

Answer 5: C

Answer 6: B

Answer 7: A

Answer 8: B

Answer 9: D

CHAPTER 10

BATCH NORMALIZATION

Batch normalization (batchnorm) is a technique to improve performance and accuracy of a neural network.

Batch normalization occurs per batch. That is why, it is called batch normalization. We normalize (mean = 0, standard deviation = 1) the output of a layer before applying the activation function, and then feed it into the next layer in a neural network. So, instead of just normalizing the inputs to the network, we normalize the inputs to each hidden layer within the network.

Many times, normalization and standardization terms are used interchangeably.

Advantages of Batch Normalization

1. Solves internal covariate shift: In a neural network, each hidden unit's input distribution changes every time when there is a parameter update in the previous layer. This is called internal covariate shift. This makes training slow and requires a very small learning rate and a good parameter initialization. This problem is solved by normalizing the layer's inputs over a mini-batch.

2. Solves vanishing and exploding Gradient issues: Unstable gradients like vanishing gradients and exploding gradients are the common issues which occur while training a neural network. By normalizing the outputs of each layer, we can significantly deal with this issue.

3. Training becomes faster: In traditional neural networks, if we use higher learning rate, we may face exploding gradient issue. Also, if we use higher learning rate, there are possibilities that network may not converge at all and keeps oscillating around the global minima. Due to this, we usually prefer lower learning rate in traditional neural networks. But, with lower learning rate, as the networks get deeper, gradients get smaller during back propagation, and so require even more iterations which increases training period. But, if we normalize the output of each layer, we can safely use higher learning rate due to which we can drastically reduce the training period.

4. Solves dying ReLU problem: ReLUs often die out during training the deep neural networks and many neurons stop contributing. But, with batch normalization, we can regulate the output of each hidden layer, which prevents this issue in deep neural networks.

5. Introduces regularization: Batch normalization all provides some sort of regularization to the neural network which increases the generalization capabilities on the network.

QUESTIONS

1. Which of the following is TRUE about Batch Normalization?

A. It occurs per epoch

B. In this process, we normalize the inputs to each hidden layer

C. It may lead to overfitting

D. All of the above

2. Which of the following is FALSE about Batch Normalization?

A. It may lead to overfitting

B. It solves dying ReLU problem

C. It introduces regularization in the network

D. It solves vanishing and exploding gradient issues

3. Which of the following is FALSE about Batch Normalization?

A. It occurs per batch

B. It increases overall training period

C. It makes the gradients stable

D. None of the above

ANSWERS

Answer 1: B

Answer 2: A

Answer 3: B

CHAPTER 11

HYPER-PARAMETERS

In order to minimize the loss and determine optimal values of weight and bias, we need to tune our neural network hyper-parameters. Hyperparameters are the parameters that the neural network can't learn itself via gradient descent or some other variant.

Learnable Parameters: Hyper-parameters are opposite of learnable parameters. Learnable parameters are automatically learned and then optimized by the neural network. For example, weights and bias are learnable by the neural networks. These are also called trainable parameters as these are optimized during the training process using gradient descent.

This is our responsibility to provide optimal values for these hyper-parameters from our experience, domain knowledge and cross-validation. We need to manually tweak these hyperparameters to get better accuracy from the neural networks.

List of hyperparameters used in neural networks

1. Number of hidden layers: Keep adding the hidden layers until the loss function does not minimize to a certain extent. General rule is that we should use a large number of hidden layers with proper regularization technique.

2. Number of units or neurons in a layer: Larger number of units in a layer may cause overfitting. Smaller number of units may cause underfitting. So, try to maintain a balance and use dropout technique.

3. Dropout: Dropout is regularization technique to avoid overfitting thus increasing the generalizing capabilities of the neural network. In this technique, we deliberately drop some units in a hidden layer to introduce generalization capabilities into it. Dropout value should range in between 20%-50% of number of neurons in a layer.

4. Activation Functions: Activation functions introduce non-linearity in a neural networks. Sigmoid, Step, Tanh, ReLU, Softmax are some of the commonly used activation functions in neural networks. Mainly, we use ReLU activation function for hidden layers and softmax for output layer.

5. Learning Rate: Learning rate determines how quickly weights and bias are updated in a neural network. If the learning rate is very small, learning process will significantly slow down and the model will converge too slowly. It may also end up in local minima and never reach global minima. Larger learning rate speeds up the learning but may not converge.

Learning rate is normally set somewhere between 0.01 and 0.0001. Usually a decaying learning rate is preferred.

6. Momentum: Momentum helps in accelerating SGD in a relevant direction. Momentum helps to know the direction of the next step with the knowledge of the previous steps. It helps to prevent oscillations by adding up the speed. A typical choice of momentum should be between 0.5 and 0.9.

7. Number of epochs: Number of epochs is the number of times the whole training data is shown to the network while training. Default number of epochs is 1.

8. Batch size: Batch size is the number of samples passed to the network at one time after which parameter update happens. This is also called mini-batch. It should be in power of 2. We commonly use batch sizes of 32, 64 and 128 depending upon the amount of data.

Hyperparameter Tuning

Following are some ways to tune hyperparameters in a neural network:

1. Coordinate Descent: It keeps all hyperparameters fixed except for one, and adjust that hyperparameter to minimize the validation error.

2. Grid Search: Grid search tries each and every hyperparameter setting over a specified range of values. This involves a cross-product of all intervals, so the computational expense is exponential in the number of parameters. Good part is that it can be easily parallelized.

3. Random Search: This is opposite of grid search. Instead of taking cross-product of all the intervals, it samples the hyperparameter space randomly. It performs better than grid search because grid search can take an exponentially long time to reach a good hyperparameter subspace. This can also be parallelized.

4. Cross-validation: We can also try cross-validation by trying different portions of dataset during training and testing.

QUESTIONS

1. Which of the following is NOT a hyperparameter in a neural network?

A. Number of hidden layers

B. Number of neurons in a layer

C. Weights and bias

D. Dropout

2. Which of the following is a hyperparameter in a neural network?

A. Activation Function

B. Learning Rate

C. Momentum

D. All of the above

3. Which of the following is a hyperparameter in a neural network?

A. Number of epochs

B. Batch size

C. Loss Function

D. All of the above

4. Which of the following is a hyperparameter tuning method in a neural network?

A. Coordinate Descent

B. Grid Search and Random Search

C. Cross-validation

D. All of the above

5. Which of the following is FALSE about Epochs in a neural network?

A. Number of epochs is the number of times the whole training data is shown to the network

B. Number of epochs is a learnable parameter

C. Time to complete one epoch depends upon the batch size

D. None of the above

6. Which of the following is FALSE about Batch Size in a neural network?

A. It is the number of samples passed to the network at one time after which parameter update happens

B. Batch size is a hyperparameter

C. Batch size should usually be taken in power of 2

D. None of the above

7. Using too many epochs while training a network may lead to:

A. High training time

B. Overfitting

C. Unnecessary time wastage

D. All of the above

8. Using large size batches while training a network may lead to:

A. Lesser accuracy

B. Overfitting

C. High consumption of computational resources

D. All of the above

9. Model accuracy is not improving after so many epochs. There could be an issue with:

A. Network architecture

B. Training data

C. Fine-tuning of hyperparameters

D. Any of the above

ANSWERS

Answer 1: C

Answer 2: D

Answer 3: D

Answer 4: D

Answer 5: B

Answer 6: D

Answer 7: D

Answer 8: C

Answer 9: D

CHAPTER 12

AUTOENCODERS

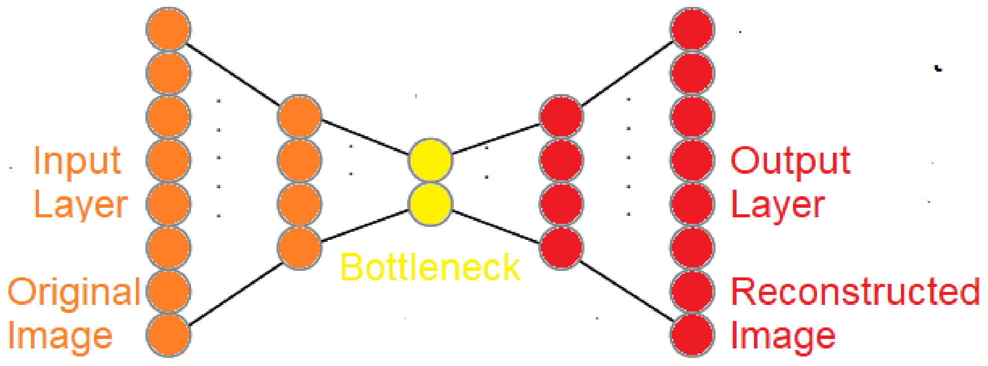
Autoencoder is a special kind of neural network in which the output is nearly same as that of the input. It is an unsupervised deep learning algorithm. We can consider an autoencoder as a data compression algorithm which performs dimensionality reduction for better visualization.

Components of Autoencoders

1. Encoder: It is the layer in which the model learns how to reduce the input dimensions and compress the input data into an encoded representation. This is the part of the network that compresses the input into a latent space representation.

2. Bottleneck / Code: It is the layer that contains the compressed representation of the input data. This is the lowest possible dimensions of the input data. It decides which aspects of the data are relevant and which aspects can be thrown away.

3. Decoder: It is the layer in which the model learns how to reconstruct the data from the encoded representation to be as close to the original input. The decoded image is a lossy reconstruction of the original image.



Reconstruction Loss

This is the method that measures how well the decoder is performing and how close the output is to the original input.

Properties of Autoencoders

1. Unsupervised: Autoencoder is considered as an unsupervised learning technique as it doesn't need explicit labels to train on.

2. Data-specific: Autoencoders are only able to compress and decompress the data similar to what they have been trained on. For example, an autoencoder which has been trained on human faces, would not perform well with the images of buildings.

3. Lossy: The output of the autoencoder will not be exactly the same as the input, it will be a close but degraded representation.

Working of an Autoencoder

Autoencoders compress the input into a latent-space representation and then reconstruct the output from this representation. We calculate the loss by comparing the input and output. This difference between the input and output is called reconstruction loss. Main objective of an autoencoder is to minimize this reconstruction loss so that the output is similar to the input. To reduce this reconstruction loss, we back propagate through the network and update the weights using gradient descent algorithm.

Generalization in Autoencoders

Autoencoder should have generalization capabilities. Our autoencoder should be able to recreate the original observation but at the same time should not memorize the training data such that the model learns a generalization. In other words, autoencoders should have some generalization capabilities. Mainly all types of autoencoders like undercomplete, sparse, convolutional and denoising autoencoders use some mechanism to have generalization capabilities.

How to increase generalization capabilities of an autoencoders?

1. Keep the code layer small so that there is more compression of data. More is the data compression, more is the generalization.

2. Limit the number of nodes in the hidden layers of the network (undercomplete autoencoders).

3. Use L1 and L2 regularization (sparse autoencoders)

4. Add random noise to the inputs and let the autoencoder recover the original noise-free data (denoising autoencoder)

Types of Autoencoders

1. Undercomplete autoencoder: In this type of autoencoder, we limit the number of nodes present in the hidden layers of the network. In this way, it also limits the amount of information that can flow through the network which makes our model to learn only the most important attributes of the input data.

By limiting the number of nodes in the hidden layers, we can make sure that our model does not memorize the training data and have some generalization capabilities.

For regularization and generalization, we don't use any regularization penalty to train our model, we just limit the number of nodes in the hidden layers.

2. Sparse autoencoder: We don’t limit the number of nodes in the hidden layers unlike undercomplete autoencoders. We introduce regularization techniques to regularize or penalize activations instead of weights in our loss function so that it activates only a small number of neurons in a given hidden layer.

Individual nodes of a trained model which activate are data-dependent, different inputs will result in activations of different nodes through the network. For regularization, we can use L1 regularization or KL-divergence regularization techniques.

3. Denoising autoencoder: In this type of autoencoder, we introduce generalization by slightly corrupting the input data (add some random noise) but still maintain the uncorrupted data as our target output. With this approach, our model isn't able to simply develop a mapping which memorizes the training data because our input and target output are no longer the same. In this way, we train the autoencoder to reconstruct the input from a corrupted version of it.

4. Convolutional autoencoder: It uses convoluted layers to compress images with the help of kernels (filter). Then it uses max pooling to further down-sample the image.

Applications of Autoencoders

1. Signal Denoising: Denoising or noise reduction is the process of removing noise from a signal. The signal can be an image, audio or a scanned document.

2. Dimensionality Reduction for Visualization: Lesser the dimension, better the visualization. Autoencoders outperform PCA in this regard as autoencoders work really well with non-linear data while PCA is only meant for linear data.

3. Anomalies and outliers detection: Autoencoders learn to generalize the patterns. So, if anything is out of pattern, it can detect easily. For an anomaly, reconstruction loss is very high as compared to the regular data.

4. Image coloring: It is also used for image coloring.

Hyper-parameters in Autoencoders

1. Code Size: It represents number of nodes in the middle layer. Smaller size results in more compression.

2. Number of Layers: An autoencoder can be as deep as we wanted to be. We can have as many layers both in encoder and decoder.

3. Number of nodes per layer: Number of nodes per layer decreases with each subsequent layer in the encoder and increases back in the decoder.

4. Loss Function: You can set any loss function like mean square error, binary cross entropy etc. If input values are in the range of 0 to 1, we typically use cross entropy otherwise we can use the mean square error.

Loss Function is usually composed of two parts:

1. Reconstruction Loss: measures how much is the difference between the original data and reconstructed data.

2. Regularization Penalty: adds some penalty so that the model learns generalization.

Restricted Boltzmann Machines (RBM)

Restricted Boltzmann Machines are shallow and two-layer (input and hidden) neural networks. There is no output layer unlike Autoencoders.

The nodes or neurons are connected to each other across the layers, but no two nodes of the same layer are linked. Due to this restriction, it is called Restricted Boltzmann Machines instead of simple Boltzmann Machines.

It is probabilistic, unsupervised, generative deep learning algorithm. RBM’s objective is to find the joint probability distribution that maximizes the log-likelihood function.

Applications

1. Dimensionality reduction

2. Collaborative filtering for recommender systems

3. Feature learning

4. Topic modelling

5. Helps improve efficiency of supervised learning

QUESTIONS

1. Which of the following is FALSE about Autoencoders?

A. It is an unsupervised deep learning algorithm

B. It is like a data compression algorithm which performs dimensionality reduction

C. More the number of code layers, more is the data compression

D. In autoencoders, output is nearly same as that of the input

2. Which of the following is FALSE about Autoencoders?

A. It can only compress and decompress the data similar to what it has been trained on

B. All autoencoders are generative in nature

C. Difference between the input and output is called reconstruction loss

D. Output is not exactly same as the input, it is a close but degraded representation

3. Which of the following is FALSE about Autoencoders?

A. It compresses the input into a latent-space representation and then reconstruct the output from it

B. Its objective is to minimize this reconstruction loss so that the output is similar to the input

C. Autoencoders are best suited for image coloring, image captioning and image recognition

D. Autoencoder possess generalization capabilities

4. Which of the following is FALSE about Encoders in Autoencoders?

A. It is the layer in which the model learns how to reduce the input dimensions

B. It compresses the input into a latent space representation

C. Number of nodes per layer increases with each subsequent layer in the encoder

D. None of the above

5. Which of the following is FALSE about Decoders in Autoencoders?

A. It is the layer in which the model learns how to reconstruct the data

B. The decoded image is a lossy reconstruction of the original image

C. Number of nodes per layer decreases with each subsequent layer in the decoder

D. None of the above

6. Which of the following is TRUE about Encoders and Decoders in Autoencoders?

A. Number of nodes per layer increases with each subsequent layer in the encoder

B. Number of nodes per layer decreases with each subsequent layer in the encoder

C. Number of nodes per layer decreases with each subsequent layer in encoder and decoder

D. Number of nodes per layer increases with each subsequent layer in encoder and decoder

7. Which of the following is FALSE about Bottleneck / Code in Autoencoders?

A. Number of code layers is a hyper-parameter which needs to be fine-tuned

B. It decides which aspects of the data are relevant and which aspects can be thrown away

C. It is the layer that contains the compressed representation of the input data

D. This compressed representation is the lowest possible dimensions of the input data

8. Which of the following is FALSE about code layer in autoencoder?

A. Small code size results in more data compression

B. Small code size results in more generalization

C. Small code size results in more accuracy

D. None of the above

9. Which of the following is FALSE about Reconstruction Loss in Autoencoders?

A. This is the method that measures how well the decoder is performing

B. This is the method that measures how close the output is to the original input

C. For an anomaly, reconstruction loss is less as compared to the regular data

D. None of the above

10. Which of the following is TRUE about Undercomplete autoencoder?

A. We limit the number of nodes in the hidden layers to increase its generalization capabilities

B. We use L1 and L2 regularization to increase its generalization capabilities

C. We add random noise to the input to increase its generalization capabilities

D. All of the above

11. Which of the following is TRUE about Sparse autoencoder?

A. We limit the number of nodes in the hidden layers to increase its generalization capabilities

B. We use L1 and L2 regularization to increase its generalization capabilities

C. We add random noise to input to increase its generalization capabilities

D. All of the above

12. Which of the following is TRUE about Denoising autoencoder?

A. We limit the number of nodes in the hidden layers to increase its generalization capabilities

B. We use L1 and L2 regularization to increase its generalization capabilities

C. We add random noise to input to increase its generalization capabilities

D. All of the above

13. Which of the following is NOT a way to increase generalization in autoencoder?

A. Use larger code size

B. Use L1 and L2 regularization

C. Add some random noise to the input

D. Limit the number of nodes in the hidden layers

14. Which of the following is NOT a hyper-parameters in Autoencoders?

A. Batch size and Code size

B. Number of code layers

C. Number of nodes per layer

D. Number of layers in encoder and decoder

15. Which of the following is NOT a hyper-parameters in Autoencoders?

A. Number and size of Kernels

B. Regularization and Dropout

C. Sparsity parameters in sparse autoencoders

D. Activation functions such as sigmoid, tanh, softmax, and ReLU

16. Which of the following is FALSE about Code Size hyper-parameter in Autoencoders?

A. It represents number of nodes in the code layer

B. Larger size results in more compression

C. Smaller code size results in more generalization

D. None of the above

17. Which of the following is FALSE about PCA and Autoencoders?

A. Both PCA and Autoencoders are dimensionality reduction techniques

B. PCA works well with non-linear data while Autoencoders are best suited for linear data

C. Output of both PCA and Autoencoders is lossy

D. None of the above

18. Which of the following is NOT an application of Autoencoders?

A. Image coloring

B. Image captioning

C. Anomalies and outliers detection

D. Dimensionality reduction

19. Which of the following is NOT an application of Autoencoders?

A. Removing noise from an image, audio or a scanned document

B. Predicting next word in a sentence

C. Detecting anomalies in the signal

D. Lowering the dimensions for better visualization

20. Which of the following is FALSE about Autoencoders and RBM?

A. Both are unsupervised deep learning algorithms

B. Both are generative deep learning models

C. Both help in data compression

D. None of the above

21. Which of the following is FALSE about Restricted Boltzmann Machines (RBM)?

A. RBM is a discriminative deep learning algorithm

B. RBM is probabilistic and unsupervised deep learning algorithm

C. RBM is a shallow and two-layer (input and hidden) neural network

D. There is no output layer in RBM unlike Autoencoders

22. Which of the following is FALSE about Restricted Boltzmann Machines (RBM)?

A. No node is connected to each other across the layers

B. No two nodes of the same layer are linked in RBM

C. It finds joint probability distribution that maximizes the log-likelihood function

D. All of the above

23. Which of the following is NOT an application of Restricted Boltzmann Machines (RBM)?

A. Dimensionality reduction

B. Image captioning

C. Feature learning and topic modelling

D. Collaborative filtering for recommender systems

ANSWERS

Answer 1: C

Answer 2: B

Answer 3: C

Answer 4: C

Answer 5: C

Answer 6: B

Answer 7: A

Answer 8: C

Answer 9: C

Answer 10: A

Answer 11: B

Answer 12: C

Answer 13: A

Answer 14: B

Answer 15: A

Answer 16: B

Answer 17: B

Answer 18: B

Answer 19: B

Answer 20: B

Answer 21: A

Answer 22: A

Answer 23: B

CHAPTER 13

NATURAL LANGUAGE PROCESSING

NLP (Natural Language Processing) is a very interesting branch of Artificial Intelligence. Natural language is a language which we human use to communicate and interact with each other. In NLP, we are teaching computers to understand, interpret and manipulate human languages. In this article, we will focus on some of the preprocessing tasks which we perform on the raw data like Tokenization, Stemming, Lemmatization and Vectorization.

While processing a natural language which we human speak, we need to take care of following things:

1. Syntax: The sentence should be grammatically correct. The arrangement of words in a sentence should follow all the grammar rules defined by a language.

2. Semantics: It deals with the meaning of words and their interpretation within sentences.

3. Pragmatics: Same as semantics but it also consider context in which the word is used.

Applications of NLP

Applications of NLP (Natural Language Processing) are unlimited. I have listed few of those:

1. Machine translation (like Google Translate)

2. Sentiment analysis (reviews and comments on e-commerce and social-networking sites)

3. Text classification, generation and automatic summarization

4. Automated question answering and conversational interfaces (like chatbots)

5. Personal assistants (like Alexa, Siri, Google Assistant, Cortana etc.)

6. Auto-correct grammatical mistakes (MS Word and Grammarly use NLP to check grammatical errors)

7. Spam filtering, auto-complete, auto-correct, auto-tagging, topic modelling, sentence segmentation, speech recognition, part of speech tagging, named entity recognition, duplicates detection and a lot more...

NLP Toolkit (library) in Python

There are a lot of libraries in Python for NLP but the most commonly used library is NLTK (Natural Language Toolkit). It provides very efficient modules for preprocessing and cleaning of raw data like removing punctuation, tokenizing, removing stopwords, stemming, lemmatization, vectorization, tagging, parsing, and more.

Pre-processing of raw data in NLP

Following are the basic steps which we need to perform while cleaning the raw data in NLP:

1. Remove Punctuation

2. Tokenization

3. Remove Stopwords

4. Stemming / Lemmatization

5. Vectorization

1. Remove Punctuation

First of all, we should remove all the punctuation marks (like comma, semicolon, colon, apostrophe, quotation marks, dash, hyphen, brackets, braces, parentheses, ellipsis etc.) from the text as these carry negligible weight.

2. Tokenization

Now create a list of words used in the text. Each word is called a token. We can use regular expression to find out tokens from the sentences otherwise NLTK has efficient modules for this task.

3. Remove Stopwords

Now we need to remove all the stopwords from the token list. Stopwords are the words which occur frequently in a sentence but carry little weight (like the, for, is, and, or, been, to, this, that, but, if, in, a, as etc.).

4.1: Stemming

It is used to reduce the number of tokens just like removing stopwords. In this process, we reduce inflected words to their word stem or root. We keep only the semantic meaning of similar words.

Examples

1) Tokens like **stemming** and **stemmed** are converted to a token **stem**.

2) Tokens like **working**, **worked**, **works** and **work** are converted to a token **work**.

Points 1 and 2 clearly illustrate that how can we reduce the number of tokens in a token list using stemming. But wait! There is a problem. Consider following examples of stemming:

3) Tokens like **meanness** and **meaning** are converted to a token **mean**. Now this is wrong. Both tokens have different meanings, even then its treating both as same.

4) Tokens like **goose** and **geese** are converted to the tokens **goos** and **gees** respectively (it will just remove "e" suffix from both the tokens). Now this is again wrong. "**geese**" is just a plural of "**goose**", even then its treating both tokens as different.

Points 3 and 4 can be resolved using Lemmatization.

NLTK library has 4 stemmers:

1) Porter Stemmer

2) Snowball Stemmer

3) Lancaster Stemmer

4) Regex-based Stemmer

I mainly use Porter stemmer for stemming the tokens in my NLP code.

4.2: Lemmatization

We saw the limitation of stemming in above examples (3 and 4). We can overcome these limitations using Lemmatization. It is more powerful and sophisticated as compared to stemming and returns more accurate and meaningful words / tokens by considering the context in which the word is used in a sentence.

But the tradeoff is that, it is slower and complex as compared to the stemming.

Examples

1) Tokens like **meanness** and **meaning** are retained as it is instead of reducing it to mean (unlike stemming).

2) Tokens like **goose** and **geese** are converted to a token **goose** which is right. We should get rid of the token "geese" as it is just a plural of "goose".

I mainly use **WordNet** Lemmatizer present in NLTK library.

5. Vectorization

Machine Learning algorithms don't understand text. These need numeric data for matrix multiplications. Till now, we have just cleaned our tokens. So, in this process, we encode our final tokens into numbers to create feature vectors so that algorithms can understand. In other words, we will fit and transform vectorization methods to our preprocessed and cleaned data which we created till lemmatization.

Document-term matrix: Let's first understand this term before proceeding further. We use document term matrix to represent the words in the text in the form of matrix of numbers. The rows of the matrix represent the text responses to be analyzed, and the columns of the matrix represent the words / tokens from the text that are to be used in the analysis.

Types of Vectorization

There are mainly 3 types of vectorization:

1) Count vectorization

2) N-grams vectorization

3) Term Frequency - Inverse Document Frequency (TF-IDF)

1) Count vectorization: It creates a document-term matrix which contains the count of each unique word / token in the text response.

2) N-grams vectorization: It creates a document-term matrix which also considers context of the word depending upon the value of N.

If N = 2, it is called bi-gram,

If N = 3, it is called tri-gram,

If N = 4, it is called four-gram and so on...

We need to be careful about value of N and choose it wisely.

Example: Consider a sentence "*NLP is awesome*". Count vectorization will create a column corresponding to each word in document-term matrix while N-gram will create columns like following in case of bi-gram:

"NLP is", "is awesome"

3) Term Frequency - Inverse Document Frequency (TF-IDF): It is just like count vectorization but instead of count, it stores weightage of each word by using following formula:

$$w_{i,j} = \mathrm{tf}_{\mathrm{i,j}} \times \log\left(\frac{\mathrm{N}}{\mathrm{df}_{\mathrm{i}}}\right)$$

w(i, j) = weightage of a particular word "i" in a document "j"

tf(i, j) = term frequency of a word "i" in document "j" i.e. number of times the word "i" occurs in a document "j" divided by total number of words in document "j"

N = number of total documents

df(i) = number of documents containing the word "i"

So, in this way, TF-IDF considers two facts while calculating the weightage of a word or token:

1) how frequent the word occurs in a particular document

2) and how frequent that word occurs in other documents

Example: Consider that we have 10 text messages and one of the text messages is "*NLP is awesome*". No other message contains the word "NLP". Now let's calculate weightage of the word NLP.

tf(i, j) = number of times the word NLP occurs in the text message divided by the total number of words in the text message. It comes out to be (1/3) as there are three words and NLP occurs only one time.

N = 10 as there are 10 text messages.

df(i) = number of text messages containing the word NLP which in our case is 1.

So, the final equation becomes:

Weightage of NLP = (1/3) * log(10/1)

In this way, we fill all the rows and column of document-term matrix in TF-IDF.

QUESTIONS

1. Which of the following is TRUE about NLP?

A. We must take care of Syntax, Semantics and Pragmatics in NLP

B. Preprocessing tasks include Tokenization, Stemming, Lemmatization and Vectorization

C. NLP can be used for spam filtering, sentiment analysis and machine translation

D. All of the above

2. Which of the following is NOT an application of NLP?

A. Personal assistants (like Alexa, Siri, Google Assistant, Cortana etc.)

B. Image recognition

C. Auto-correct grammatical mistakes

D. Named entity recognition

3. Which of the following is an application of NLP?

A. Google Translate

B. Google Assistant

C. Chatbots

D. All of the above

4. Which of the following is FALSE about Tokenization in NLP?

A. It is used to remove stopwords from the text

B. Each word in a text is called a token

C. We can use regular expressions to find out tokens from the text

D. None of the above

5. Which of the following is a correct order of preprocessing of raw data in NLP?

A. Remove Punctuation, Tokenization, Remove Stopwords, Stemming / Lemmatization, Vectorization

B. Tokenization, Vectorization, Remove Punctuation, Remove Stopwords, Stemming / Lemmatization

C. Stemming / Lemmatization, Remove Punctuation, Tokenization, Remove Stopwords, Vectorization

D. Remove Stopwords, Remove Punctuation, Stemming / Lemmatization, Vectorization, Tokenization

6. Which of the following is FALSE about preprocessing of raw data in NLP?

A. We remove stopwords and do stemming to decrease the number of tokens

B. We should remove all the punctuation marks and stopwords from the text

C. Lemmatization is a process of removing punctuation and stopwords from the text

D. Vectorization is used to encode tokens into numbers to create feature vectors

7. Which of the following is FALSE about Stemming and Lemmatization in NLP?

A. Lemmatization is more powerful and sophisticated as compared to stemming

B. Lemmatization is fast and but more complex as compared to the stemming

C. Both are used to reduce the inflected words to their word stem or root

D. Both are used to reduce the number of tokens

8. Which of the following is TRUE about Stemming and Lemmatization in NLP?

A. Stemming considers context of a word in which it is used in a sentence while Lemmatization does not

B. Stemming provides more accurate results as compared to Lemmatization

C. Stemming is faster than Lemmatization

D. All of the above

9. Which of the following is TRUE about Vectorization in NLP?

A. It is used to encode tokens into feature vectors which algorithms can understand

B. Document term matrix is used to represent the words in the text in the form of matrix of numbers

C. Count, N-gram and TF-IDF are the types of Vectorization

D. All of the above

10. Which of the following is TRUE about types of Vectorization in NLP?

A. Count vectorization considers count and weightage of each word in a text

B. N-gram vectorization considers context of the word depending upon the value of N

C. TF-IDF vectorization considers both count and context of the words in the text

D. All of the above

11. Which of the following is FALSE about Term Frequency - Inverse Document Frequency (TF-IDF) Vectorization in NLP?

A. It considers context of the word in a particular document

B. It considers how frequent the word occurs in a particular document

C. It considers how frequent the word occurs in other documents

D. None of the above

12. Which of the following is FALSE about NLTK library in Python?

A. It is the most commonly used library in Python for NLP

B. It provides efficient modules for preprocessing and cleaning of raw data in NLP

C. It can be used for tokenization, stemming, lemmatization and vectorization

D. All of the above

13. Which of the following stemmers are available in NLTK library in Python?

A. Porter Stemmer

B. Snowball Stemmer

C. Lancaster Stemmer

D. All of the above

ANSWERS

Answer 1: D

Answer 2: B

Answer 3: D

Answer 4: A

Answer 5: A

Answer 6: C

Answer 7: B

Answer 8: C

Answer 9: D

Answer 10: B

Answer 11: A

Answer 12: D

Answer 13: D

CHAPTER 14

DEEP LEARNING FRAMEWORKS

There are a lot of frameworks available to implement deep learning algorithms. You don't need to implement CNN, RNN, LSTM algorithms from scratch because these frameworks do this complex task for you very easily, quickly and efficiently. You just need to learn how to use these frameworks.

Some of the widely used frameworks available for deep learning are TensorFlow, PyTorch, Keras, Theano, CNTK, DL4J, Caffe, MXNet, Chainer, Sonnet, Swift, ONNX, Gluon, MATLAB, PaddlePaddle etc.

All of these frameworks are open source, efficient and support CUDA. Instead of writing hundreds of lines of code, we can use any of these frameworks and build deep learning model quickly.

QUESTIONS

1. Which of the following frameworks can be used to implement deep learning models?

A. TensorFlow

B. Keras

C. MXNet

D. All of the above

2. Which of the following frameworks can be used to implement deep learning models?

A. PyTorch

B. Caffe

C. Theano

D. All of the above

3. Which of the following frameworks can be used to implement deep learning models?

A. DL4J

B. CNTK

C. Chainer

D. All of the above

4. Which of the following frameworks can be used to implement deep learning models?

A. Sonnet

B. Gluon

C. Swift

D. All of the above

5. Which of the following frameworks can be used to implement deep learning models?

A. ONNX

B. PaddlePaddle

C. MATLAB

D. All of the above

ANSWERS

Answer 1: D

Answer 2: D

Answer 3: D

Answer 4: D

Answer 5: D

CHAPTER 15

TENSORFLOW

TensorFlow library was developed by the Google Brain Team for complex numeric calculations (like numpy). It relies on a lot of matrix multiplications. Later on, Google started using it as a library for deep learning. First stable version of TensorFlow appeared in 2017. It is an open source library under Apache Open Source license.

Tensors and TensorFlow

A tensor is a vector or matrix of n-dimensions that represents all types of data. We can say that a tensor is a collection of feature vectors (i.e. array) of n-dimensions. Tensors can be considered as dynamically sized multi-dimensional array. The shape of the data is the dimensionality of the matrix or array. One dimensional tensor is known as scalar.

“TensorFlow” has been derived from the operations which neural networks perform on the tensors. It’s literally a flow of tensors. Tensor goes inside, flows through various nodes in neural network, and then comes out. That is why, it is called TensorFlow.

TensorFlow is made up of two terms – Tensor and Flow: In TensorFlow, the term tensor refers to the representation of data as multi-dimensional array whereas the term flow refers to the series of operations that one performs on tensors.

Tensor Ranks: 0 (scaler), 1 (vector), 2 (matrix), 3 (3-tensor)......., n (n-tensor)

Why TensorFlow?

1. It provides both C++ and Python APIs.

2. It has faster compilation time as compared to other deep learning libraries like Keras and PyTorch.

3. It supports both CPU and GPU computing devices.

4. It is available on both mobile and desktop.

5. TensorFlow runtime is a cross platform library.

Versions of TensorFlow

1. TensorFlow with CPU support only

2. TensorFlow with GPU support

The model can be trained and used on GPUs as well as CPUs.

Google Colaboratory

It is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud. With Colaboratory you can write and execute code, save and share your analyses, and access powerful computing resources, all for free from your browser.

Graph (Computational graph)

Graph is made up of nodes and edges. Series of TensorFlow operations are arranged as nodes in the computational graph.

Nodes: Each nodes take 0 or more tensors as input and produces a tensor as output. Node carries the mathematical operation and produces an endpoints outputs.

Properties of a node: unique label (name), dimension (shape), data type (dtype)

Datatypes: float32, float64, int8, int16, int32, int64, uint8, string and bool

Types of nodes: constant, placeholder, Variable, SparseTensor

In case you have not provided the data type explicitly, TensorFlow will infer the type of the constant / variable from the initialized value.

Must initialize a variable in TensorFlow

Constants are initialized when you call tf.constant but variables are not initialized when you call tf.Variable.

To initialize all the variables in a TensorFlow program, you must explicitly call a special operation as below:

```
init = tf.global_variables_initializer()

sess.run(init)
```

Variables must be initialized before a graph is used for the first time.

TensorFlow variables are in-memory buffers that contain tensors, but unlike normal tensors that are only instantiated when a graph is run and are immediately deleted afterwards, variables survive across multiple executions of a graph.

Edges: Edges explain the input/output relationships between nodes. Edge of the nodes is the tensor, i.e. a way to populate the operation with data. Each operation is called an op node.

Executing a Graph

There are two steps involved while executing a graph:

1. Building a Computational Graph: Just create nodes and assign them operations.

2. Running a Computational Graph: We need to run the computational graph within a session. Session is also called TensorFlow Runtime. Session places the graph operations onto devices, such as CPUs or GPUs, and provides methods to execute them. A session encapsulates the control and state of the TensorFlow runtime i.e. it stores the information about the order in which all the operations will be performed and passes the result of already computed operation to the next operation in the pipeline.

Advantages of Computational Graph

1. Parallel execution: The operations assigned to different nodes of a computational graph can be performed in parallel, thus, providing a better performance in terms of computations.

Nodes and edges can be spread over several clusters of computers (in distributed manner).

2. Portability: The portability of the graph allows to preserve the computations for immediate or later use. The graph can be saved and executed in the future.

Pipeline and Batches

If you have a dataset of 50 GB, and your computer has only 16 GB of memory, then the machine will crash. In this situation, you need to build a TensorFlow pipeline. The pipeline will load the data in batches. Each batch will be pushed to the pipeline and be ready for the training. It allows you to use parallel computing. It means TensorFlow will train the model across multiple CPUs or GPUs.

Tip: Use Pipeline if you have a large dataset. Use Pandas for less than 10GB data.

Epoch

An epoch defines how many times you want the model to see the data. One epoch is counted when all your data is once forward and backward propagated through the entire neural network.

Let's take an example. Suppose you have a dataset with 1200 observations. You are going to use a pipeline and have created 3 batches containing 400 observations each. Now, it will take 3 iterations (1200 / 400) to completely propagate the data forward and backward through the neural network to complete one epoch.

Estimators

It is a TensorFlow API used to implement algorithms. We can import following estimators APIs to solve a lot of classification and regression problems.

```
tf.estimator.LinearRegressor
tf.estimator.LinearClassifier
tf.estimator.BoostedTreesRegressor
tf.estimator.BoostedTreesClassifier
tf.estimator.DNNClassifier
tf.estimator.DNNLinearCombinedClassifier
```

Estimators are used for creating computational graphs, initializing variables, training the model and saving checkpoint and logging files for Tensorboard. In order to user estimators we need to create feature columns and input functions.

Input functions are used for passing input data to the model for training and evaluation. Feature columns are specifications for how the model should interpret the input data.

TensorBoard

TensorBoard enables to monitor graphically and visually what TensorFlow is doing. TensorFlow is based on graph computation; it allows the developer to visualize the construction of the neural network with Tensorboard.

QUESTIONS

1. Which of the following is FALSE about TensorFlow?

A. TensorFlow is a computational framework used to build deep learning models

B. It is an open source library for numerical computation

C. It is high level framework like Keras to implement CNN, RNN, and LSTM etc.

D. It provides both C++ and Python APIs

2. Which of the following is FALSE about sessions and graphs in TensorFlow?

A. We must initialize all the variables and constants before running the session

B. All the computations happen within a session

C. Graph is composed of nodes and edges

D. We need to first build and then run the graph to execute all the operations

3. Which of the following is TRUE about sessions and graphs in TensorFlow?

A. Session is also called TensorFlow Runtime

B. Session places the graph operations onto devices, such as CPUs or GPUs

C. Computational Graph helps in parallel execution and portability

D. All of the above

4. Which of the following is TRUE about Tensors in TensorFlow?

A. A tensor is a vector or matrix of n-dimensions that represents all types of data

B. Tensors can be considered as dynamically sized multi-dimensional array

C. One dimensional tensor is known as scalar

D. All of the above

5. Which of the following is TRUE about Tensors in TensorFlow?

A. Tensor with rank zero is also called scaler

B. Tensor with rank one is also called matrix

C. Tensor with rank two is also called vector

D. All of the above

6. Which of the following is TRUE about Tensors in TensorFlow?

A. Tensor with rank two is also called matrix

B. Tensor with rank two is also called vector

C. Tensor with rank two is also called scaler

D. All of the above

7. Which of the following is TRUE about Tensors in TensorFlow?

A. Tensor with rank one is also called matrix

B. Tensor with rank one is also called vector

C. Tensor with rank one is also called scaler

D. All of the above

8. Which of the following is TRUE about Tensors in TensorFlow?

A. Tensor with rank zero is also called matrix

B. Tensor with rank zero is also called vector

C. Tensor with rank zero is also called scaler

D. All of the above

9. Which of the following MUST be initialized before running a graph in TensorFlow?

A. Variables

B. Placeholders

C. Constants

D. Sessions

10. Which of the following is a correct function to initialize variables in TensorFlow?

A. initialize_global_variables

B. global_variables_initializer

C. global_variable_initializer

D. initialize_global_variable

11. Which of the following is TRUE about TensorBoard?

A. It enables to monitor graphically and visually what TensorFlow is doing

B. It allows the developer to visualize the construction of the neural network

C. It also helps in logging and monitoring

D. All of the above

12. Which of the following is TRUE about Estimator in TensorFlow?

A. It is a TensorFlow API used to implement algorithms

B. It is used to create computational graphs, initializing variables, training the model etc.

C. In order to use estimators, we need to create feature columns and input functions

D. All of the above

ANSWERS

Answer 1: C

Answer 2: A

Answer 3: D

Answer 4: D

Answer 5: A

Answer 6: A

Answer 7: B

Answer 8: C

Answer 9: A

Answer 10: B

Answer 11: D

Answer 12: D

CHAPTER 16

KERAS: INTRODUCTION

Keras is a widely used deep learning framework to implement neural networks. It is very easy to understand and implement. It also has a large community support.

Below are some points which illustrate strengths and limitations of Keras framework:

Strengths of Keras

1. High level framework: Keras is an open source and high level deep learning framework, written in Python.

2. Supports multiple backends: Keras uses TensorFlow as backend by default. You can also configure it to use Theano or CNTK as backend. In future, it may support more low level frameworks as backend.

3. Cross-Platform: Keras can run on all major operating systems. It supports multiple devices and platforms like iOS, Android, Google Cloud, JVM, and Raspberry Pi etc.

4. Multiple CPU and GPU support: Keras is compatible with both CPU and GPU. You can switch to any mode as per your requirement.

5. Easy to understand and implement: Keras is very easy to understand and implement. You can easily implement complex neural networks like CNN, LSTM, and Autoencoders etc. with few lines of code. Keras is a wrapper around the complex low level frameworks like TensorFlow, Theano and CNTK. So, with Keras, you don't need to understand low level details about implementing neural networks. In this way, we can say that Keras is a boon for beginners.

6. Pre-trained models: Keras contains a lot of pre-trained neural network models for our general purpose requirements. For example, for image classification, we don't need to create a CNN model from scratch. We can fine-tune an existing and well trained **VGG16** model for this purpose. Similarly, there are a lot of other pre-trained models available for image classification with Keras like **InceptionV3**, **ResNet50**, **MobileNet**, **Xception**, **InceptionResNetV2** etc. which we just need to fine-tune as per our requirement.

7. Great community: Keras has a great community support. You can easily find a lot of tutorials, detailed articles on various concepts, solved examples and a lot more. Keras is also very well documented. For detailed documentation of Keras, please visit their official website (https://keras.io/).

Limitations of Keras

As stated in point 1 and 2, Keras is only a high level API which uses other frameworks like TensorFlow, Theano and CNTK to perform low level tasks. If you want to research or design your own custom neural network in deep learning, you should not use Keras as it provides only limited choices. For this purpose, you need to use low level frameworks like TensorFlow, PyTorch, CNTK, Theano etc.

QUESTIONS

1. Which of the following is FALSE about Keras?

A. Keras is an open source and cross-platform framework

B. Keras is a low level framework like TensorFlow, PyTorch, Theano and CNTK

C. Keras is multiple CPU and GPU compatible and has a great community support

D. None of the above

2. In which of the following situations, you should NOT prefer Keras over TensorFlow?

A. When you want to quickly build prototype using neural networks

B. When you want to implement simple neural networks in your initial learning phase

C. When you are doing a critical and intensive research on any field

D. When you want to create simple tutorials for your students and subscribers

3. Which of the following neural network layers are supported by Keras?

A. Convolutional layers (Conv2D)

B. Recurrent layers (LSTM)

C. Dense layers

D. All of the above

ANSWERS

Answer 1: B

Answer 2: C

Answer 3: D

CHAPTER 17

KERAS: BACKEND

1. Which of the following is TRUE about backend in Keras?

A. Keras can use TensorFlow, Theano and CNTK as backend

B. By default, CNTK is used as backend

C. CNTK is a widely used backend framework in Keras

D. None of the above

2. Which of the following frameworks can be used as backend in Keras?

A. PyTorch

B. Caffe

C. DL4J

D. None of the above

3. Which of the following frameworks can be used as backend in Keras?

A. PyTorch

B. CNTK

C. DL4J

D. MXNet

4. Which of the following frameworks can be used as backend in Keras?

A. Theano

B. Sonnet

C. Chainer

D. Caffe

5. Which of the following frameworks can be used as backend in Keras?

A. Swift

B. Gluon

C. TensorFlow

D. MATLAB

6. Which of the following frameworks is used as backend in Keras by default?

A. TensorFlow

B. CNTK

C. Theano

D. MXNet

7. Which of the following frameworks CANNOT be used as backend in Keras?

A. TensorFlow

B. Theano

C. PyTorch

D. CNTK

ANSWERS

Answer 1: A

Answer 2: D

Answer 3: B

Answer 4: A

Answer 5: C

Answer 6: A

Answer 7: C

CHAPTER 18

KERAS: FINE-TUNE A MODEL

Keras contains a lot of pre-trained models which we can fine-tune as per our requirements.

Fine Tuning

Suppose you already have an efficient deep learning model which performs task A. Now you have to perform a task B which is quite similar to task A. If you use Keras, you don't need to create a separate model from scratch for task B. Just fine-tune the existing model which is efficiently performing task A.

Example: You have a well-trained model which identifies all types of cars. Car model has already learned a lot of features like edges, shapes, textures, head lights, door handles, wheels, windshield etc. Now you have to create a model which can identify trucks. We know that many features of cars and trucks are similar. So, why to create a new model for trucks from scratch? Let's just fine-tune the existing car model to create a new model for truck.

Transfer Learning

Transfer learning and fine-tuning terms are very similar in many ways and widely used almost interchangeably. We can transfer the learning from the existing model on cars to the new model on trucks. So, transfer learning happens while fine-tuning an existing model.

Advantages of Transfer Learning and Fine Tuning

Creating a new model is a very tough and time consuming task. We need to decide a lot of things while creating a model like:

1. Different types of layers to use (fully connected, convoluted, capsule, LSTM etc.)

2. How many layers to use?

3. How many nodes in a layer?

4. Which activation functions to use in which layer?

5. Which regularization techniques to use?

6. Which optimizer to use?

7. Tuning various hyperparameters like initializing weights, learning rate, momentum, batch size, number of epochs etc.

So, if we can fine-tune an existing model, we can very well escape from above tasks and save our time and energy.

How to fine-tune a model?

We need to make some reasonable changes and tweaks to our existing model to create a new model. Below are some basic steps to fine-tune an existing model:

1. Remove the output layer: First of all, remove the output layer which was identifying cars. Add a new output layer which will now identify trucks.

2. Add and remove hidden layers: Trucks have some features different from cars. So accordingly, add some hidden layers which will learn new features of trucks. Remove those hidden layers which are not required in case of trucks.

3. Freeze the unchanged layers: Freeze the layers which are maintained (not changed) so that no weight update happens on these layers when we again train our model on the new data with trucks. Weight should only be updated for newly added hidden layers.

QUESTIONS

1. Which of the following is FALSE about fine-tuning a model in Keras?

A. Keras provides a lot of pre-trained models which we can fine-tune as per our needs

B. Some of the pre-trained models are VGG16, InceptionV3, ResNet, MobileNet, Xception etc**.**

C. VGG16 and MobileNet pre-trained models can be fine-tuned for NLP

D. InceptionResNetV2 pre-trained model can be fine-tuned for image classification

2. Which of the following pre-trained models in Keras can be fine-tuned for image classification?

A. VGG16

B. MobileNet

C. InceptionResNetV2

D. All of the above

3. Which of the following steps are usually performed while fine-tuning a model?

A. We freeze the layers for which we do not want to update weights and biases

B. We remove existing output layer and create a new output layer as per our requirement

C. We add/remove hidden layers as per our requirement

D. All of the above

4. We freeze some hidden layers while fine-tuning a model so that:

A. weights and biases do not update

B. gradients do not vanish

C. gradients do not explode

D. accuracy and performance of the model is improved

5. Which of the following is FALSE about ImageNet?

A. It is a pre-trained model for image recognition in Keras

B. It is a dataset with millions of labelled images of different categories

C. It runs an annual image recognition competition called ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

D. None of the above

ANSWERS

Answer 1: C

Answer 2: D

Answer 3: D

Answer 4: A

Answer 5: A

CHAPTER 19

KERAS: IMPLEMENTATION

1. In Keras, we set the “input_dim” parameter in which layer of the neural network?

A. Input layer

B. Hidden layers

C. Dropout layer

D. Output layer

2. Which of the following functions is used to train a Keras Sequential model?

A. model.compile

B. model.fit

C. model.evaluate

D. model.predict

3. Which of the following functions is used to test a Keras Sequential model?

A. model.compile

B. model.fit

C. model.evaluate

D. model.predict

4. Which of the following functions is used to make predictions in Keras Sequential model?

A. model.compile

B. model.predict

C. model.evaluate

D. model.fit

5. Which of the following parameters is not required while compiling a model in Keras?

A. activation function

B. loss

C. metrics

D. optimizer

6. Which of the following parameters is required while training a model in Keras?

A. activation function

B. number of epochs

C. regularization method

D. weight initialization method

7. Which of the following parameters is not required while creating a layer in Keras?

A. activation function

B. number of epochs

C. regularization method

D. weight initialization method

8. “ImageDataGenerator” class in Keras help in:

A. Image Augmentation

B. Image Captioning

C. Increasing image resolution

D. All of the above

9. Which of the following is a correct function to load image in Keras?

A. load_image

B. load_img

C. img_load

D. image_load

10. Which of the following is a correct function to convert image into numpy array in Keras?

A. img_to_numpy_array

B. image_to_numpy_array

C. img_to_array

D. image_to_array

11. Which of the following modules contains pre-trained models in Keras?

A. keras.preprocessing

B. keras.applications

C. keras.utils

D. keras.layers

12. HDF5 file in Keras contains:

A. Architecture and weights of the model

B. Training configuration like loss function and optimizer

C. State of the optimizer (to resume the training from where you left)

D. All of the above

13. Which of the following is FALSE about saving a model in Keras?

A. It is recommended to use pickle or cPickle to save a Keras model

B. We can save both model architecture and weights as h5 file

C. We can also save model architecture separately as JSON file

D. We can save model weights separately as h5 file

ANSWERS

Answer 1: A

Answer 2: B

Answer 3: C

Answer 4: B

Answer 5: A

Answer 6: B

Answer 7: B

Answer 8: A

Answer 9: B

Answer 10: C

Answer 11: B

Answer 12: D

Answer 13: A

CHAPTER 20

INTERVIEW QUESTIONS

We have covered a lot of objective type questions on deep learning. Let's look at some deep learning interview questions. Following list contains 89 questions divided in different categories. You can get detailed answers to these questions from my blog (http://theprofessionalspoint.blogspot.com/2019/06/100-basic-deep-learning-interview.html).

Introduction

1. What is Deep Learning? How is it different from machine learning? What are the pros and cons of deep learning over machine learning?

2. How does deep learning mimic the behavior of human brain? How will you compare an artificial neuron to a biological neuron?

Perceptron

3. What is a Perceptron? How does it work? What is a multi-layer perceptron?

4. What are the various limitations of a Perceptron? Why cannot we implement XOR gate using Perceptron?

Neural Networks

5. What are the various layers in a neural network?

6. What are the various types of a neural network?

7. What are Deep and Shallow neural networks? What are the advantages and disadvantages of deep neural networks over shallow neural networks?

Weights and Bias

8. What is the importance of weights and biases in a neural network? What are the things to keep in mind while initializing weights and biases?

9. What is Xavier Weight Initialization technique? How is it helpful in initializing the weights? How does weight initialization vary for different types of activation functions?

10. Explain forward and backward propagation in a neural network. How does a neural network update weights and biases during back propagation?

Activation Functions

11. What do you mean by activation functions in neural networks? Why do we call them squashing functions? How do activation functions bring non-linearity in neural networks?

12. Explain various activation functions like Step (Threshold), Logistic (Sigmoid), Hyperbolic Tangent (Tanh), and ReLU (Rectified Linear Unit). What are the various advantages and disadvantages of using these activation functions?

13. Dying and Leaky ReLU: What do you mean by Dying ReLU? When a neuron is considered as dead in a neural network? How does leaky ReLU help in dealing with dying ReLU?

14. What is the difference between Sigmoid and Softmax activation functions?

Batches

15. Explain the terms: Epochs, Batches and Iterations in neural networks.

16. What do you mean by Batch Normalization? What are its various advantages?

Loss Function

17. What is the difference between categorical_crossentropy and sparse_categorical_crossentropy? Which one to use and when?

Gradient Descent

18. What is Gradient Descent? How is it helpful in minimizing the loss function? What are its various types?

19. Explain Batch, Stochastic, and Mini Batch Gradient Descent. What are the advantages and disadvantages of these Gradient Descent methods?

20. Explain these terms in context of SGD: Momentum, Nesterov Momentum, AdaGrad, AdaDelta, RMSprop, Adam.

21. What is the difference between Local and Global Minima? What are the ways to avoid local minima?

22. Explain Vanishing and Exploding Gradients.

23. What is Learning Rate? How does low and high learning rate affect the performance and accuracy of a neural network?

24. If loss in a neural network is not decreasing during training period after so many iterations, what could be the possible reasons?

CNN (ConvNets)

25. What is Convolutional Neural Network? Explain various layers in a CNN?

26. What are the Filters (Kernels) in CNN? What is Stride?

27. What do you mean by Padding in CNN? What is the difference between Zero Padding and Valid Padding?

28. What do you mean by Pooling in CNN? What are the various types of pooling? Explain Max Pooling, Min Pooling, Average Pooling and Sum Pooling.

29. What are the various hyperparameters in CNN which need to be tuned while training process?

30. How is CNN different from traditional fully connected neural networks? Why we cannot use fully connected neural networks for image recognition?

31. Suppose we have an input of **n X n** dimension and filter of **f X f** dimension. If we slide this filter over the input in the convolutional layer, what will be the dimension of the resulting output?

CapsNets

32. What is Capsule Neural Network (CapsNets)? How is it different from CNN (ConvNets)?

Computer Vision

33. What is computer vision? How does deep learning help in solving various computer vision problems?

RNN

34. Explain RNN (Recurrent Neural Network). Why RNN is best suited for sequential data?

35. What do you mean by feedback loop in RNN?

36. What are the various types of RNN? Explain with example: One to One, One to Many, Many to One, and Many to Many RNN.

37. What is Bidirectional RNN?

38. What are the various issues with RNN? Explain Vanishing and Exploding Gradients. What are the various ways to solve these gradient issues in RNN?

39. What are the various advantages and disadvantages of RNN?

40. What are the various applications of RNN?

41. What are the differences between CNN and RNN?

LSTM

42. How does LSTM (Long Short Term Memory) solve Vanishing Gradient issue in RNN?

43. What are the gated cells in LSTM? What are the various types of gates used in LSTM?

44. What are the various applications of LSTM?

Regularization

45. What are the main causes of overfitting and underfitting in a neural network?

46. What are the various regularization techniques used in a neural network?

47. Explain L1 and L2 Regularization techniques used in a neural network.

48. What is Dropout? How does it prevent overfitting in a neural network? What are its various advantages and disadvantages?

49. What is Data Augmentation? How does it prevent overfitting in a neural network?

50. What is Early Stopping? How does it prevent overfitting in a neural network?

Learnable Parameters and Hyperparameters

51. What are the learnable parameters in a neural network? Explain with an example.

52. What are the various hyperparameters used in a neural network? What are the various ways to optimize these hyper-parameters?

53. How will you manually calculate number of weights and biases in a fully connected neural network? Explain with an example.

54. How will you manually calculate number of weights and biases in a convolutional neural network (CNN)? Explain with an example.

Transfer Learning

55. What do you mean by Transfer Learning and Fine-tuning a model? What are its various advantages? What are the various steps to fine-tune a model?

Autoencoders

56. What are Autoencoders? What are the various components of an autoencoder? Explain encoder, decoder and bottleneck. How does an autoencoder work?

57. What do you mean by latent space representation and reconstruction loss in an autoencoder?

58. What are the various properties of an autoencoder?

59. What are the various types of an autoencoder? Explain Undercomplete autoencoder, Sparse autoencoder, Denoising autoencoder, Convolutional autoencoder, Contractive autoencoders and Deep autoencoders.

60. How do we add regularization capabilities to autoencoders?

61. What are the various applications of an autoencoder?

62. What are the various hyperparameters we need to tune in an autoencoder?

63. How will you compare Autoencoders with PCA (Principal Component Analysis)?

64. What is RBM (Restricted Boltzman Machine)? What is the difference between an Autoencoder and RBM?

NLP (Natural Language Processing)

65. What are the various steps involved in preprocessing of the raw data in NLP? Explain Tokenization, Stemming, Lemmatization and Vectorization.

66. What is the difference between Stemming and Lemmatization?

67. What are the various types of Vectorization? Explain Count Vectorization, N-grams Vectorization and Term Frequency - Inverse Document Frequency (TF-IDF).

Frameworks

68. What are the various frameworks available to implement deep learning models? What should be the characteristics of an ideal deep learning framework?

TensorFlow

69. Explain TensorFlow architecture.

70. What is a Tensor? Explain Tensor Datatypes and Ranks.

71. What are Constants, Placeholders and Variables in a TensorFlow? Why do we need to initialize variables explicitly?

72. What is a Computational Graph? What are the nodes and edges in it? How to build and run the graph using session? What are its various advantages?

73. What is a Tensor Board? How is it useful?

74. What is a TensorFlow Pipeline? How is it useful?

75. Explain these terms: Feed Dictionary and Estimators.

76. Write a sample code to demonstrate constants, placeholders and variables in TensorFlow?

77. Write a sample code using TensorFlow to demonstrate gradient descent?

78. Implement a Linear Classification Model using TensorFlow Estimator.

Keras

79. What do you know about Keras framework? What are its various advantages and limitations?

80. How will you build a basic sequential model using Keras?

81. How will you solve a regression problem using sequential model in Keras?

82. How will you build a basic CNN model using Keras?

83. How will you build a basic LSTM model using Keras?

84. What are the various pre-trained models available in Keras? How are these pre-trained models useful for us?

85. How will you use VGG16 model to recognize a given image?

86. How will you fine-tune VGG16 model for image classification?

87. How will you fine-tune MobileNet model for image classification? What is the difference between VGG16 and MobileNet model?

Google Cloud Vision API

88. How will you use Google Cloud Vision API for Label Detection?

89. How will you use Google Cloud Vision API for Text Detection (OCR: Optical Character Recognition)?