To IITD-AIA FSM

Subject: Weekly progress report (week-0)

Dear Sir,

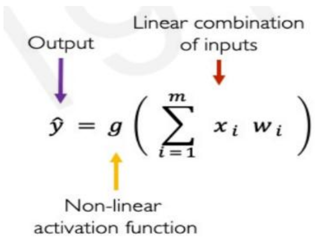
In this I have done the learning task given to me and also implemented the learnings in python for practice.

Topics covered-

- TensorFlow, Gradient Descent, Loss function
- RNN, Transformers
- Music Generation from ABC notation
- Computer vision, CNN

June-1   MIT Introduction to Deep Learning | 6.S191

Started with Basics of deep learning. It is useful in learning the underlying features directly from data. **TensorFlow** Library is used for implementing the neural network in this course. Perceptron in Deep learning, it is the basic model for biological neuron and is used for supervised algorithm (Binary classifier). Learnt about Mathematical formula of perceptron and how to code it in python from scratch.


Output
Linear combination of inputs
$\hat{y} = g\left(\sum_{i=1}^{m} x_i w_i\right)$
Non-linear activation function

Activation Functions are used to implement non linearities in the neural network.

- **Sigmoid (tf.math.sigmoid)**
- **Tanh(tf.math.tanh)**
- **Relu (tf.nn.relu)**

Multiple perceptron and layers are connected to form the deep neural network. To train this network, we need to minimize the loss function.

- **Cross entropy loss** can be used with model that output a probability between 0 and 1 (softmax_cross_entropy_loss) function
- **Mean squared error** can be used with regression models that output continuous real numbers.

Main job is to find optimum weights which will have minimum loss function. Gradient descent algorithm can be used to find the optimum weight. Find the gradient with the help of back propagation and update the weight with Learning rate. We need to have stable learning rates converges smoothly and avoid local minima by using the adaptive learning method. Gradient descent algorithm are

- **SGD** (tf.keras.optimizers.SGD)
- **Adam (**tf.keras.optimizers.Adam)
- **Adadelta** (tf.keras.optimizers.Adadelta)

Gradient descent algorithm is computationally expensive. 2 other methods which can be used are

- **Stochastic Gradient descent** – Easy to compute but noisy
- **Batch gradient descent** - More accurate, smooth convergence and allows for larger learning rates.

To overcome the problem of overfitting, we can use the regularization method

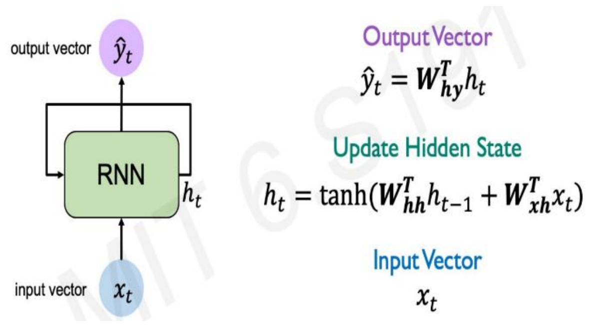
- **Dropout** (randomly set some activation at 0)
- **Early stopping**

MIT Deep learning course has Lab exercise which taught the basics of implementation of Neural network. Link for the

GitHub- https://github.com/Kush-2103/Deep_learning/blob/main/Lab1(TF).ipynb

June 2 Recurrent Neural Networks, Transformers, and Attention

In this lecture, I have learnt about RNNs, And transformers. Perceptron and feed forward neural network do not have the application of sequential data. Neurons with recurrence. h(variable) represents internal state which is passed to next network. Recurrent Neural Network (RNN) is a type of Neural Network where the output from the previous step is fed as input to the current step.



Design criteria for sequence modelling:

- Handle variable length sequences
- Long term dependencies
- Maintain order
- Share parameters across sequence

RNNs meet all these criteria

Neural network does not interpret words, we need numerical inputs. (Next word prediction) That is why we use embeddings. Embedding means transform indexes into a vector of fixed size.

- **One hot embedding**
- **Learned embedding**

Back propagation algorithm is difficult to implement in RNN. Because while finding the gradients we encountered with 2 problems

- **Vanishing gradient** – Many Values are less than 1, so if we multiply these it will get decreased very fast. To solve it, we can use Relu Activation function
- **Exploding Gradient**- To solve it, we can clip the gradient.

Lecture also briefly talked about **LSTM (Long Short-Term Memory).** These are able to track many information throughout many timestamps. Forget Store, Update, Output are the 4 cells which used for flow of information

Limitation of RNNs-

- Encoding Bottleneck
- No parallelization
- Not long memory

To overcome the limitation, we use **SELF ATTENTION** with neural networks. It is used to model sequences without recurrence.

Need to identify and attend to most important features in the input. Operations done for it for a self-attention head that can plug into a larger network. Each head attends to a different part of input. It is also used in BERT and GPT-3 models.

June 3 Music Generation with LLM

Course has the lab exercise of music generation with the help of RNN.

building a Recurrent Neural Network (RNN) for music generation. We will train a model to learn the patterns in raw sheet music in ABC notation and then use this model to generate new music.

I am using **a mitdeeplearning** library for using the inbuilt function by MIT . Course has given a Irish folk song.

We need to create a numerical representation of our text-based dataset. To do this, we generate two lookup tables: one that maps characters to numbers, and a second that maps numbers back to characters.

RNN model is based on LSTM. It will have 3 layers-

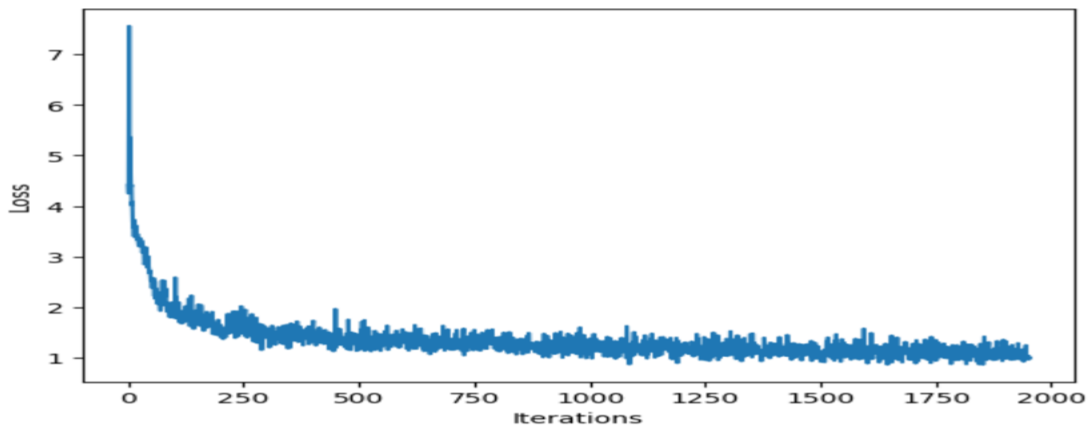
1) **tf.keras.layers.Embedding**: This is the input layer, consisting of a trainable lookup table that maps the numbers of each character to a vector
with embedding_dim dimensions.
2) **tf.keras.layers.LSTM:** Our LSTM network, with size units=rnn_units.
3) **tf.keras.layers.Dense:** The output layer, with vocab_size outputs.

I have used the **sparse_categorical_crossentropy** loss. we will instantiate a new model and an optimizer. Then, we will use the **tf.GradientTape** method to perform the backpropagation operations. I have used **Adam Optimizer.**

Initialize a "seed" start string and the RNN state, and set the number of characters we want to generate.

Google Colab Code-
https://colab.research.google.com/drive/1M3eZbfhbhvIV6wQig7C0_HB1zI4VXRHY

LOSS Function for the above model of batch size=4



June 4 Convolution Neural Networks

This lecture starts with applications of Computer vision. It ranges from facial detection, autonomous driving to medical applications.

Learnt about the feature extraction for image classification. search for particular features in an image to identify its class. There are various problems in manual feature extraction such illumination variance, view point variant, scale variation, deformation etc. That's why we need automatic feature extractor model, CNN can help us in this.

If we use fully connected neural network for image processing then we will not have spatial information because we flatten the image and also, we will have lot of parameters which we will not make the model scalable. we can use the spatial information by patches, Convolution operation. It helps in feature extraction by adding the weights to patch.

Filters can help us in the edge detection, sharpening the image.

CNN can help us in the classification of the images too. 3 steps which constitutes the CNN are-

- **Convolution**: Apply filters to generate maps.
- **Non**-Linearity: ReLu is mostly used for this operation.
- **Pooling**: Down sampling operation. Various types of pooling can be used such as max pooling, mean pooling etc.

Above 3 can be used for Feature extraction. For image classification we need to do 2 more steps

- **Flatten the layers**
- **Dense and SoftMax operation**

For object detection CNN can have too many regions and it will be slow. So, we can use

- R-CNN (we select the region) and
- Faster R-CNN (Propose the learning region) for efficient object detection. It uses the warped function

For Image segmentation we can use **Fully Convolution Network (FCN).** Network Design with all convolutional layers, down sampling, and up sampling operations.

I have practice image classification model using FCN, CNN on MNIST dataset

Code->
https://colab.research.google.com/drive/1_xcgDDTRG26ej_UwuYQbt8ns335H8WvI#scrollTo=WF0s06HumAIN

Fully connected NN accuracy- 96.2%

CNN accuracy- 98.1%