CO4

DEEP LEARNING

23AD2205A

Topic:

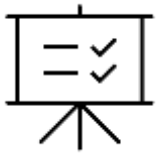
VANISHING AND EXPLODING GRADIENTS IN RNNS, SEQUENCE PREDICTION WITH RNNS

Session - 21

AIM OF THE SESSION

To familiarize students with the sequence prediction problems

INSTRUCTIONAL OBJECTIVES



This Session is designed to:

1. Discuss the Contractive Autoencoders and Variational autoencoder
2. Demonstrate the concept of Contractive Autoencoders and Variational autoencoder
   Discussion on Contractive Autoencoders and Variational autoencoder

LEARNING OUTCOMES

At the end of this session, you should be able to: concepts for real time applications

1. To build Contractive Autoencoders and Variational autoencoder
2. To apply different types of Contractive Autoencoders and Variational autoencoder

Issues of Standard RNNs

Vanishing Gradient:

- Text generation, machine translation, and stock market prediction are examples of time-dependent and sequential data problems modeled with RNNs.
- Training RNNs is difficult due to the vanishing gradient problem, where gradients decay over time.

Exploding Gradient:

- Occurs when the gradient grows exponentially during training.
- Large error gradients cause very large updates to the neural network model weights, leading to instability.

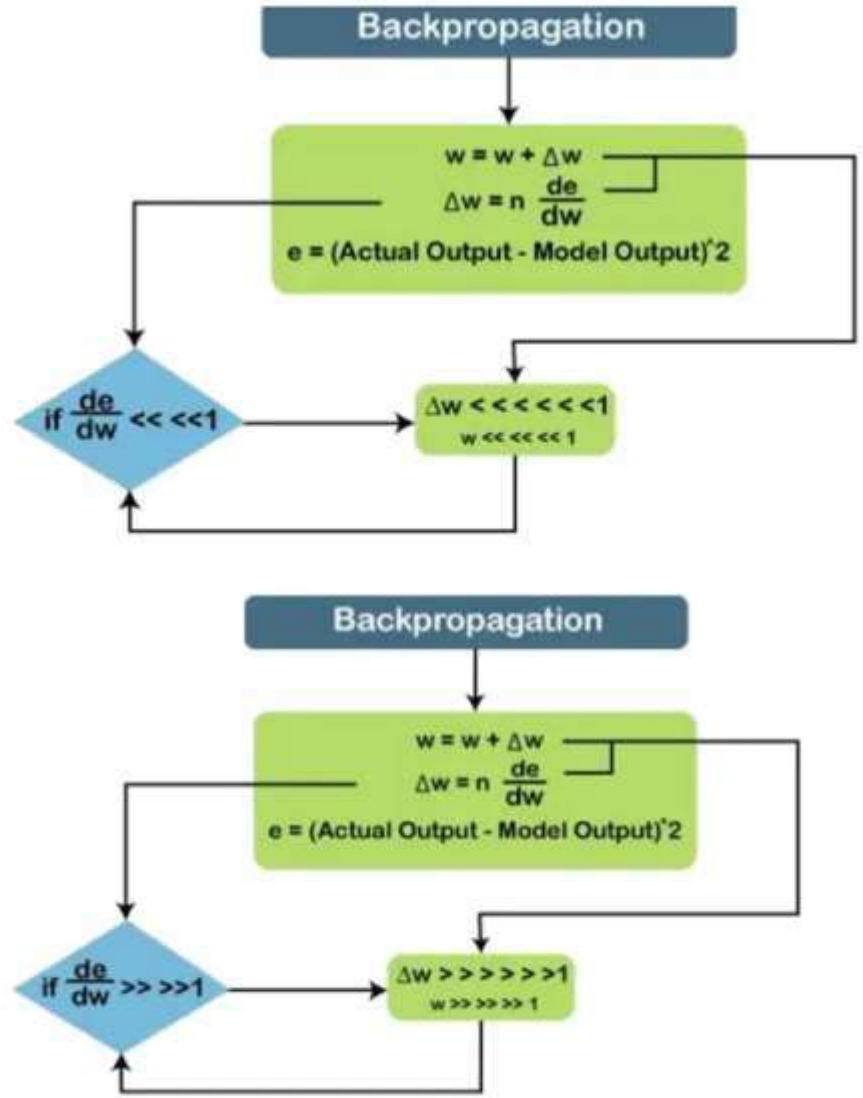
How to identify a vanishing or exploding gradients problem?

Vanishing

- **Weights** of earlier layers can become **0**.
- **Training stops** after a few iterations.

Exploding

- **Weights** become unexpectedly **large**.
- **Gradient** value for error persists over **1.0**.

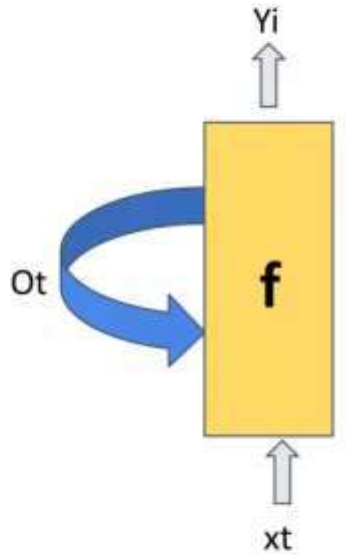


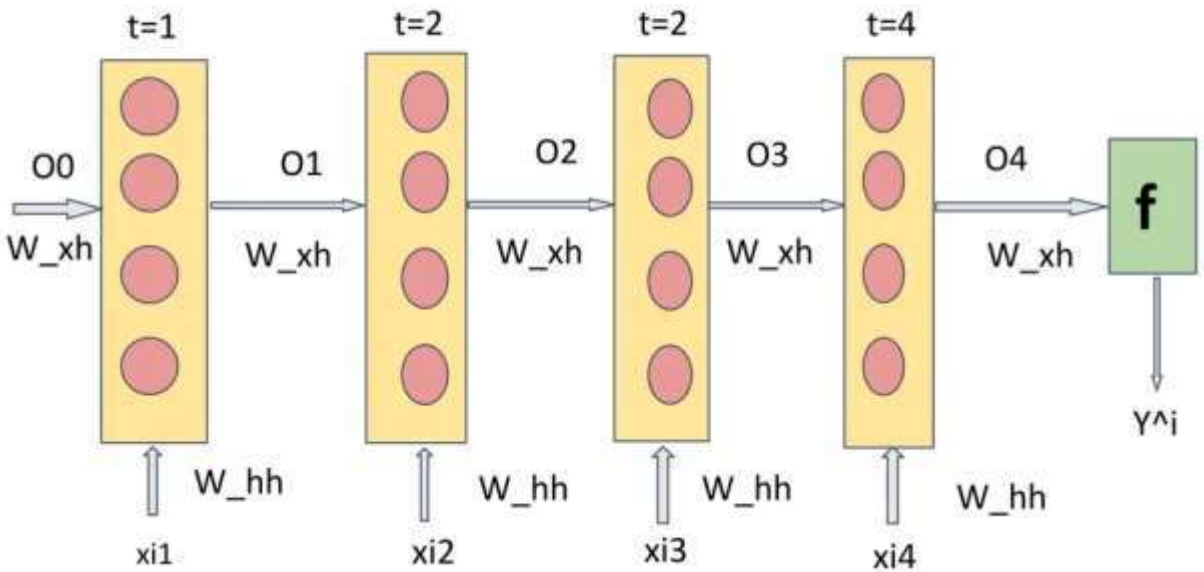
Understanding Sequence Prediction with RNNs
Training through RNN

- A single-time step of the input is provided to the network.
- Then calculate its current state using a set of current input and the previous state.
- The current ht becomes ht-1 for the next time step.
- One can go as many time steps according to the problem and join the information from all the previous states.
- Once all the time steps are completed the final current state is used to calculate the output.
- The output is then compared to the actual output i.e the target output and the error is generated.
- The error is then back-propagated to the network to update the weights and hence the network (RNN) is trained using Backpropagation through time.

Steps for training a RNN

- Initial input is sent with same **weight** and **activation function**.
- Current state calculated by using **current input** & **previous state output**
- Current state **Xt becomes Xt-1** for second time step.
- Keeps on repeating for all the steps
- Final step calculated by current state of final state and all other previous steps.
- An error is generated by calculating the difference between the **actual output** and **generated output** by RNN model.
- Final step is when the process of back propagation occurs





Recurrance formula

$$h_t = f_w(h_{t-1}, x_t)$$

ht= new hidden state
fw= some functions of parameter w
ht-1= old state
xt= input vector at some time spent

$$O1 = f(Xi1w\_hh + O0W\_xh) \qquad O3 = f(Xi3W\_hh + O2W\_xh)$$
$$O2 = f(Xi2w\_hh + O1W\_xh) \qquad O4 = f(Xi4W\_hh + O3W\_xh)$$

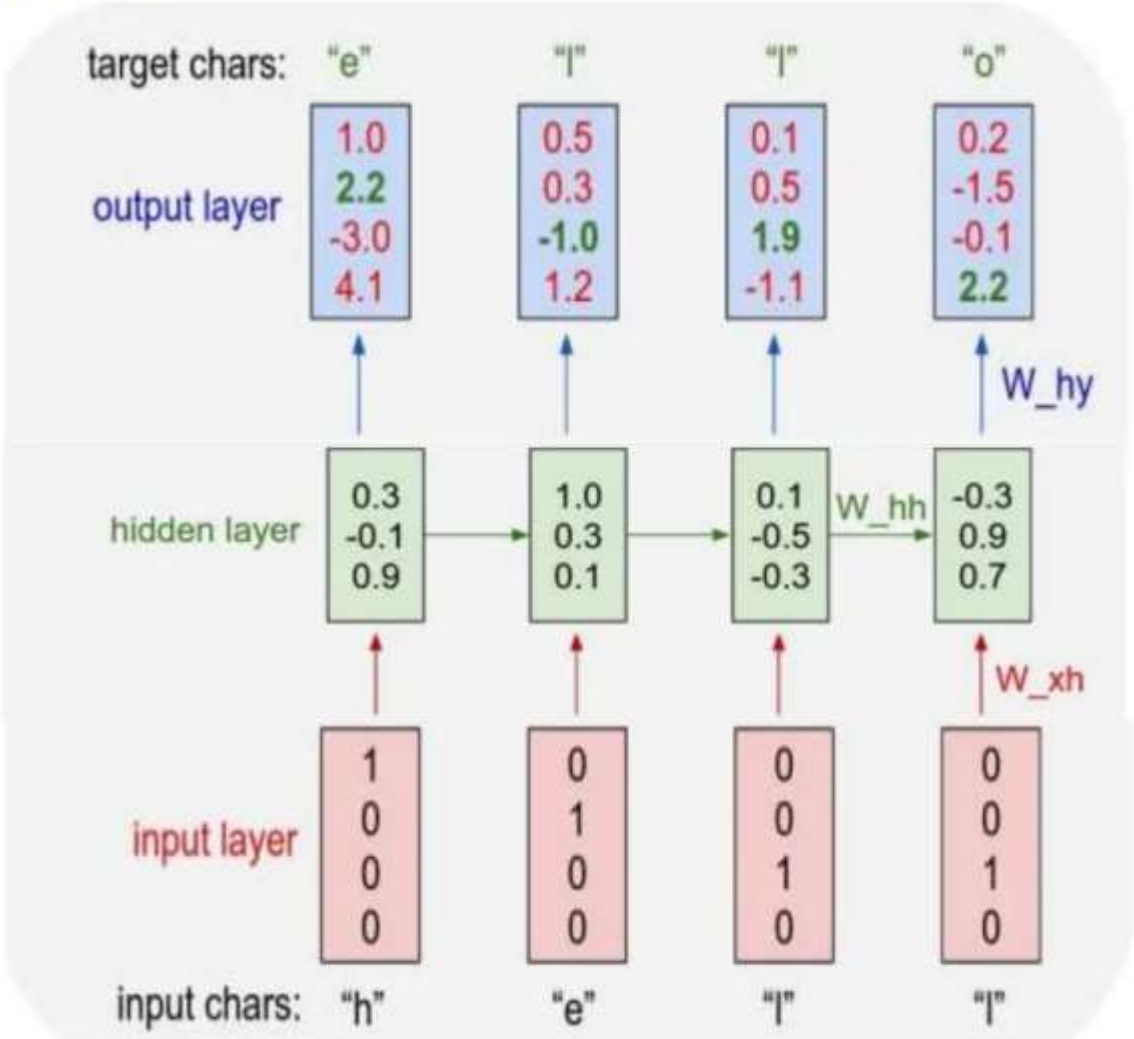
Example: Character-level Language Model

Vocabulary: [h,e,l,o]

Example training sequence: “hello”

$$h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t + B_h)$$

$$y_t = W_{hy}h_t + B_y$$



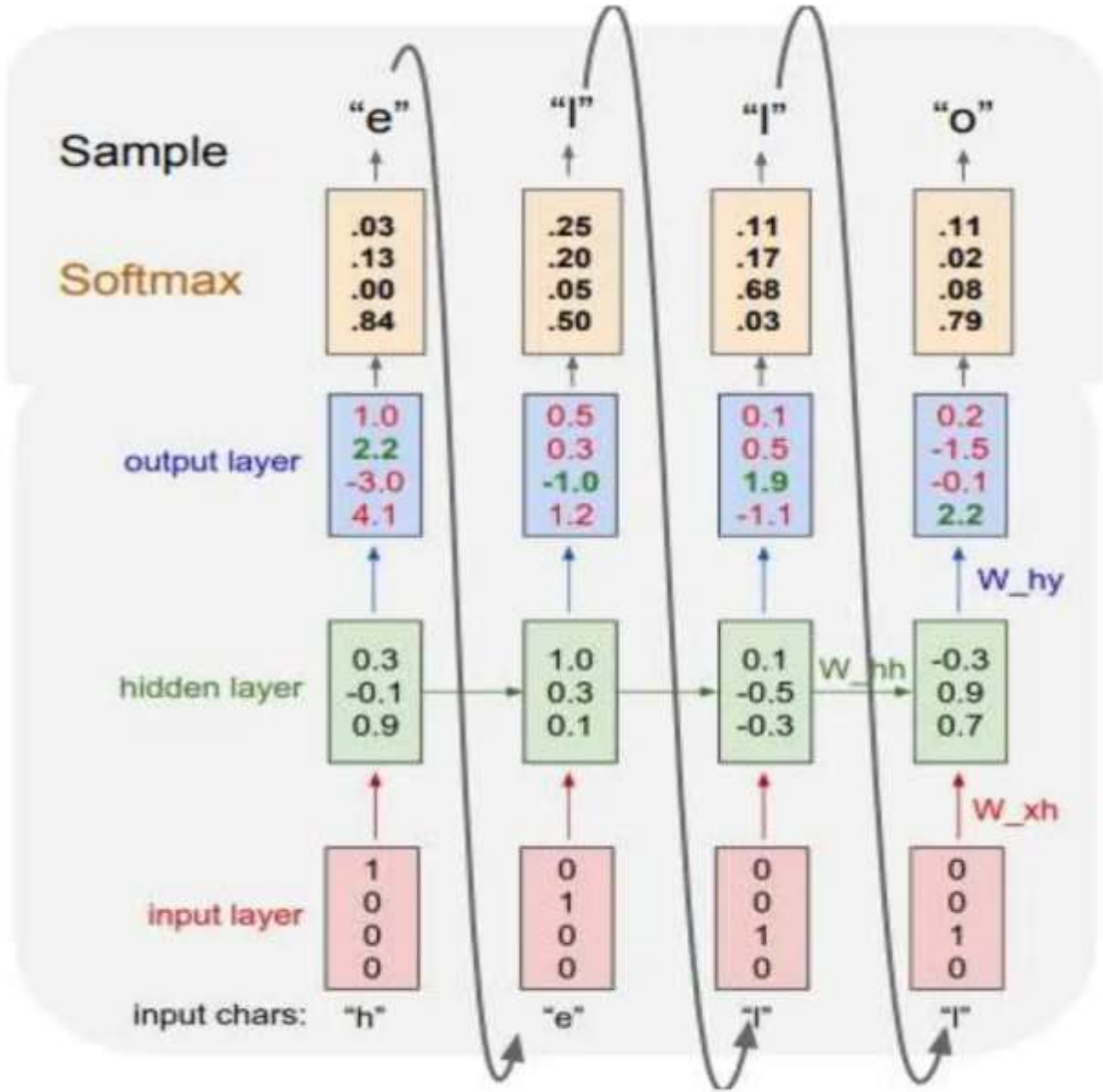
Continued…

Vocabulary: [h,e,l,o]

At test-time sample characters

one at a time,

feed back to model



Back Propagatipon

To reduce lose function derivative of y^i

∂L/∂y^i

By Chain rule W_xh is dependent on y^i, **∂L/∂y^i**

∂L/∂w_xh= (∂L/∂y^i * ∂y^i/∂w_xh)

Weight Updation,

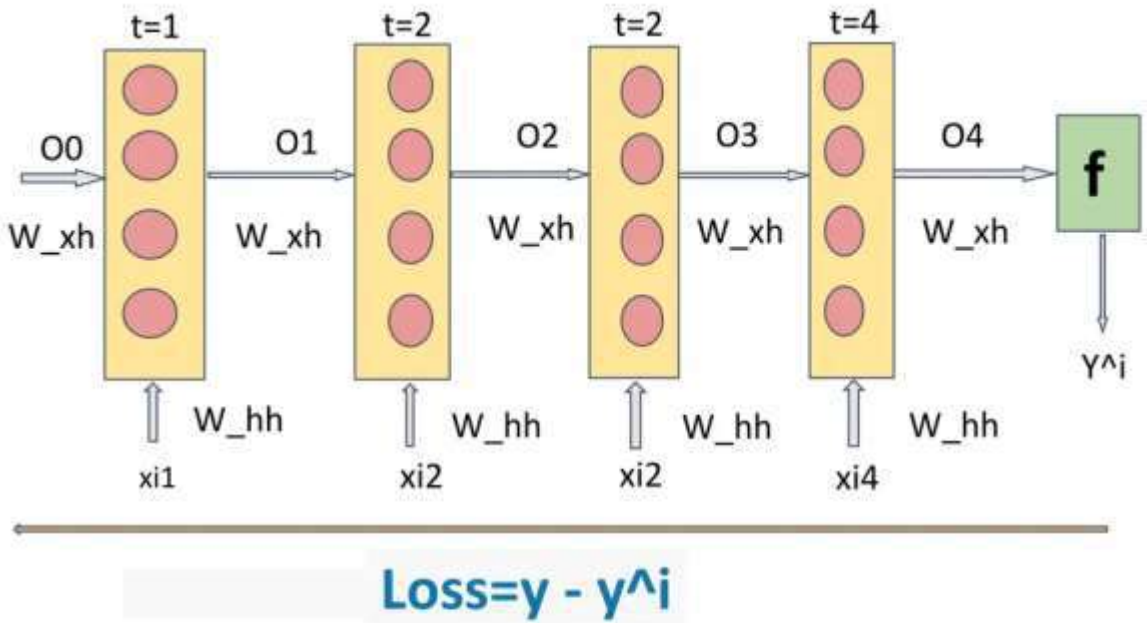
W_hh_new= W_xh – ∂L/∂w_xh

Weight Updation W_xh w.r.t O3 in Backward Propagation at time t3

By Chain Rule O4 is dependent on W_hh, y^i dependent on O4, loss is dependent on y^I, **∂L/∂y^**

∂L/∂w_xh= (∂L/∂y^i * ∂y^i/∂O4 * ∂O4/∂w_hh)

W_new_hh=W_xh – (∂L/∂y^i * ∂y^i/∂O4 * ∂O4/∂w_hh)



Advantages and Disadvantages of RNN

Advantages

- An RNN remembers each and every piece of information through time.
- It is useful in time series prediction only because of the feature to remember previous inputs as well. This is called Long Short Term Memory.
- Recurrent neural networks are even used with convolutional layers to extend the effective pixel neighborhood.

Disadvantages

- Gradient vanishing and exploding problems.
- Training an RNN is a very difficult task.It cannot process very long sequences if using tanh or relu as an activation function.

Applications of RNN

- Language Modelling and Generating Text
- Speech Recognition
- Machine Translation
- Image Recognition, Face detection
- Time series Forecasting
- Variation Of Recurrent Neural Network (RNN)

To overcome the problems like vanishing gradient and exploding gradient descent several new advanced versions of RNNs are formed some of these are as;

- Bidirectional Neural Network (BiNN)
- Long Short-Term Memory (LSTM)
- Bidirectional Neural Network (BiNN)

Conclusion

- RNN SummaryRecurrent Neural Networks (RNNs) are ideal for handling sequential data, effectively capturing dependencies like those in language modeling and time-series prediction. However, they face challenges such as vanishing and exploding gradients, often requiring advanced variants like LSTM and GRU for improved performance.

- BPTT SummaryBackpropagation Through Time (BPTT) enables RNNs to learn from sequential data by updating weights based on past inputs, capturing long-term dependencies. Despite its importance, BPTT can be computationally intensive and prone to gradient issues, which advanced techniques like LSTM

SELF-ASSESSMENT QUESTIONS

Recurrent Neural Networks (RNNs) are designed to handle _____ data, making them suitable for tasks like language modeling and time-series prediction.

sequential

RNNs can face challenges such as _____ gradients, which can complicate training and limit their effectiveness in capturing long-term dependencies.

vanishing and exploding

Backpropagation Through Time (BPTT) helps RNNs learn from sequential data by updating _____ based on past inputs.

dependencies

TERMINAL QUESTIONS

Describe how the hidden state in Recurrent Neural Networks (RNNs) contributes to handling sequential data.

Explain why RNNs are more effective than feedforward neural networks for tasks like language modeling and time-series prediction.

Illustrate how different types of RNNs (One to Many, Many to One, Many to Many) can be applied to specific real-world problems.

Analyze the impact of using a Recurrent Unit instead of a traditional neuron in processing sequential inputs.

Compare the advantages and limitations of standard RNNs with Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU).

REFERENCES FOR FURTHER LEARNING OF THE SESSION

Books:

1 Ian Goodfellow and Yoshua Bengio and Aaron Courville (2016) Deep Learning Book
2.Deep Learning Book. eep Learning with Python, Francois Chollet , Manning publications, 2018

Resources

- https://www.tensorflow.org/tutorials/generative/autoencoder
- https://www.linkedin.com/company/autoencoder?originalSubdomain=in
- https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-are-autoencoders-in-deep-learning
- https://blog.keras.io/building-autoencoders-in-keras.