studocu

DL decode

BE Information Technology (Savitribai Phule Pune University)

Scan to open on Studocu

Studocu is not sponsored or endorsed by any college or university
Downloaded by ribhav soni (ribhavsoni19@gmail.com)

Unit III

3 Recurrent Neural Networks

3.1 : Basics of Recurrent Neural Networks

Q.1 What is recurrent neural networks ?

Ans. : • A class of neural networks for processing sequential data is known as recurrent neural networks (RNN).

• RNN are neural networks that are specialized for processing a series of values x(1),..., x (τ), just like convolutional networks are specialized for processing a grid of values X, such as an image.

• RNNs are designed to recognize the sequential characteristics in data and use patterns to predict the next likely scenario. Unlike other neural networks, an RNN has an internal memory that enables it to remember historical input; this allows it to make decisions by considering current input alongside learning from previous input.

Q.2 Explain unfolding computational graphs.

Ans. : • The structure of a number of calculations, such as those involved in mapping inputs and parameters to outputs and loss, can be formalized using a computational graph. The concept of unfolding a recursive or recurrent computation into a computational network with a repeated structure, often corresponding to a series of occurrences, is explained in this section. The sharing of parameters across a deep network structure is the outcome of unfolding this graph.

• Take the traditional form of a dynamical system, for instance :

$$s^{(t)} = f(s^{(t-1)}; \theta) \quad \ldots(Q.2.1)$$

where s(t)is called the state of the system.

This document is available on studocu
Downloaded by ribhav soni (ribhavsoni19@gmail.com)

• Because the definition of s at time t goes back to the identical definition at time t – 1, equation (Q.2.1) is recurring.

• The graph can be unfolded for a limited number of time steps τ by using the definition τ –1 times. For example, if we unfold equation (Q.2.1) for τ = 3 time steps, we obtain

$$s^{(3)} = f(s^{(2)}; \theta) \quad \ldots(Q.2.2)$$

$$= f(f(s^{(1)}; \theta); \theta) \quad \ldots(Q.2.3)$$

• By continually using the definition in this manner to unfold the equation, an expression that does not involve recurrence has been produced. In the present, a conventional directed acyclic computational network can represent such an expression. Fig. Q.2.1 shows the unfolded computational graph of equations (Q.2.1) and (Q.2.3).

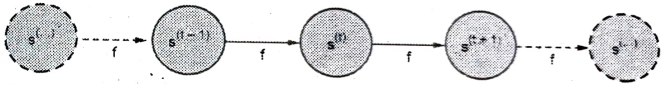


Fig. Q.2.1 A computational graph that has been unfurled serves as an illustration of the classical dynamical system given by equation (Q.2.1)

• The state at each node at time t is represented and the function f translates the state at time t to the state at time t + 1. For each time step, the same parameters (i.e., the same value of used to parameterize f) are applied.

• As another example, let us consider a dynamical system driven by an external signal $x^{(t)}$,

$$s^{(t)} = f(s^{(t-1)}, x^{(t)}; \theta) \quad \ldots(Q.2.4)$$

where we see that the state now contains information about the whole past sequence.

• There are several methods for constructing recurrent neural networks. Any function that involves recurrence may be seen as a

recurrent neural network, much like practically any function can be regarded as a feedforward neural network.

- Equation (Q.2.5) or a related equation is frequently used by recurrent neural networks to specify the values of its hidden units. We now rewrite equation (Q.2.4) using the variable h as the state to show that the state is the network's hidden units :

$$h^{(t)} = f h(h^{(t-1)}, x^{(t)}; \theta) \quad \ldots(Q.2.5)$$

- Typical RNNs will include additional architectural features, like output layers that read data from the state h to make predictions, as shown in Fig. Q.2.2.
- The recurrent network often learns to utilize $h^{(t)}$ as a type of lossy summary of the task-relevant elements of the previous sequence of inputs up to t when it is trained to execute a task that involves forecasting the future from the past. Since it converts an arbitrary length sequence $(x^{(t)}, x^{(t1)}, x^{(t2)}, \ldots, x^{(2)}, x^{(1)})$ to a fixed length vector h, this summary is inherently lossy. This summary may retain some former sequence elements with greater precision than others depending on the training criterion. For instance, it might not be necessary to store all of the data in the input sequence up to time t, only enough to predict the rest of the sentence if the RNN is used in statistical language modelling, which typically predicts the next word given previous words.
- The circumstance when we require $h^{(t)}$ to be rich enough to allow one to roughly recover the input sequence, like in autoencoder systems, is the most challenging.

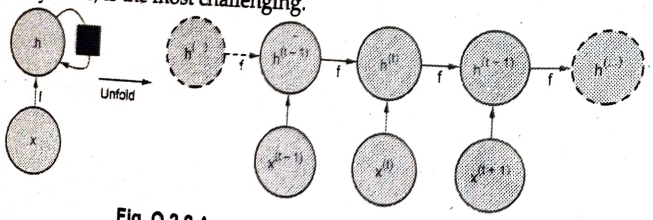


Fig. Q.2.2 An outputless recurrent network

- Simply by combining information from the input x into the state h that is transmitted forward over time, this recurrent network processes information from the input x. Circuit schematic (left). A onetime step delay is shown by the black square. (Right) The same network as a computational graph that has been unfolded, where each node is now connected to a specific time occurrence.

Q.3 List the advantages of unfolding process.

Ans. : Advantages :

1. Regardless of sequence length, learned model has same input size. Because it is specified in terms of transition from one state to another state rather than specified in terms of a variable length history of states
2. Possible to use same function f with same parameters at every step.

Q.4 Explain architecture of recurrent neural network.

Ans. : • Fig. Q.4.1 shows architecture of recurrent neural network.

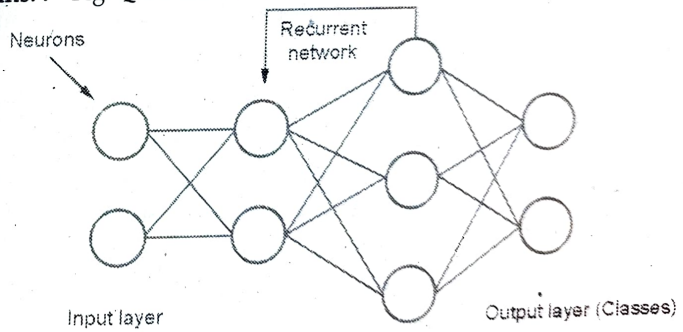


Fig. Q.4.1 Architecture of recurrent neural network

- A recurrent neural network is a class of artificial neural networks that contain a network like series of nodes, each with a directed or one-way connection to every other node. These nodes can be classified as either input, output, or hidden. Input nodes receive

Downloaded by ribhav soni (ribhavsoni19@gmail.com)

data from outside of the network, hidden nodes modify the input data and output nodes provide the intended results.

- RNNs can be used in the cases where, for example, next word to place in a sentence can be predicted. Here, the information regarding the previous words is needed to predict the next word in the sentence. RNN can solve this issue with the help of hidden layers, which is one of the most important and unique features about the RNN. These hidden layers remember the previous information to be used in the next layers.
- The recurrent neural network scans through the data from left to right. RNNs are very powerful, because they combine two properties :
  1. Distributed hidden state that allows them to store a lot of information about the past efficiently.
  2. Non-linear dynamics that allows them to update their hidden state in complicated ways.
- With enough neurons and time, RNNs can compute anything that can be computed by your computer.

Q.5 How to compute the gradient in a recurrent neural network ?

Ans. : • It is simple to calculate the gradient using a recurrent neural network. The unrolled computational network is simply subjected to the generalized back-propagation method. No particular algorithms are required. The back-propagation through time (BPTT) technique applies back-propagation to the unrolled graph. Then, to train an RNN, gradients generated from back-propagation can be employed with any general-purpose gradient-based approach.

- We give an example of how to compute gradients via BPTT for the aforementioned RNN equations in order to give the reader an understanding of how the BPTT method functions (Equation

studocu This document is available on Downloaded by ribhav soni (ribhavsoni10@gmail.com)

(Q.2.1) and equation (Q.2.4)). The parameters U, V, W, b and c, as well as the series of nodes indexed by t for $x^{(t)}$, $h^{(t)}$, $o^{(t)}$ and $L^{(t)}$ are all nodes in our computational graph. Based on the gradient calculated at nodes that follow it in the graph, we must recursively compute the gradient ∇NL for each node N.

- We start the recursion with the nodes immediately preceding the final loss

$$\frac{\partial L}{\partial L^{(t)}} = 1 \qquad \ldots(Q.5.1)$$

- In this derivation, we assume that the vector $\hat{y}$ of probabilities over the output is obtained by passing the outputs $o^{(t)}$ as the argument to the softmax function. Additionally, we presume that given the current input, the loss is the negative log-likelihood of the real goal $y^{(t)}$.
- The gradient $\nabla o^{(t)} L$ on the outputs at time step t, for all i, t, is as follows :

$$(\nabla o^{(t)} L)_i = \frac{\partial L}{\partial o_i^{(t)}} = \frac{\partial L}{\partial L^{(t)}} \frac{\partial L^{(t)}}{\partial o_i^{(t)}} = \hat{y}_i^{(t)} - 1_{i,y^{(t)}} \qquad \ldots(Q.5.2)$$

- Starting at the conclusion of the series, we work our way backward. At the final time step τ, $h^{(\tau)}$ only has $o^{(\tau)}$ as a descendent, so its gradient is simple :

$$\nabla h^{(T)} L = V^T \nabla o^{(T)} L \qquad \ldots(Q.5.3)$$

- We can then iterate backwards in time to back-propagate gradients through time, from $t = \tau - 1$ down to $t = 1$, noting that $h^{(t)}$ (for $t < \tau$) has as descendants both $o^{(t)}$ and $h^{(t+1)}$.
- Its gradient is thus given by,

$$\nabla h^{(t)} = \left(\frac{\partial h^{(t+1)}}{\partial h^{(t)}}\right) (\nabla h^{(t+1)} L) + \left(\frac{\partial o^{(t)}}{\partial h^{(t)}}\right) T(\nabla o^{(t)} L) \qquad \ldots(Q.5.4)$$

$$= W^T (\nabla h^{(t+1)} L) \operatorname{diag}(1 - h^{(t+1)})^2) + V^T (\nabla o^{(t)} L) \quad \ldots(Q.5.5)$$

where diag $(1 - (h^{(t+1)2}))$ indicates the diagonal matrix containing the elements $1 - (h_i^{(t+1)})^2$. This is the Jacobian of the hyperbolic tangent associated with the hidden unit i at time t + 1.

- We may acquire the gradients on the parameter nodes once we have the gradients on the internal nodes of the computational network. We must be careful when designating calculus operations using these variables since they are shared over several time steps. The bprop approach, which computes the contribution of a single edge in the computational graph to the gradient, is used in the equations we want to employ.
- The calculus ∇W f operator, on the other hand, accounts for the contribution of W to the value of f resulting from each edge in the computational graph. In order to clear up this uncertainty, we construct dummy variables W(t), which are duplicates of W that are only utilized at time step t. The weights' contribution to the gradient at time step t is then shown by the symbol $\nabla W(t)$.
- The gradient on the remaining parameters is represented by the following notation:

$$\nabla c L = \sum_t \left(\frac{\partial o^{(t)}}{\partial c}\right)^T \nabla o^{(t)} L = \sum_t \nabla o^{(t)} L \quad \ldots(Q.5.6)$$

$$\nabla b L = \sum_t \left(\frac{\partial h^{(t)}}{\partial b^{(t)}}\right)^T \nabla h^{(t)} L = \sum_t \operatorname{diag}(1 - h^{(t)})^2) \nabla h^{(t)} L \quad \ldots(Q.5.7)$$

$$\nabla W L = \sum_t \sum_i \left(\frac{\partial L}{\partial o_i^{(t)}}\right) \nabla V o_i^{(t)} = \sum_t \nabla o^{(t)} L h(t)^T \quad \ldots(Q.5.8)$$

$$\nabla W L = \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}}\right) \nabla W^{(t)} h_i^{(t)} \quad \ldots(Q.5.9)$$

$$= \sum_t \operatorname{diag}(1 - (h^{(t)})^2) (\nabla h^{(t)} L) h^{(t-1)T} \quad \ldots(Q.5.10)$$

$$\nabla U L = \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}}\right) \nabla U^{(t)} h_i^{(t)} \quad \ldots(Q.5.11)$$

$$= \sum_t \operatorname{diag}(1 - (h^{(t)})^2) (\nabla h^{(t)} L) x^{(t)T} \quad \ldots(Q.5.12)$$

- Since it has no parameters as ancestors in the computational graph defining the loss, we do not need to compute the gradient with respect to x(t) for training.

Q.6 What is a directed graphical model in RNN ?

Ans. : • Cross-entropies between training objectives $y^{(t)}$ and outputs o were the losses $L^{(t)}$ in the example recurrent network we have created so far $o^{(t)}$. In theory, practically any loss may be used with a recurrent network, much like with a feedforward network. The job should guide the loss selection.

- We often want to interpret the output of the RNN as a probability distribution, similar to how we would with a feedforward network and we typically use the cross-entropy of that distribution to quantify the loss.
- The cross-entropy loss associated, for instance, with a feedforward network and a unit Gaussian output distribution is called mean squared error.
- When we choose a training target for predictive log-likelihood, as equation (Q.2.4). Using the previous inputs, we train the RNN to estimate the conditional distribution of the following sequence element, y(t).
- This may mean that we maximize the log-likelihood

$$\log p(y^{(t)} | x^{(1)}, \ldots, x^{(t)}) \quad \ldots(Q.6.1)$$

or, if the model includes connections from the output at one time step to the next time step,

$$\log p(y^{(t)} | x^{(1)}, \ldots, x^{(t)}, y^{(1)}, \ldots, y^{(t-1)}) \quad \ldots(Q.6.2)$$

Downloaded by ribhav soni (ribhavsoni19@gmail.com)

- One method to capture the whole joint distribution across the entire sequence is to decompose the joint probability over the sequence of y values into a series of one-step probabilistic predictions. The directed graphical model does not contain any edges from any $y^{(i)}$ in the past to any $y^{(t)}$ in the present when we do not feed previous y values as inputs that condition the next step prediction. Given the sequence of x values in this instance, the outputs y are conditionally independent. The directed graphical model has edges from all previous $y^{(i)}$ values to the present $y^{(t)}$ value when we feed the network the real y values (not their prediction, but the actual observed or created values) back.
- Fig. Q.6.1 A fully connected graphical model for a sequence of values $y^{(1)}, y^{(2)}, \ldots, y^{(t)}, \ldots$ shows that each prior observation $y^{(t)}$ may have an impact on the conditional distribution of some $y^{(t)}$ (for $t > I$ given the values from before. It may be exceedingly wasteful to parameterize the graphical model directly using this graph (as in Equation 6), as there are an increasing number of inputs and parameters for each member of the series. As seen in Fig. Q.6.2, RNNs achieve the same complete connectivity but with efficient parameterization.

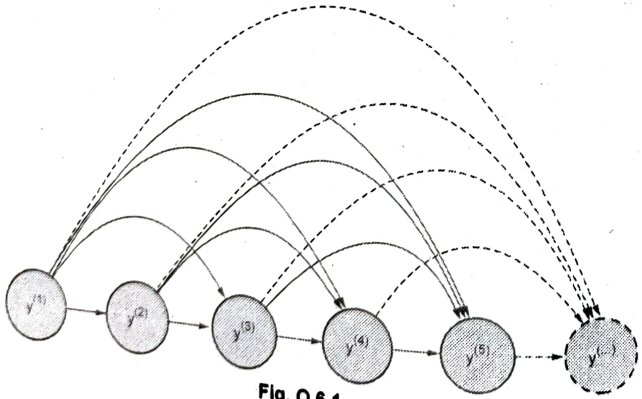


Fig. Q.6.1

This document is available on

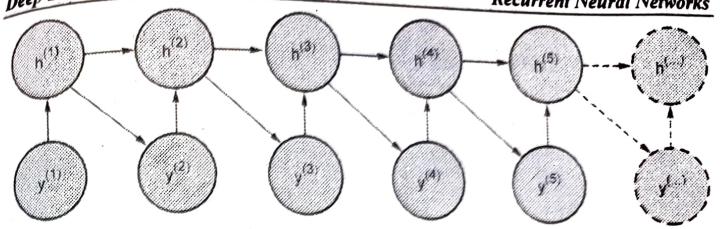


Fig. Q.6.2

- Let's take the scenario when the RNN models merely a series of scalar random variables $Y = \{y^{(1)}, \ldots, y^{(\tau)}\}$ with no extra inputs x as a basic illustration. Simply expressed, the output at time step t1 is the input at time step t. The directed graphical model over the y variables is thus defined by the RNN. Using the chain rule for conditional probabilities, we parameterize the joint distribution of these observations :

$$P(Y) = P(y^{(1)}, \ldots, y^{(T)})$$

$$= \prod_{t=1}^{T} P(y^{(t)} | y^{(t-1)}, y^{(t-2)}, \ldots, y^{(1)}) \quad \ldots(Q.6.3)$$

- Thus, naturally, for t = 1, the right-hand side of the bar is empty. Because of this, the negative log-likelihood of a set of values $y^{(1)}, \ldots$ and $y^{(T)}$ in this model is

$$L = \sum_{t} L^{(t)} \quad \ldots(Q.6.4)$$

Where

$$L(t) = -\log P(y^{(t)} = y | y^{(t-1)}, y^{(t-2)}, \ldots, y^{(1)}) \quad \ldots(Q.6.5)$$

Q.7 Explain advantages and disadvantages of RNN.

Ans. : Advantages :

a) RNN can process inputs of any length.

b) RNN model is modeled to remember each information throughout the time which is very helpful in any time series predictor.

studocu
Downloaded by ribhav soni (ribhavsoni19@gmail.com)

c) Even if the input size is larger, the model size does not increase.

d) The weights can be shared across the time steps.

e) RNN can use their internal memory for processing the arbitrary series of inputs which is not the case with feedforward neural networks.

Disadvantages :

a) Due to its recurrent nature, the computation is slow.

b) Training of RNN models can be difficult.

c) Prone to problems such as exploding and gradient vanishing.

Q.8 Why RNN is called as recurrent ?

Ans. : RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being dependent on the previous computations and user already know that they have a "memory" which captures information about what has been calculated so far.

Q.9 What is difference between RNN and CNN ?

Ans. :

Sr. No.	RNN	CNN
1.	RNN is applicable for time series and sequential data.	CNN is applicable for sparse data like images.
2.	RNN can have no restriction in length of inputs and outputs.	CNN has finite inputs and finite outputs.
3.	RNN is primarily used for speech and text analysis.	CNN can be used for video and image processing.
4.	RNN works on loops to handle sequential data.	CNN has a feedforward network.
5.	While training the model, CNN uses a simple back-propagation.	While training the model, RNN uses back-propagation through time to calculate the loss.

3.2 : Types of Recurrent Neural Networks

Q.10 Explain types of recurrent neural networks.

Ans. : 1. One-to-one : This neural network is used for fixed sized input to fixed sized output for example image classification. This was formerly known as Vanilla RNN, usually characterized by a single variety of input, such as a word or image. At the same time, the outputs are produced as a single token value. All traditional neural networks fall into this category.

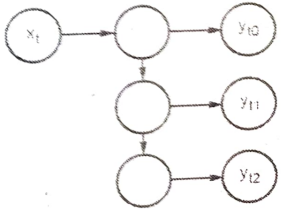
2. **One-to-many :** A single input is used to create multiple outputs. A popular application for one to many is music generation.

3. **Many-to-one :** Consists of several inputs that used to create a single output. An example is sentiment analysis. Input is a movie's review (multiple words in input) and output is sentiment associated with the review.

4. **Many-to-many :** Several inputs are used for generating several outputs. Name entity recognition is a famous example of this category.

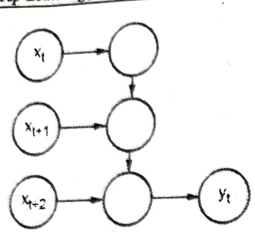
- Fig. Q.10.1 shows types of RNN.

(a) One-to-one

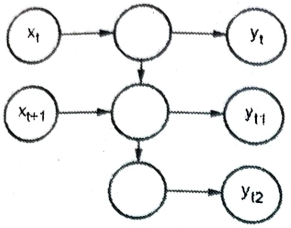


(b) One-to-many

Downloaded by ribhav soni (ribhavsoni19@gmail.com)



(c) Many-to-one



(d) Many-to-many

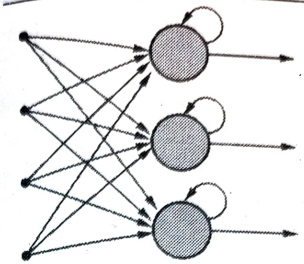
Fig. Q.10.1

Q.11 What is difference between RNNs and feed-forward neural networks ?

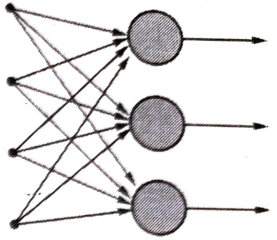
Ans. : • In a feed-forward neural network, the information only moves in one direction - from the input layer, through the hidden layers, to the output layer. The information moves straight through the network.

- Feed-forward neural networks have no memory of the input they receive and are bad at predicting what is coming next. Because a feed-forward network only considers the current input, it has no notion of order in time. It simply can not remember anything about what happened in the past except its training.
- In a RNN the information cycles through a loop. When it makes a decision, it considers the current input and also what it has learned from the inputs it received previously.

This document is available on studocu
Downloaded by ribhav soni (ribhavsoni13@gmail.com)



(a) Recurrent Neural Network



(b) Feed-forward Neural Network

Fig. Q.11.1

3.3 : Long Short-Term Memory Networks

Q.12 Explain memoryless models for sequences.

Ans. : 1. Autoregressive models : Predict the next term in a sequence from a fixed number of previous terms using "delay taps".

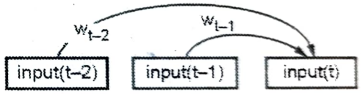
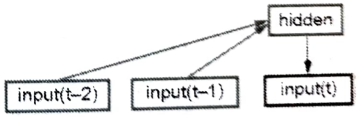


Fig. Q.12.1

2. Feed-forward neural nets : These generalized autoregressive models by using one or more layers of non-linear hidden units.



Q.12.2

Q.13 What are the major obstacles of RNNs ?

Ans. : • RNNs face two types of challenges : Exploding gradient and vanishing gradient.

A gradient is used to measure the changes in the output of a function when the inputs are slightly modified. If you consider

gradient as the slope of a function, then a higher gradient signifies a steeper slope.

- This helps a model to learn faster. Similarly, if the slope is zero, then the model will stop the learning process. A gradient indicates the change in weights with regards to change in error.
- **Exploding Gradient :** This is a scenario when user will encounter an algorithm that has assigned extremely high value to the weights.
- **Vanishing Gradient :** The second challenge is vanishing gradient occurs when the values assigned are too small. This causes the computational model to stop learning or more processing time to produce a result. This problem has been tackled in recent times with the introduction of the concept of LSTM.

Q.14 How can you overcome the challenges of vanishing and exploding gradience ?

Ans. : • Vanishing gradience can be overcome with Relu activation function, LSTM, GRU.

- Exploding gradience can be overcome with Truncated BTT, Clip gradience to threshold and RMSprop to adjust learning rate.

Q.15 Explain long short-term memory.

Ans. : • Long short-term memory (LSTM) is responsible for memory extension. LSTM forms the building units for the layers of an RNN. The purpose of LSTM is to enable RNNs to memorize inputs for an extended period.

- Fig. Q.15.1 shows block diagram of LSTM. (See Fig. Q.15.1 on next page.)
- Due to the existence of memory, LSTM has the possibility of reading, writing and deleting information from its memory, much like your personal computers. The gated cell in an LSTM network

decides whether an input should be stored or erased depending upon the importance of the information through weights.

- Over time, the algorithm can understand the importance of the information more precisely. The gates of an LSTM are divided as input gate, forget and the output gate.

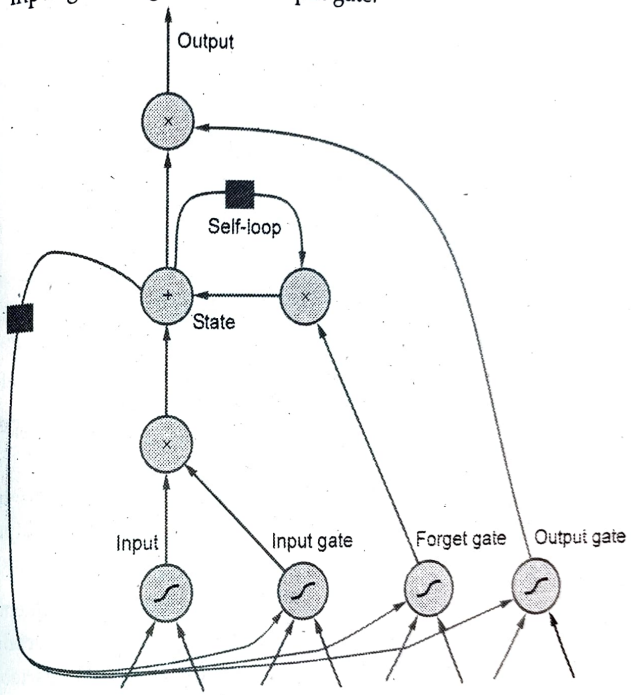


Fig. Q.15.1

- The LSTM cell manipulates input information with three gates.
  a) Input gate controls the intake of new information
  b) Forget gate determines what part of the cell state to be updated. It is decided by the sigmoid function.
  c) Output gate determines what part of the cell state to output.

Downloaded by ribhav soni (ribhavsoni19@gmail.com)

3.4 : Encoder Decoder Architectures

Q.16 Draw and explain encoder decoder architectures.

Ans. : • An RNN may be trained to translate input sequences into output sequences that are not always the same length. This occurs in many applications where the input and output sequences in the training set are typically not the same length, such as speech recognition, machine translation, or question answering.

- The RNN's input is frequently referred to as the "context." A representation of this context, C, is what we aim to create. the setting A vector or series of vectors called C may be used to condense the input sequence $X = (x^{(1)}, \ldots, x^{(nx)})$.
- The concept is relatively straightforward : (1) The input sequence is processed by an encoder, reader, or input RNN. The context C is generally simply a function of the encoder's final concealed state. (2) To produce the output sequence $Y = (y^{(1)}, \ldots, y^{(ny)})$, a decoder, writer or output RNN is conditioned on the fixed-length vector.
- Fig. Q.16.1 An example of an encoder-decoder or sequence-to-sequence RNN architecture that can learn to produce an output sequence $(y^{(1)}, \ldots, y^{(ny)})$ from an input sequence $(x^{(1)}, x^{(2)}, \ldots, y^{(nx)})$. It is made up of a decoder RNN that creates the output sequence and an encoder RNN that reads the input sequence. The decoder RNN receives an input of the typically fixed-size context variable C, which represents a semantic summary of the input sequence, from the encoder RNN's final hidden state. [Refer Fig. Q.16.1 on next page]
- To maximize the average of $\log P(y^{(1)}, \ldots, y^{(ny)} | x^{(1)}, \ldots, x^{(nx)})$ across all the pairings of x and y sequences in the training set, the two RNNs in a sequence-to-sequence architecture are trained simultaneously. The input sequence that is sent as input to the decoder RNN is generally represented by the final state h_{nx} of the encoder RNN.

This document is available on

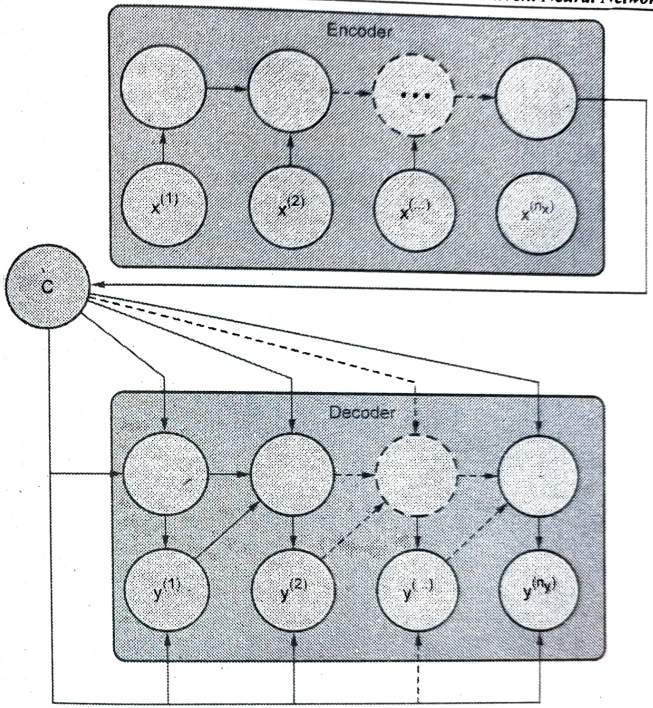


Fig. Q.16.1

- The decoder RNN is just a vector-to-sequence RNN. A vector-to-sequence RNN can accept input in at least two different ways. The input may be given as the RNN's starting state or it may be linked to the hidden components at each time step. Both of these approaches can be combined.
- There is no need that the hidden layer size of the encoder and decoder be the same.
- When the context C generated by the encoder RNN has a size that is [illegible] accurately describe a lengthy sequence, this design clearly has a constraint. As opposed to being a fixed-size vector,

studocu
Downloaded by ribhav soni (ribhavsoni19@gmail.com)

they suggested making C a variable-length sequence. They also included an attention mechanism that can be trained to link components of the C sequence to those in the output sequence.

3.5 : Recursive Neural Networks

Q.17 Write short note on recursive neural networks.

Ans. : • Another generalization of recurrent networks is represented by recursive neural networks, which have a different type of computational graph with a deep tree-like structure rather than the chain-like structure of RNNs. Fig. Q.17.1 depicts the usual computing graph for a recursive network.

- Recursive networks have been effectively used in computer vision, natural language processing, and processing data structures as input to neural nets.
- Recursive nets have a number of distinct advantages over recurrent nets, including the ability to substantially lower the depth (defined as the number of compositions of nonlinear operations) from τ to $O(\log \tau)$ for a series of the same length, which may be helpful in handling long-term dependencies. A balanced binary tree is an example of a tree structure that is independent of the data.
- The computational graph of a recursive network is a generalization of the recurrent network from a chain to a tree. A fixed-size representation (the output o) with a fixed set of parameters can be created from a variable-size sequence ($x^{(1)}, x^{(2)}, \ldots, x^{(t)}$) (the weight matrices U, V, W). The image shows a scenario of supervised learning where a target y is given and is connected to the entire sequence.

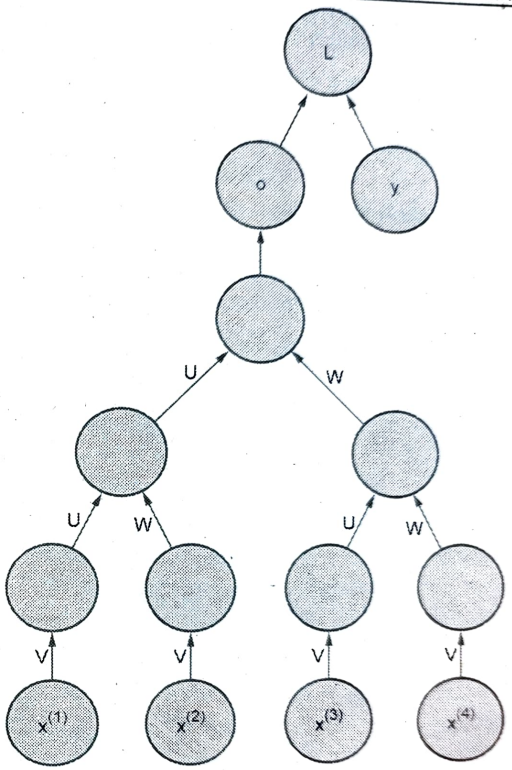


Fig. Q.17.1

- In some application fields, outside approaches can offer recommendations for the best tree structure. For instance, while processing sentences in natural language, the recursive network's tree structure can be adjusted to match the sentence's parse tree as supplied by the natural language parser. The ideal situation would be for the learner to independently identify and infer the tree structure that is optimal for every given input.

Downloaded by ribhav soni (ribhavsoni19@gmail.com)

- The recursive net concept has a wide range of conceivable variations. The inputs and targets are linked to specific nodes of the tree and the data is associated with a tree structure. Every node's computation need not be the conventional artificial neuron computation. When concepts are represented by continuous vectors, apply tensor operations and bilinear forms, which have previously been proven to be beneficial for modelling interactions between concepts.

END...

This document is available on studocu
Downloaded by ribhav soni (ribhavsoni15@gmail.com)

Unit IV

4 Autoencoders

4.1 : Under Complete Autoencoders

Q.1 What is autoencoder ?

Ans. : • An autoencoder is a special type of neural network that is trained to copy its input to its output.

- For example, given an image of a handwritten digit, an autoencoder first encodes the image into a lower dimensional latent representation, then decodes the latent representation back to an image.
- An autoencoder learns to compress the data while minimizing the reconstruction error. An autoencoder is unsupervised learning technique. It is a artificial neural network used to learn data encodings of unlabeled data or the task of representation learning.

Q.2 Explain properties of Autoencoder.

Ans. : • Data-specific : Autoencoders are only able to meaningfully compress data similar to what they have been trained on. Since they learn features specific for the given training data, they are different than a standard data compression algorithm like gzip.

- Lossy : The output of the autoencoder will not be exactly the same as the input, it will be a close but degraded representation.
- Unsupervised : Autoencoders are considered an unsupervised learning technique since they don't need explicit labels to train on.

Q.3 Explain architecture of Autoencoder.

Ans. : • Fig. Q.3.1 shows architecture of Autoencoder. (See Fig. Q.3.1 on next page.)

- Autoencoders are a specific type of feedforward neural networks trained to copy its input to output. A bottleneck is imposed in the

network to represent a compressed knowledge of the original input. The input is compressed into a lower-dimensional code and then the output is reconstructed from this representation. The code is also called as latent-space representation which is a compact "summary" or "compression" of the input.

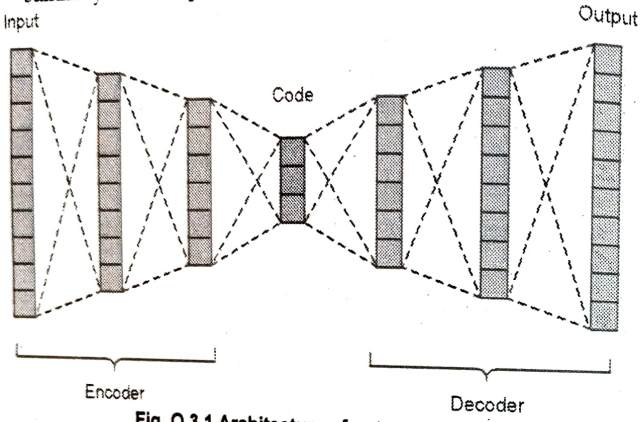


Fig. Q.3.1 Architecture of autoencoder

- An autoencoder consists of 3 components : encoder, code and decoder. The encoder compresses the input and produces the code, the decoder then reconstructs the input only using this code.
- As autoencoder is a special case of feedforward networks , training techniques similar to feedforward neural network such as minibatch gradient descent following gradients computed by back-propagation can be used for training.
- Both the encoder and decoder are fully-connected feedforward neural networks. Code is a single layer of an ANN with the dimensionality of user choice. The number of nodes in the code layer is a hyperparameter that we set before training the autoencoder.
- An autoencoder learns to copy its inputs to its outputs under some constraints : for example, limiting the dimensionality of the latent space, or adding noise to the inputs.

Q.4 Which hyperparameters must be set before training the Autoencoders ?

Ans. : There are four hyperparameters that must be set before training the autoencoders. They are as follows.

1. Code size : It is the number of nodes in the middle layer. Smaller the size more is the compression.
2. Number of layers : The autoencoder can be as deep as we like without considering the input and output.
3. Number of nodes per layer : The number of nodes per layer decreases with each subsequent layer of the encoder, and increases back in the decoder. Also the layer structure of decoder is symmetric to the encoder.
4. Loss function : Mean squared error or binary cross-entropy can be used as loss function. Cross-entropy is used if the input values are in the range [0, 1] else mean squared error is used.

Q.5 List the types of autoencoder.

Ans. : The different types of autoencoders are as follows :

1. Undercomplete autoencoders
2. Sparse autoencoders
3. Contractive autoencoders
4. Denoising autoencoders
5. Variational autoencoders

4.2 : Regularized Autoencoders

Q.6 Explain Sparse Autoencoder with its advantages and disadvantages.

Ans. : • Sparse autoencoders have hidden nodes greater than input nodes. They can still discover important features from the data. A generic sparse autoencoder is visualized where the obscurity of a node corresponds with the level of activation.

Downloaded by ribhav soni (ribhavsoni09@gmail.com)

- Fig. Q.6.1 shows simple single-layer sparse auto encoder with equal numbers of inputs (x), outputs (xhat) and hidden nodes (h).

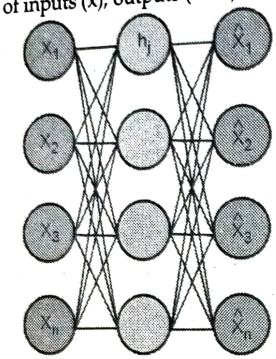


Fig. Q.6.1 Single-layer sparse auto encoder

- Sparsity constraint is introduced on the hidden layer. This is to prevent output layer copy input data.
- Sparsity may be obtained by additional terms in the loss function during the training process, either by comparing the probability distribution of the hidden unit activations with some low desired value, or by manually zeroing all but the strongest hidden unit activations.
- **Advantages :**
  1. Sparse autoencoders have a sparsity penalty, a value close to zero but not exactly zero. Sparsity penalty is applied on the hidden layer in addition to the reconstruction error. This prevents overfitting.
  2. They take the highest activation values in the hidden layer and zero out the rest of the hidden nodes.
- **Disadvantages :**
  1. For it to be working, it's essential that the individual nodes of a trained model which activate are data dependent, and that different inputs will result in activations of different nodes through the network.

This document is available on studocu
Downloaded by ribhav soni (ribhavsoni15@gmail.com)

Q.7 Discuss about Denoising Autoencoders.

Ans. : • Autoencoder can learn useful representations by changing the reconstruction error term of the cost function rather than adding a penalty Ω to the cost function.

- In contrast to traditional autoencoder that minimize some loss function as given in equation (Q.7.1) the denoising autoencoder (DAE) minimizes the loss function given by equation (Q.7.2)

$$L(x, g(f(x))) \quad \ldots(\text{Q.7.1})$$

  - *L is a loss function that penalize g(f(x)) for being dissimilar from x, e.g.* L^2 *norm of their difference.*

$$L(x, g(f(\tilde{x}))) \quad \ldots(\text{Q.7.2})$$

  - $\tilde{x}$ *is a corrupted copy of x by adding some form of noise.*
- This helps to avoid the autoencoders to copy the input to the output without learning features about the data.
- Denoising autoencoders thus provide yet another example of how useful properties can emerge as a byproduct of minimizing reconstruction error.
- They are also an example of how overcomplete, high-capacity models may be used as autoencoders as long as care is taken to prevent them from learning the identity function.

4.3 : Stochastic Encoders and Decoders

Q.8 Write short note on Stochastic Encoders and Decoders.

Ans. : • Autoencoders are feedforward networks and use the same loss functions and output unit that are used in traditional feedforward networks

- For designing the output units and the loss function of a feedforward network, an output distribution p(y | x) is defined and the negative log-likelihood –log p(y | x) is minimized where y a vector of targets, e.g. class labels .

- But in an autoencoder, target as well as the input is x and still the same strategy can be applied. So by using same strategy as in feedforward network, we can assume that for a given code h, decoder is providing a conditional distribution $p_{decoder}(x|h)$. Autoencoder can then be trained by minimizing $-\log p_{decoder}(x \mid h)$ where the exact form of the loss function depends on the form of $p_{decoder}$.
- Similar to traditional feedforward networks, linear output units are used to parameterize the mean of a Gaussian distribution for real valued x. The negative log-likelihood yields a mean squared error criterion in this case. Binary x values correspond to a Bernoulli distribution whose parameters are given by a sigmoid output unit, discrete x values correspond to a softmax distribution and so on.
- Given h, the output variables are treated as conditionally independent so that evaluation of probability distribution is inexpensive. For modeling outputs with correlations, mixture density outputs can be used.
- Encoding function f(x) can be generalized to an encoding distribution $p_{encoder}(h \mid x)$, as shown in Fig. Q.8.1. Fig. Q.8.1 shows the structure of stochastic autoencoder. Here both encoder and decoder involve some noise injection. The output of encoder and decoder can be seen as sampled from a distribution, $p_{encoder}(h \mid x)$ for encoder and $p_{decoder}(x \mid h)$ for the decoder.

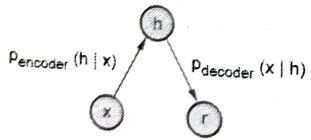


Fig. Q.8.1 The structure of a stochastic autoencoder

- Any latent variable model $p_{model}(h, x)$ defines a stochastic encoder

$$p_{encoder}(h \mid x) = p_{model}(h \mid x) \quad \ldots(Q.8.1)$$

and a stochastic decoder

$$p_{decoder}(x \mid h) = p_{model}(x \mid h) \quad \ldots(Q.8.2)$$

- In general, the encoder and decoder distributions need not be necessarily conditional distributions compatible with a unique joint distribution $p_{model}(x, h)$.

4.4 : Denoising Autoencoders

Q.9 Define Denoising Autoencoders.

Ans. : Denoising autoencoders are a stochastic version of standard autoencoders that reduces the risk of learning the identity function. Denoising autoencoders attempt to get around this risk of identity-function affiliation by introducing noise, i.e. randomly corrupting input so that the autoencoder must then "denoise" or reconstruct the original input.

Q.10 What is use of denoising autoencoder ?

Ans. : Denoising autoencoder helps :

a) The hidden layers of the autoencoder learn more robust filters

b) Reduce the risk of overfitting in the autoencoder

c) Prevent the autoencoder from learning a simple identify function

Q.11 Explain Denoising Autoencoders.

Ans. : • Fig. Q.11.1 shows denoising autoencoder.

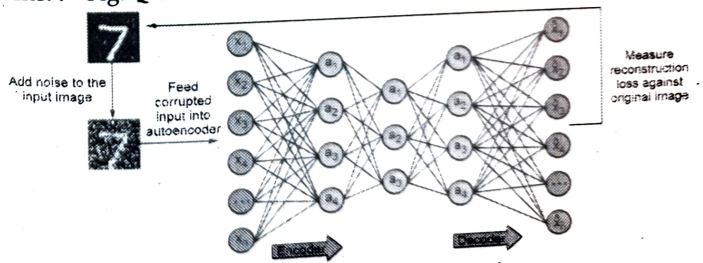


Fig. Q.11.1 Denoising autoencoder

Downloaded by ribhav soni (ribhavsoni13@gmail.com)

- The denoising autoencoder (DAE) receives a corrupted data point as input. It is trained to predict the uncorrupted original, data point as the output.
- Fig. Q.11.2 illustrates the training procedure of DAE. A corruption process $C(\tilde{x} \mid x)$ is introduced. It represents a conditional distribution over corrupted samples $\tilde{x}$, given a data sample x.

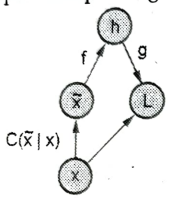


Fig. Q.11.2 The computational graph of the cost function for a denoising autoencoder

- The autoencoder then learns a reconstruction distribution $p_{reconstruct}(x \mid \tilde{x})$ estimated from training pairs $(x,\tilde{x})$ as follows :
  1. Sample a training example x from the training data.
  2. Sample a corrupted version $\tilde{x}$ from $C(\tilde{x} \mid x = x)$.
  3. Use $(x,\tilde{x})$ as a training example for estimating the autoencoder reconstruction distribution $p_{reconstruct}(x \mid \tilde{x}) = p_{decoder}(x \mid h)$ with h the output of encoder $f(\tilde{x})$ and $p_{decoder}$ typically defined by a decoder g(h).
- Gradient-based approximate minimization (such as minibatch gradient descent) can be performed on the negative log-likelihood $-\log p_{decoder}(x \mid h)$.
- As long as the encoder is deterministic, the denoising autoencoder is a feedforward network. So it can be trained using exactly the same techniques as that of any other feedforward network. DAE performs stochastic gradient descent on the following expectation:–

This document is available on studocu
Downloaded by ribhav soni (ribhavsoni19@gmail.com)

$$- E_{\tilde{x} \sim \hat{p}_{data}(x)} E_{\tilde{x} \sim C(\tilde{x} \mid x)} \log p_{decoder}(x \mid h = f(\tilde{x})) \quad \ldots(Q.11.1)$$

where $\hat{p}_{data}(x)$ is the training distribution.

4.5 : Contractive Autoencoders

Q.12 Write short note on Contractive Autoencoders.

Ans. : • The main goal of Contractive Autoencoder (CAE) is to have a robust learned representation that is less sensitive to small variation in the data.

- A penalty term is applied to the loss function so as to make the representation robust.
- In order to make the derivatives of f to be as small as possible, contractive autoencoder introduces an explicit regularizer on the code h = f(x)
- The penalty term is Frobenius norm of the Jacobian matrix which is calculated with respect to input for the hidden layer. Frobenius norm of the Jacobian matrix is the sum of square of all elements. This is shown in Fig. Q.12.1.

$$L = I x - g(f(x)) I + \lambda \| J_f(x) \|_F^2$$

$$\| J_f(x) \|_F^2 = \sum_{i,j} \left(\frac{\partial h_j(x)}{\partial x_i} \right)^2$$

Fig. Q.12.1 Loss function with penalty term - Frobenius norm of the Jacobian matrix

- Contractive autoencoder is similar to denoising autoencoder in a sense that in presence of small Gaussian noise the denoising reconstruction error is equivalent to a contractive penalty on the reconstruction function that maps x to r = g(f(x)).

- This means, in case of denoising autoencoders the reconstruction function resist small but finite sized perturbations of the input, while in contractive autoencoders the feature extraction function resist infinitesimal perturbations of the input.
- CAE surpasses results obtained by regularizing autoencoder using weight decay or by denoising. CAE is a better as compare to denoising autoencoder to learn useful feature extraction.
- The model is encouraged to learn an encoding where similar inputs have similar encodings. So the model is forced to learn how to **contract** a neighborhood of inputs into a smaller neighborhood of outputs.
- Fig. Q.12.2 indicates how the derivative of the reconstructed data (i.e slope) is essentially zero for local neighborhoods of input data.
- This can be achieved by penalizing the instances where a small change in the input leads to a large change in the encoding space. For this the loss term should penalizes large *derivatives* of *hidden* layer activations w.r.t. the input training instances.

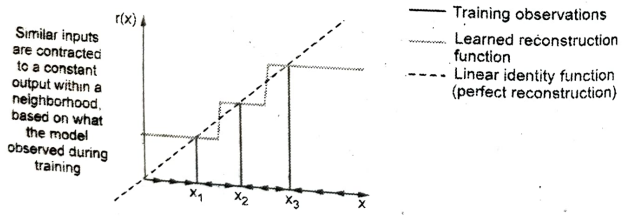


Fig. Q.12.2 Slope of the reconstructed data

- Regularization loss term used is the squared Frobenius norm $\|A\|_F$ of the Jacobian matrix J for the hidden layer activations w.r.t. the input observations. A Frobenius norm is an L_2 norm for a matrix. The Jacobian matrix represents all first-order partial derivatives of a vector valued function.

- These values can be calculated using equation (Q.12.1) and (Q.12.2). where

  m : Observations

  n : Hidden layer nodes, and

Loss function can be defined by equation (Q.12.3)

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{i,j}|^2} \quad \ldots (Q.12.1)$$

$$J = \begin{bmatrix} \frac{\delta a_1^{(h)}(x)}{\delta x_i} & \cdots & \frac{\delta a_1^{(h)}(x)}{\delta x_m} \\ \vdots & \ddots & \vdots \\ \frac{\delta a_n^{(h)}(x)}{\delta x_i} & \cdots & \frac{\delta a_n^{(h)}(x)}{\delta x_m} \end{bmatrix} \quad \ldots (Q.12.2)$$

$$L(x,\hat{x} + \lambda \sum_i \|\nabla_x a_i^{(h)}(x)\|^2 \quad \ldots (Q.12.3)$$

where

$\nabla_x a_i^{(h)}(x)$: Gradient field of hidden layer activations w.r.t. the input x, summed over all i training examples.

4.6 : Applications of Autoencoders

Q.13 Explain how the dimensionality reduction feature of autoencoder is useful in information retrieval task.

Ans. : • Information retrieval is the task of searching the entries in a database that resemble a query entry.

- Dimensionality reduction benefits the task of information retrieval. In case of certain type of low dimensional data search can become more efficient due to dimensionality reduction. As one of the application of autoencoder is dimensionality reduction, they can be applied to information retrieval using semantic hashing.

Downloaded by ribhav sood (ribhavsood@gmail.com)

- By training the dimensionality reduction algorithm to produce low dimensional and binary code , all database entries can be stored in a hash table that maps binary code vectors to entries.
- Using this hash table information retrieval can be performed by returning all database entries having same binary code as the query. By flipping some bits from encoding of the query, slightly less similar entries can also be searched very efficiently.
- This approach of information retrieval is suitable for both textual and image data.
- Typically encoding function with sigmoids is used on final layer for producing binary codes for semantic hashing. The sigmoid unit must be saturated to nearly 0 or nearly 1. For this additive noise is injected before sigmoid function during training and its magnitude should increase over time.
- So in order to preserve as much information as possible against this noise , the network must increase the magnitude of the inputs to the sigmoid function, until saturation occurs.

Q.14 Explain any two application of Autoencoders.

Ans. : 1. Anomaly detection

- A neural network trained with specific dataset learns the data it commonly "sees" and represents the input dataset. This network can also represent the difference between input and output and tell us when it "sees" unusual data.
- Autoencoders can be used in such systems where it is difficult to describe unusual or anomalous data. Undercomplete autoencoders are used for anomaly detection.
- If autoencoder is trained on specific image dataset say "D", then it is supposed to reconstruct the image as it is with low reconstruction loss.

This document is available on studocu
Downloaded by Ribhav soni (ribhavsoni1@gmail.com)

- But autoencoder cannot perform reconstruction task for image which is not present in the training dataset because latent attributes will not be adapted by network for any unseen image. As a result the reconstruction loss for such image will be very high. So by setting appropriate threshold it can be easily identified as anomaly or unusual image.
- Due to this autoencoders are good at powering anomaly detection systems.

2. Image denoising

- Image denoising is performed to obtain the proper information about the content of an image. Denoising autoencoders can be used for this.
- Denoising autoencoders do not search for noise in image, instead they extract the image from noisy data fed as input to them by learning their representations. Then the noise free image is formed by decompressing these representations
- Denoising autoencoders can perform efficient and highly accurate image denoising and can be used for denoising complex images that could not be denoised using traditional methods.

END...

Unit V

5 Representation Learning

5.1 : Greedy Layer - Wise Unsupervised Pre-training

Q.1 What is Representation Learning ?

Ans. : • Representation learning is concerned with training machine learning algorithms to learn useful representations.

- Deep neural networks can be considered representation learning models that typically encode information which is projected into a different subspace. These representations are then usually passed on to a linear classifier to, for instance, train a classifier.
- Representation learning can be divided into :
  a) Supervised representation learning : Learning representations on task A using annotated data and used to solve task B
  b) Unsupervised representation learning : Learning representations on a task in an unsupervised way. These are then used to address downstream tasks and reducing the need for annotated data when learning news tasks.

Q.2 What is Greedy Algorithm ?

Ans. : Greedy algorithms break a problem into many components, then solve for the optimal version of each component in isolation. Unfortunately, combining the individually optimal components is not guaranteed to yield an optimal complete solution.

Q.3 Write and explain an algorithm for Gready Layer-Wise Unsupervised Pretraining.

Ans. : • Greedy Layer - Wise Unsupervised Pretraining relies on single-layer representation learning algorithm. Each layer is pretrained using unsupervised learning, taking the output of previous layer and producing as output a new representation of the data, whose distribution is hopefully simpler.

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

$$F \leftarrow \text{Identify function}$$
$$\tilde{X} = X$$
$$\text{for } k = 1 \ldots m \text{ do}$$
$$f^{(k)} = L(\tilde{X})$$
$$f \leftarrow f^{(k)} \circ f$$
$$\tilde{X} \leftarrow f^{(k)}(\tilde{X})$$
$$\text{end for}$$
$$\text{if fine-tuning then}$$
$$f \leftarrow T(f \cdot X \cdot Y)$$
$$\text{end if}$$
$$\text{Return f}$$

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

Q.4 When and why does unsupervised pretraining ideas work ?

Ans. :

1. Idea1 : Choice of initial parameters for a deep NN can have a significant regularizing effect on the model.
   - It remains possible that pretraining initializes the model in a location that would otherwise be inaccessible. For example : a region surrounded by areas where the cost function varies so

Downloaded by ribhav soni (ribhavsoni19@gmail.com)

much from one example to another that mini-batches give only a very noisy estimate of the gradient a region surrounded by areas where the Hessian matrix is so pooly conditioned that gradient descent methods must use very small steps.

- We cannot characterize the exactly what aspects of the pretrained parameters are retained during the supervised training stages.

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

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

Q.5 Why it is called Greedy layer-wise pretraining ?

Ans. : • Greedy because, it is a greedy algorithm that optimizes each piece of the solution independently

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

Q.6 Explain disadvantage of Unsupervised Pretraining.

Ans. :

1. Unsupervsied pretraining does not offer a clear way to adjust the strength of the regularization arising from the unsupervised stage. When we perform unsupervised and supervised learning

This document is available on

simultaneously, instead of using the pretraining strategy, there is a single hyperparameter, usually a coefficient attatched to the unsupervised cost, that determine hows strongly supervised objective will regularize the supervised model.

2. Two separate training phases has its own hyperparameters. The performance of the second phase cannot be predicted during the first phase, so there is a long delay between proposing hyperparameters for the first phase and being able to update them using feedback from the second phase. Most principled approach : use validation set error in the supervised phase to select hyperparameters of the pretraining phase.

5.2 : Transfer Learning and Domain Adaption

Q.7 Define Transfer learning and domain adaptation.

Ans. : Transfer learning and domain adaptation refer to situation where what has been learned in one setting is exploited to improve generalization in another settings.

Q.8 What is transfer learning ? Explain its types.

Ans. : • Transfer learning, multitask learning and domain adaptation can be achieved via representation learning when there exist features that are useful for different settings or tasks, corresponding to underlying factors that appear in more than one setting.

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

studocu

Downloaded by ribhav soni (ribhavsoni19@gmail.com)

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

5.3 : Distributed Representation

Q.9 What is distributed representation ? Explain symbolic representation.

Ans. : • Distributed representations is composed of many elements that can be set separately from each other.

- Symbolic representation : The input is associated with a single symbol or a category. If there are n symbols in the dictionary, one can imagine n feature detectors, each corresponding to the detection of the presence of the associated category.
- Only n different configurations of the representation space are possible, carving n different regions in input space. It is also called one-hot representation.

Q.10 What is Nondistributed representations ? Explain example of it.

Ans. : • Nondistributed representations may contain many entries but without significant meaningful separate control over each entry.

- Examples of learning algorithm based on nondistributed representation learning are :
  a. Clustering method, include the k-means algorithm : Each input assigned to exactly one cluster
  b. K Nearest neighbor : If k > 1, multiple values describe each input, but they cannot be controlled separately from each other, so this does not qualify as a true distributed representation.
  c. Decision tree : Only one leaf is activated when the input is given
  d. Gaussian mixture and mixtures of expert : Each input is represented with multiple values, but those values cannot be readily be controlled separately from each other.

5.4 : Variants of CNN : DenseNet

Q.11 Write short note on Dense Block.

Ans. : • Standard ConvNet uses several convolutions to extract high-level characteristics from the input picture.

- Identity mapping is suggested in ResNet to promote gradient propagation. It uses element-by-element addition. It may be thought of as an algorithm that is handed a state from one ResNet module to another.
- Each layer in DenseNet receives extra inputs from all levels that came before it and transmits its own feature-maps to all layers that came after it. You utilize concatenation. Each layer receives "collective knowledge" from the levels that came before it.
- Fig. Q.11.1 shows the DenseNet block.

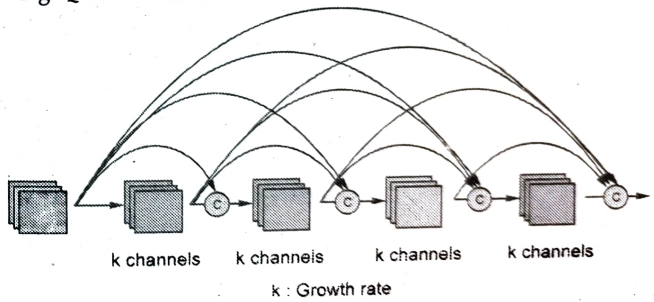


Fig. Q.11.1 Dense Block in DenseNet with growth rate k

- Each layer receives feature maps from all layers that came before it, allowing for a more compact and thin network with fewer channels. The extra number of channels for each layer is the growth rate k.
- Therefore, it has greater memory and processing efficiency.

Downloaded by ribhav soni (ribhavsoni10@gmail.com)

Q.12 Draw and explain DenseNet Architecture.

Ans. : • Basic DenseNet composition layer :

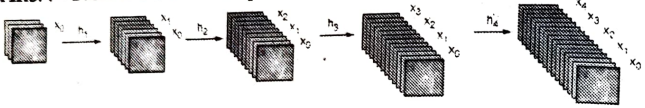


Fig. Q.12.1 Composition layer

- Pre-Activation Batch Norm (BN), ReLU and 3 × 3 Conv are performed for each composition layer with output feature maps of k channels, for example, to transform x_0, x_1, x_2 and x_3 to x_4. The Pre-Activation ResNet came up with this concept.
- **DenseNet-B (Bottleneck layers)**
  - BN-ReLU-1 × 1 Conv is carried out prior to BN-ReLU-3 × 3 Conv to lessen the complexity and size of the model.

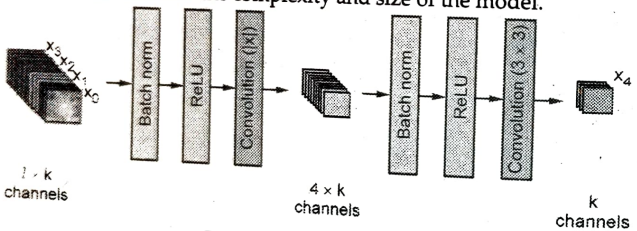


Fig. Q.12.2 DenseNet-B

- **Multiple Dense Blocks with Transition Layers**
  - The transition layers between two adjacent dense blocks are 1 × 1 Conv and 2 × 2 average pooling.
  - Within the dense block, feature map sizes are uniform, making it simple to concatenate them.
  - A softmax classifier is applied once a global average pooling is completed at the conclusion of the final dense block. (Refer Fig. Q.12.3 on next page)

This document is available on studocu

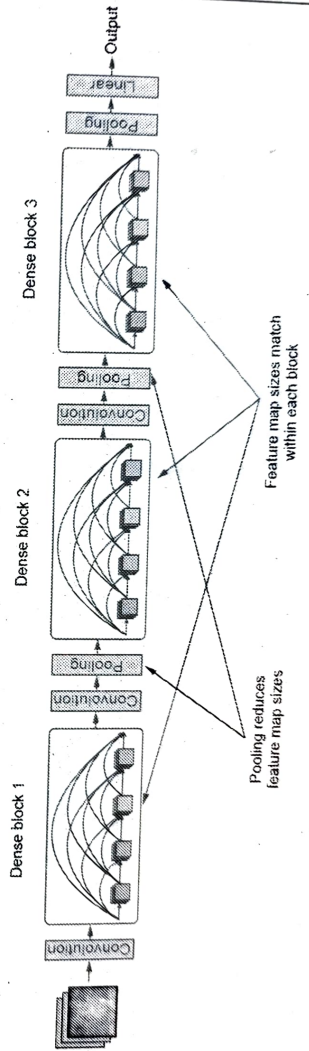


Fig. Q.12.3 Multiple dense blocks

Downloaded by ribhav soni (ribhavsoni19@gmail.com)

- **DenseNet-BC (Further Compression)**
  - The transition layer produces m output feature maps if a dense block has m feature-maps, where $0 < \theta \leq 1$ is referred to as the compression factor.
  - The quantity of feature-maps across transition layers is constant when $\theta = 1$. DenseNet-C, or DenseNet with a value of $\theta < 1$, and $\theta = 0.5$ in the experiment.
  - The model is known as DenseNet-BC when both the bottleneck and transition layers with $\theta < 1$ are implemented.
  - DenseNets with/without B/C, various L layers, and k growth rates are also trained at this point.

Q.13 Why do we need DenseNets ?

Ans. : DenseNet was specially developed to improve accuracy caused by the vanishing gradient in high-level neural networks due to the long distance between input and output layers and the information vanishes before reaching its destination.

Q.14 List the advantages of DenseNets.

Ans. : Advantages :

1. Parameter efficiency : Every layer adds only a limited number of parameters- for e.g. only about 12 kernels are learned per layer
2. Implicit deep supervision : Improved flow of gradient through the network- Feature maps in all layers have direct access to the loss function and its gradient

END...

Downloaded by ribhav soni (ribhavsoni19@gmail.com)

Unit VI

6 Applications of Deep Learning

6.1 : Overview of Deep Learning Applications : Image Classification

Q.1 How is deep learning applied to computer vision tasks ?

Ans. : • With the help of convolutional neural networks, deep learning is able to perform the following tasks :

a) Object recognition b) Face recognition

c) Motion detection d) Pose estimation

e) Semantic segmentation

- Object recognition (detection) : Nowadays AI is able to recognize both static and dynamically moving objects with 99 % accuracy. In general, it is a matter of dividing the image into fragments and letting algorithms find the similarities to one of the existing objects in order to assign it to one of the classes. Classification plays an important role in this process and the success of object recognition largely depends on the richness of the object database.
- Face recognition is the identification of a specific person known to the system from the database.
- Motion detection : Motion detection is a key part of any surveillance system. This may be used to trigger an alarm, send a notification to someone, or simply record the event for later analysis. One way to detect motion is by using a motion detector, which detects changes between frames of an image sequence. The simplest form of motion detection is threshold.
- Pose estimation : Human pose recognition is a challenging computer vision task due to the wide variety of human shapes and appearance, difficult illumination and crowded scenery. For these

tasks, photographs, image sequences, depth images, or skeletal data from motion capture devices are used to estimate the location of human joints.

- Semantic segmentation is a type of deep learning that attempts to classify each pixel in an image into one of several classes, such as road, sky or grass. These labels are then used during training so that when new images are processed they can also be segmented into these categories based on what they look like compared with previously seen pictures.

Q.2 What is image classification ?

Ans. : • Image classification is the task of categorizing and assigning labels to groups of pixels or vectors within an image dependent on particular rules. The categorization law can be applied through one or multiple spectral or textural characterizations.

- Image classification techniques are mainly divided into two categories : Supervised and unsupervised image classification techniques.

Q.3 Explain supervised and unsupervised classification.

Ans. : 1. Supervised classification : Supervised image classification methods use previously classified reference samples in order to train the classifier and subsequently classify new, unknown data. Therefore, the supervised classification technique is the process of visually choosing samples of training data within the image and allocating them to pre-chosen categories, including vegetation, roads, water resources and buildings. This is done to create statistical measures to be applied to the overall image.

2. Unsupervised classification : Unsupervised classification technique is a fully automated method that does not leverage training data. This means machine learning algorithms are used to analyze and cluster unlabeled datasets by discovering hidden patterns or data groups without the need for human intervention.

This document is available on studocu

Q.4 What is image classification in deep learning ?

Ans. : • Image classification is where a computer can analyses an image and identify the 'class' the image falls under. For example, input an image of a sheep. Image classification is the process of the computer analyzing the image and telling you it is a sheep.

- Early image classification relied on raw pixel data. This meant that computers would break down images into individual pixels. The problem is that two pictures of the same thing can look very different. They can have different backgrounds, angles, poses, etc. This made it quite the challenge for computers to correctly 'see' and categories images.
- Image classification with deep learning most often involves convolutional neural networks, or CNNs. In CNNs, the nodes in the hidden layers don't always share their output with every node in the next layer.
- Deep learning allows machines to identify and extract features from images. This means they can learn the features to look for, in images by analyzing lots of pictures.

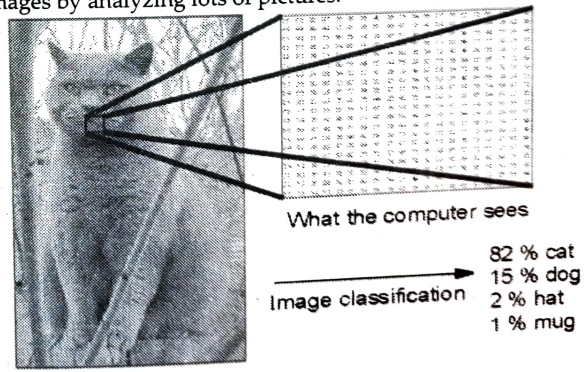


Fig. Q.4.1

Downloaded by ribhav soni (ribhavsoni19@gmail.com)

- Image is analyzed in the form of pixels. Image is an array of pixels where size of the matrix depends on resolution of an image. Image classification task is done by grouping pixels into specified categories referred to as classes.
- The image is segregated into its most prominent features using the algorithm giving the classifier an idea about the class of the image it may belong to. Thus the feature extraction process is most important step in image classification. Also data fed to the algorithm plays an important role particularly in supervised image classification technique.

Q.5 List the application areas of image classification.

Ans. : Image classification forms basis of computer vision problems.

1. In self-driving cars image classification can be used to detect traffic lights, trees, vehicles, peoples around etc.
2. In healthcare it can be used to analyze medical images and depict the symptoms of illness.
3. With ubiquitous technologies such as AI and IOT, huge amounts of data in the form of images, video and speech are generated. Image and video data posted by persons can be used in recommendation system in online shopping or places to visit etc.

Q.6 How image is classified using convolutional neural network ?

Ans. : • Fig. Q.6.1 shows CNN for image classification.

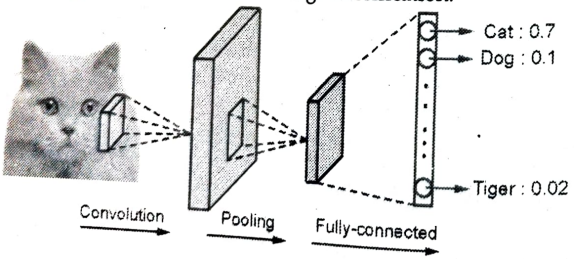


Fig. Q.6.1

- CNN layers can be of four main types : Convolution, ReLu, pooling and fully-connected layer.

1. **Convolution Layer :** A convolution is the simple application of a filter to an input that results in an activation. The convolution layer has a set of trainable filters that have a small receptive range but can be used to the full-dept of data provided. Convolution layers are the major building blocks used in convolutional neural networks.
2. **ReLu Layer :** ReLu layers, also known as Rectified linear unit layers, are activation functions applied to lower overfitting and build the accuracy and effectiveness of the CNN. Models that have these layers are easier to train and produce more accurate results.
3. **Pooling Layer :** This layer collects the result of all neurons in the layer preceding it and processes this data. The primary task of a pooling layer is to lower the number of factors being considered and give streamlined output.
4. **Fully-Connected Layer :** This layer is the final output layer for CNN models that flattens the input received from layers before it and gives the result.

6.2 : Social Network Analysis

Q.7 What is a social network ?

Ans. : A social network is a group of collaborating, and/or competing individuals or entities that are related to each other. Social network is formally defined as a set of social actors, or nodes, members that are connected by one or more types of relations.

Q.8 What is social network analysis ?

Ans. : Social Network Analysis (SNA) is the study of social relations among a set of actors.

Downloaded by ribhav soni (ribhavsoni29@gmail.com)

Q.9 List the principles of social network analysis.

Ans. : 1. Actors and their actions are viewed as interdependent rather than independent, autonomous units.

2. Relational ties (linkages) between actors are channels for transfer or "flow" of resources (either material or nonmaterial)

3. Network models focusing on individuals view the network structure environment as providing opportunities for or constraints on individual action.

4. Network models conceptualize structure (social, economic, political and so forth) as lasting patterns of relations among actors.

Q.10 Explain terminology used in social network analysis.

Ans. : • Terminology is used in social network analysis : Actor, relational tie, dyad, triad, subgroup, group and relation.

- Actor : Actor is discrete individual, corporate, or collective social units. Examples : People in a group, departments within in a corporation, public service agency in a city, nation-states in the world system.
- Relational tie : Actors are linked to another by social ties. A tie establishes a linkage between a pair of actors.
- Dyad : It is a tie between two actors and consists of a pair of actors and the tie(s) between them.
- Triad : Triples of actors and associated ties. A subset of three actors and the tie(s) among them.
- Subgroup of actors is defined as any subset of actors and all ties among them.
- Group : Group is the collection of all actors on which ties are to be measured.

This document is available on

- Relation : It is the collection of ties of a specific kind among members of a group. Example : The set of friendship among pairs of children in a classroom.
- Network can be categorized by the nature of the sets of actors and the properties of the ties among them. The number of modes in a network refers to the number of distinct kinds of social entities in the network.
- One-mode networks are a single set of actors. Two-mode networks are focus on two sets of actors, or one set of actors and one set of events

Q.11 What is social network analysis ? Explain.

Ans. : • Social Network Analysis [SNA] is the mapping and measuring of relationships and flows between people, groups, organizations, computers, URLs and other connected information/knowledge entities. The term "social network" has been introduced by Barnes in 1954.

- SNA is the study of social relations among a set of actors. The methods of data collection in network analysis are aimed at collecting relational data in a reliable manner.
- Data collection is typically carried out using standard questionnaires and observation techniques that aim to ensure the correctness and completeness of network data.
- SNA is based on an assumption of the importance of relationships among interacting units.
- The social network perspective encompasses theories, models and applications that are expressed in terms of relational concepts or processes.
- The nodes in the network are the people and groups while the links show relationships or flows between the nodes. SNA

studocu

Downloaded by ribhav soni (ribhavsoni2@gmail.com)

provides both a visual and a mathematical analysis of human relationships.

- The advantage of social network analysis is that, unlike many other methods, it focuses on interaction. Network analysis allows us to examine how the configuration of networks influences how individuals and groups, organizations, or systems function.
- **Features of social network analysis :** Structural intuition, systematic relational data, graphic representation and mathematical or computational models.
- **Social network analysis :**
  a) Refers to the set of actors and the ties among them.
  b) Views on characteristics of the social units arising out of structural or relational processes or focuses on properties of the relational system themselves.
  c) Inclusion of concepts and information on relationships among units in a study.
  d) The task is to understand properties of the social (economic or political) structural environment, and
  e) How these structural properties influence observed characteristics and associations among characteristics.
  f) Relational ties among actors are primary and attributes of actors are secondary.
  g) Each individual has ties to other individuals, each of whom in turn is tied to a few, some, or many others and so on.

Q.12 Explain different application of social network analysis.

Ans. : • Social Network Analysis (SNA) is an important and valuable tool for knowledge extraction from massive and un-structured data. Social network provides a powerful abstraction of the structure and dynamics of diverse kinds of inter-personal connection and interaction.

- Facebook is a social networking service and website that connects people with other people and share data between people. A user can create a personal profile, add other users as friends, exchange data, create and join common interest communities.
- Twitter is a social net-working and microblogging service. The users of Twitter can exchange text-based posts called **tweets.** A tweet is a maximum 140 characters long but can be augmented by pictures or audio recording. The main concept of Twitter was to build a social network formed by friends and followers. Friends are people who you follow, followers are those who follow you.
- The role of social networks in labor markets deserves attention for at least two reasons : First, because of the central role networks play in disseminating information about job openings they place a critical role in determining whether labor markets function efficiently; and second, because network structure ends up having implications for things like human capital investment as well as inequality.
- Social Network Analysis (SNA) primarily focuses on applying analytic techniques to the relationships between individuals and groups and investigating how those relationships can be used to infer additional information about the individuals and groups.
- SNA is used in a variety of domains. For example, business consultants use SNA to identify the effective relationships between workers that enable work to get done; these relationships often differ from connections seen in an organizational chart.
- Law enforcement personnel have used social networks to analyze terrorist networks and criminal networks. The capture of Saddam Hussein was facilitated by social network analysis : Military officials constructed a network containing Hussein's tribal and family links, allowing them to focus on individuals who had close ties to Hussein.

Downloaded by ribhav soni (ribhavsoni19@gmail.com)

Q.13 How social network analysis helps to deep learning ?

Ans. : • For social network analysis, the primary step is to encode the network data into network embeddings. Network embeddings are low dimensional representations of network data. Applications like classifications, clustering, link prediction etc. are possible due to network representation learning.

- Deep neural networks can be used to learn representations from network data. Three categories based on type of neural network are :
  1) Look-up table based models
  2) Autoencoder based models
  3) Graph Convolutional Network (GCN) based models.

1. Models with embedding look - Up tables

- Look up tables can be used for network representation learning instead of using multiple layers of nonlinear transformations. Node index is directly mapped with its corresponding representation vector using the look up tables.
- Look up table can be implemented using matrix. Each row of the matrix corresponds to the representation of one node.
- Main building blocks of the model with embedding look-up tables are shown in Fig. Q.13.1. Sampling and modeling are the two key components of this approach.

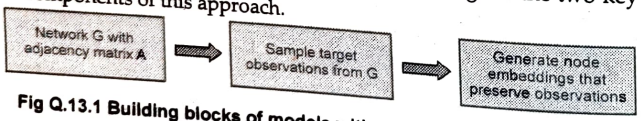


Fig Q.13.1 Building blocks of models with embedding look - up tables

2. Autoencoder based models

- Two neural network modules of an autoencoder are :
  i) Encoder and ii) Decoder.

This document is available on studocu
Downloaded by shav soni (ribhavsoni19@gmail.com)

- Fig. Q.13.2 shows autoencoder based network representation. The function of encoder is to map the features of each node into a latent space. Information about the network. is then reconstructed by decoder from this latent space.
- The size of hidden representation layer is usually small as compare to input/output layer. The non linear network structure is captured by this compressed representation. The output is decoded from these low dimensional representations. Minimizing the reconstruction error between input and decoded output is the main objective function of autoencoder.
- Autoencoder is fed with the rows of the proximity matrix $S \in R^{|V| \times |V|}$ so as to learn and generate embeddings $Z \in R^{|V| \times D}$ at the hidden layer.

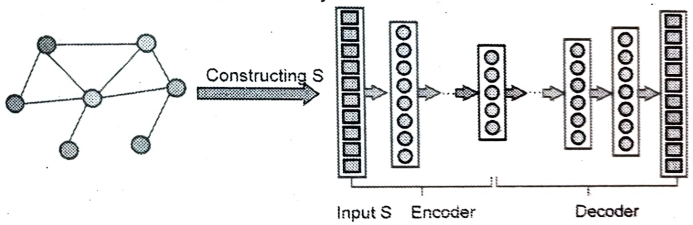


Fig Q.13.2 Autoencoder based network representation

Q.14 What is graph convolutional approaches ?

Ans. : • Graph Convolutional Network (GCN) is an approach for semi-supervised learning on graph-structured data. It is based on an efficient variant of convolutional neural networks which operate directly on graphs.

- The choice of convolutional architecture is motivated via a localized first-order approximation of spectral graph convolutions. The model scales linearly in the number of graph edges and learns hidden layer representations that encode both local graph structure and [illegible] nodes.

6.3 : Speech Recognition

Q.15 What is speech recognition ?

Ans. : Speech recognition is the ability of a machine or program to identify words spoken aloud and convert them into readable text.

Q.16 How to formulate Automatic Speech Recognition (ASR) ?

Ans. : • The main goal of an ASR system is to transform an audio input signal $x = (x_1, x_2, x_3, \ldots, x_T)$ with a specific length T into a sequence of words or characters $y = (y_1, y_2, \ldots, y_N)$, $y_n \in V$, where V is the vocabulary

• Fig. Q.16.1 shows Automatic Speech Recognition.

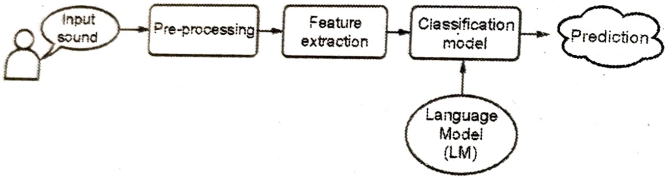


Fig. Q.16.1 ASR

- Processing steps for ASR system are pre-processing, feature extraction, classification and language modeling.
- The pre-processing step aims to improve the audio signal by reducing the signal-to-noise ratio, reducing the noise and filtering the signal.
- The features that are used for ASR, are extracted with a specific number of values or coefficients, which are generated by applying various methods on the input. Feature extraction techniques are Mel-Frequency Cepstral Coefficients (MFCCs) and Discrete Wavelet Transform (DWT).
- The classification model aims to find the spoken text which is contained on the input signal. It takes the extracted features from the pre-processing step and generates the output text.

- The Language Model (LM) is an important module as it captures the grammatical rules or the semantic information of a language. Language models are important in order to recognize the output taken from the classification model as well as to make corrections on the output text.

Q.17 How recurrent neural networks support speech recognition ?

Ans. : • RNNs perform computations on the time sequence since their current hidden state is dependent on all the previous hidden states. More specifically, they are designed to model time-series signals as well as capture long-term and short-term dependencies between different time-steps of the input.

• Fig. Q.17.1 shows bidirectional RNN used for automatic speech recognition.

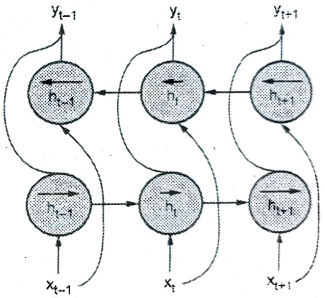


Fig. Q.17.1 Bidirectional RNN for ASR

The input sequence is $x = (x_1, x_2, \ldots, x_T)$

Hidden sequence is $h = (h_1, h_2, \ldots, h_N)$ and

Output sequence is $y = (y_1, y_2, \ldots, y_N)$

• RNNs compute the sequence of hidden vectors h as :

$$h_t = H(W_x h\, x_t + W_{hh}\, h_{t-1} + b_h)$$

$$y_t = W_{hy}\, h_t + b_y$$

where W are the weights, b are the bias vectors and H is the nonlinear function.

Downloaded by ribhav soni (ribhavsoni19@gmail.com)

- In speech recognition, information regarding future context of speech is equally important as the past context. Bidirectional RNN (BiRNNs) process the input vector in both forward and backword directions and finds hidden state vector for each direction. That is why instead of using unidirectional RNN, bidirectional RNN are widely used for speech recognition.
- Only frame-wise classification of audio input signal can be performed using both feed forward and recurrent neural networks. So force alignment between input audio and corresponding transcribed output is needed. This can be done using Hidden Markov Models (HMM) or Connectionist Temporal Classification (CTC) loss.
- CTC is an objective function. Alignment between the input speech signal and the output sequence of the words is computed using CTC.

6.4 : Recommender System

Q.18 What is recommender systems ? Explain in detail.

Ans. : • Recommender systems are a way of suggesting like or similar items and ideas to a users specific way of thinking. Recommender systems are widely used on the Web for recommending products and services to users

- Recommender systems try to automate aspects of a completely different information discovery model where people try to find other people with similar tastes and then ask them to suggest new things.
- The goal of a recommender system is to generate meaningful recommendations to a collection of users for items or products that might interest them. Suggestions for books on Amazon, or movies

This document is available on studocu

on Netflix, are real-world examples of the operation of industry strength recommender systems.

- Fig. Q.18.1 shows recommendation systems concept.

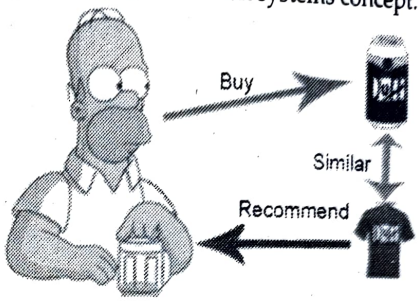


Fig. Q.18.1 Recommendation systems

- Recommendation systems are a key part of almost every modern consumer website. The systems help drive customer interaction and sales by helping customers discover products and services they might not ever find themselves.
- Recommender systems predict the preference of the user for these items, which could be in the form of a rating or response. When more data becomes available for a customer profile, the recommendations become more accurate.

There are a variety of applications for recommendations including movies (e.g. Netflix), consumer products (e.g., Amazon or similar on-line retailers), music (e.g. Spotify), or news, social media, online dating and advertising.

Recommendation process :

- Every recommendation system follows a specific process in order to produce product recommendations.
- The recommendation approaches can be classified based on the information sources they use. Three possible sources of information can be identified as input for the recommendation

Downloaded by ribhav soni (ribhavsoni19@gmail.com)

process. The available sources are the user data (demographics), the item data (keywords, genres) and the user-item ratings.

1. **Collection** : Data collected can be explicit (ratings and comments on products) or implicit (page views, order history, etc.).
2. **Storing :** The type of data used to create recommendations can help user decide the kind of storage we should use - NoSQL database, object storage, or standard SQL database.
3. **Analyzing :** The recommender system finds items with similar user engagement data after analysis.
4. **Filtering :** This is the last step where data gets filtered to access the relevant information required to provide recommendations to the user. To enable this, user will need to choose an algorithm suiting the recommendation system.

Q.19 What are challenges of recommender system ?

Ans. : Following are the challenges for building recommender systems :

1. Huge amounts of data, tens of millions of customers and millions of distinct catalog items.
2. Results are required to be returned in real time.
3. New customers have limited information.
4. Old customers can have a glut of information.
5. Customer data is volatile.

Q.20 Explain various types of recommender system ?

Ans. : In general, there are three types of recommender system :

1. Collaborative recommender system is a system that produces its result based on past ratings of users with similar preferences.
2. Content based recommender system is a system that produces its result based on the similarity of the content of the documents or items.

3. Knowledge based recommender system is a system that produces its result based on additional and means-end knowledge.
4. Demographic based recommender system : This type of recommendation system categorizes users based on a set of demographic classes. This algorithm requires market research data to fully implement. The main benefit is that it doesn't need history of user ratings
5. Hybrid recommender systems combine various inputs and different recommendation strategies to take advantage of the synergy among them.

Q.21 What are advantages and disadvantages of collaborative filtering ?

Ans. : Advantages

1. Collaborative filtering application is to recommend interesting or popular information as judged by the community.
2. Collaborative filtering system can make more personalized recommendation by analyzing information from your past activity or the history of other users of similar taste.

Disadvantages

1. Many commercial recommender systems are based on large datasets. As a result, the user-item matrix used for collaborative filtering could be extremely large and sparse, which brings about the challenges in the performances of the recommendation.
2. As the numbers of users and items grow, traditional CF algorithms will suffer serious scalability problems.
3. Gray sheep refers to the users whose opinions do not consistently agree or disagree with any group of people and thus do not benefit from collaborative filtering.

Downloaded by ribhav soni (ribhavsoni10@gmail.com)

4. A collaborative filtering system doesn't automatically match content to one's preferences.

Q.22 Explain types of collaborative filtering ?

Ans. : • There are two types of collaborative filtering algorithms : User based and item based.

1. User based

- User-based collaborative filtering algorithms work off the premise that if a user (A) has a similar profile to another user (B), then A is more likely to prefer things that B prefers when compared with a user chosen at random.
- The assumption is that users with similar preferences will rate items similarly. Thus missing ratings for a user can be predicted by first finding a neighborhood of similar users and then aggregate the ratings of these users to form a prediction.
- The neighborhood is defined in terms of similarity between users, either by taking a given number of most similar users (k nearest neighbors) or all users within a given similarity threshold. Popular similarity measures for CF are the Pearson correlation coefficient and the Cosine similarity.
- For example, a collaborative filtering recommendation system for television tastes could make predictions about which television show a user should like given a partial list of that user's tastes (likes or dislikes).
- Note that these predictions are specific to the user, but use information gleaned from many users. This differs from the simpler approach of giving an average score for each item of interest, for example based on its number of votes.
- User-based CF is a memory-based algorithm which tries to mimics word-of-mouth by analyzing rating data from many individuals.

This document is available on

- The two main problems of user-based CF are that the whole user database has to be kept in memory and that expensive similarity computation between the active user and all other users in the database has to be performed.

2. Item-based collaborative filtering

- Item-based CF is a model-based approach which produces recommendations based on the relationship between items inferred from the rating matrix. The assumption behind this approach is that users will prefer items that are similar to other items they like.
- The model-building step consists of calculating a similarity matrix containing all item-to-item similarities using a given similarity measure. Popular are again Pearson correlation and Cosine similarity. All pair-wise similarities are stored in $n \times n$ similarity matrix S.
- Item-based collaborative filtering has become popularized due to its use by YouTube and Amazon to provide recommendations to users. This algorithm works by building an item-to-item matrix which defines the relationship between pairs of items.
- When a user indicates a preference for a certain type of item, the matrix is used to identify other items with similar characteristics that can also be recommended.
- Item-based CF is more efficient than user-based CF since the model is relatively small ($N \times k$) and can be fully pre-computed. Item-based CF is known to only produce slightly inferior results compared to user-based CF and higher order models which take the joint distribution of sets of items into account are possible. Furthermore, item-based CF is successfully applied in large scale recommender systems (e.g., by Amazon.com).

studocu
Downloaded by ribhav soni (ribhavsoni19@gmail.com)

Q.23 Draw and explain architecture of content based recommendation of documents and products.

Ans. : • Content - based recommenders refer to such approaches, that provide recommendations by comparing representations of content describing an item to representations of content that interests the user. These approaches are sometimes also referred to as content - based filtering.

- Content - based recommendation systems try to recommend items similar to those a given user has liked in the past.
- In a movie recommendation application, a movie may be represented by such features as specific actors, director, genre, subject matter, etc.
- The user's interest or preference is also represented by the same set of features, called the user profile.
- Recommendations are made by comparing the user profile with candidate items expressed in the same set of features. The top-k best matched or most similar items are recommended to the user.
- The simplest approach to content - based recommendation is to compute the similarity of the user profile with each item.

High level architecture content-based recommender systems

- Fig. Q.23.1 shows high level architecture content-based recommender systems. (See Fig. Q.23.1 on next page.)

1. Content Analyzer

- Extracts the features (keywords, n-grams) from the source
- Conversion from unstructured to structured item
- Data stored in the repository represented items

2. Profile Learner

- To build user profile
- Updates the profile using the data in Feedback repository

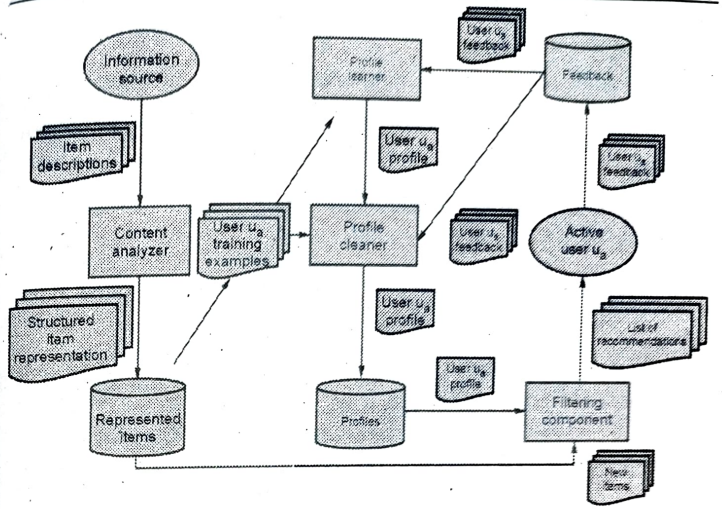


Fig. Q.23.1 High level architecture content-based recommender systems

3. Filtering Component

- Matching the user profile with the actual item to be recommended
- Uses different strategies
- Users have no detailed knowledge of collection makeup and the retrieval environment. Most users often need to reformulate their queries to obtain the results of their interest.

Q.24 Explain advantages and disadvantages of content based filtering.

Ans. : Advantages :

1. User Independence : Recommends only the items that interest the user
2. Transparency : Recommendation is based on the item features, explicitly list the contents features
3. New Item : Helps in recommending new items that are not yet rated by other users.

Downloaded by ribhav soni (ribhavsoni19@gmail.com)

Drawbacks :

1. The user will never be recommended for different items.
2. Business cannot be expanded as the user does not try a different type of product.
3. Overspecialization : Recommends those items that score high with the user profile.
4. Cold Start Problem : For a new user, systems don't have historical information to recommend items.

Q.25 Explain difference between collaborative filtering and content based filtering.

Ans. :

Sr. No.	Collaborative Filtering	Content Filtering
1.	Collaborative-Filtering systems focus on the relationship between users and items.	Content-based systems focus on properties of items.
2.	**Example :** Netflix movie recommendations	**Example :** Pandora.com music recommendations
3.	**Pro :** Does not assume access to side information about items	**Con :** Assumes access to side information about items
4.	Cannot recommend new items	It can recommend new items
5.	Item features are inferred from ratings.	Match the item features with user preferences.
6.	**Con :** Does not work on new items that have no ratings	**Pro :** Got a new item to add ? No problem, just be sure to include the side information.

This document is available on studocu

6.5 : Natural Language Processing

Q.26 What is natural language processing ?

Ans. : • Natural Language Processing (NLP) describes the interaction between human language and computers. It is a technology that many people use daily and has been around for years, but is often taken for granted.

- A few examples of NLP that people use every day are : Spell check, autocomplete, voice text messaging, spam filters, related keywords on search engines and Siri, Alexa, or Google Assistant

Q.27 Explain phases of natural language processing.

Ans. : • NLP has two phases 1) Data preprocessing and 2) Algorithm development.

- **Data preprocessing :** In preprocessing text data is features in text data are highlighted so as to make it suitable to analyze and process by machine preprocessing can be done by, tokenization, stop word removal, lemmatization and speech tagging.
- **Algorithm development :** After preprocessing the data, NLP algorithm is developed to process it. Two main types of algorithms used for NLP are,

1. **Rule based system :** It uses carefully designed linguistic rules of a language.
2. **Machine learning based system :** It uses statistical methods. Models learns to perform the task from the training data provided. NLP algorithms can design their own rules by using combination of machine learning, neural network and deep learning through repeated processing and learning.

Q.28 Explain convolutional neural network based framework for NLP.

Ans. : • CNN can be used to constitute words or n-grams for [illegible] features. Fig. Q.28.1 shows the CNN based

Downloaded by ribhav soni (ribhavsoni19@gmail.com)

framework. Here the words are transformed into vector representation through look-up table. This results in word embedding approach where weights are learned during training of network.

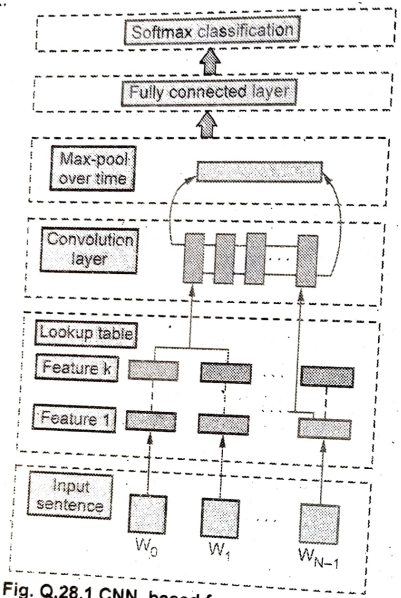


Fig. Q.28.1 CNN based framework for NLP

- The steps to perform sentence modeling with CNN are as follows :

1. Sentences are tokenized into words. Then it is further transformed into word embedding matrix of dimension 'd' . This forms the input embedding layer.
2. Convolutional filters are applied to this input layer of word embeddings to produce feature map.
3. Then max pooling is applied to each filter. This reduces the dimensionality of the output and produce fixed length output. Thus the final sentence representation is created.

- The drawback of CNN is that it cannot handle long distance contextual information and also inefficient in preserving sequential order of context. Therefore Recurrent Neural Networks (RNN) are more suitable for this.

Q.29 Explain recurrent neural network based framework for NLP.

Ans. : • RNN are effective for sequential data processing. In RNN computation is recursively applied to each instance of input sequence from previous computed results. Recurrent unit is sequentially fed with the sequences represented by fixed size vector of tokens. RNN based framework is shown in Fig. Q.29.1.

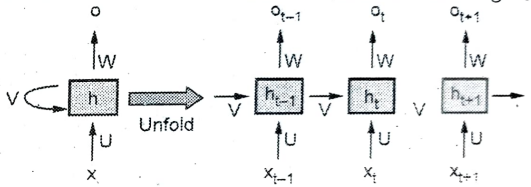


Fig. Q.29.1 RNN based framework for NLP

- The advantage of RNN is that it can memorize the results of previous computation and utilize that information in current computation. So it is possible to model context dependencies in inputs of arbitrary length with RNN and proper composition of input can be created.
- Mainly RNNs are used in different NLP tasks like,
  1. Natural language generation (e.g. image captioning, machine translation, visual question answering)
  2. Word - level classification (e.g. Named Entity Recognition (NER))
  3. Language modeling
  4. Semantic matching
  5. Sentence-level classification (e.g., sentiment polarity)

END...

Downloaded by ribhav soni (ribhavsoni19@gmail.com)