A Study on Deep Learning

Sanket Ninawe[*]
Indiana University
999 Hidden Lake Dr
New Jersey, USA
sninawe@iu.edu

ABSTRACT

This study provides basic ideas behind deep learning. A theoretical and practical study of concepts of regular feed-forward neural networks used for Deep Learning is done here. A brief introduction of typical algorithms used in Deep Learning along with implementations using packages like tensorflow and keras is done here. The course was a mixture of both theory and practical knowledge.

Keywords

Deep Learning; tensorflow; regression

1. INTRODUCTION

As an introduction to the course we went through what is machine learning, what are machine learning models and types of machine learning (supervised, unsupervised and semi-supervised). We also talked about regression and classification types of supervised learning algorithms. We choose the most basic machine learning algorithm - Linear Regression and saw how it is used to solve a problem statement. In this section we also introduced concepts of cost or loss and its optimization. In the assignment we worked on using Gradient Descent Algorithm to minimize the cost function and calculate the optimum weights.

[*]

2. EXPLORATION OF DEEP NETWORK

In this section we studied the concepts of Neurons and neural network which are the building blocks for a Deep Network. We first saw what a biological neuron is and based on the concept how an artificial neuron is conceptualized. We then introduced on how inputs of a machine learning problem are combined and how weight for each neurons can be used to store information through learned concepts. We also saw how addition of bias simplifies learning and simplifies mathematical representation. These bias terms also help in storing the information learned during the training of the model. A single neuron can be said to be a linear model with a well defined linear equation.

Next we defined what is an activation function is and based how an activation function is used what are the categories or types of activation functions. Activation functions are nothing but an output from a neuron. It can be an intermediate output or can also be the final output of the Neural network. An activation function simply transforms the output of a neuron into a form which is more suitable to the task in hand.

Neuron can be categorized into following types -
1. Linear Neurons(it is just a straight line passing through the origin have a slope of 1)
2. Binary Threshold Neurons(these produce binary decisions based on the threshold values provided)
3. Rectified Linear Neurons(is a non-linear in terms of function and is above 0 when it is linear, whereas at and below 0 it makes a hard decision)
4. Sigmoid Neuron(it tends to 1 for very high values function as z$\rightarrow \infty whereas it tends to 0 as z \rightarrow -\infty$
$5. tanh Neuron(A scaled Sigmoid function)$

As part of assignment we discussed the advantages and disadvantages of Sigmoid Neurons.

2.1 Learning with Neurons

Here we discussed on how a neuron stores information and how corresponding weights can change the output of a neuron. The limitations of preceptron is that is can classify only linearly separable points and thus it cannot learn linearly inseparable points. Then we learned algorithms which will help us learn weights pragmatically like Preceptron learning algorithm(based on misclassification error, used for binary classification) and Gradient Descent algorithm(iterative optimization technique, which it does initially by guessing the value of the parameter randomly and then adjusting it continuously until it converges to the minimum).

PLA(Perceptron Learning Algorithm) -
1. Choose a random w initially.
2. Use this to classify the samples.
3. Randomly choose a misclassified sample
4. Update the weight vector as follows:
$\mathrm{w}_{new} \leftarrow w_{old} + y_p x_p$
5.*Repeat until all points are samples are correctly classified.*

We also studied stochastic gradient descent (SGD) where we choose cost function on all the trainning samples, only is chosen to update the weight. This also will converge and thus this can be used on large training data and learning can be online.

2.2 Vectorization and Backpropagation

The real power of a neural network comes from the fact that they can be stacked up together in a layer. Moreover many such layers can be stacked up horizontally to form an even larger network of interconnected network of neurons. Feed-forward neural networks is the most common architecture of neural network.
Some characteristics of feed-forward neural networks are:
1. The first and the last layers are always the input and the output layers respectively.
2. They may or may not contain hidden layers. If they contain more than one hidden layers, they are called deep neural networks.
3. The output layer can have a single neuron or multiple neurons.

There are two other architectures - Recurrent Neural Network (RNNs) and Convolutional Neural Networks (CNNs). RNNs are typically used for sequential data (e.g., a sentence as a word sequence) while CNNs are for images.

Vectorization is used to implement neural networks in modern deep learning framework. The vectorized implementation for calculating matrix multiplication of inputs and weights is more compact, and often faster than the naive implementation.

Backpropagation is an algorithm to compute gradient for training neural networks and is simply application of chain rules to composite function across various input, hidden and output layers of a Neural Network.

2.3 Logistic and Softmax algorithms

Logistic Regression is a linear machine learning model most suited for classification problems. It can be considered as a neuron with the logistic activation function and cross-entropy error as its cost function. The logistic function is given by the formula:

$$y = \frac{e^z}{1 + e^z} \tag{1}$$

It lies between 0 and 1 and hence satisfies one requirement of the probability distribution function.

Softmax algorithm for multiclass classification -

When we have a problem where we have more 2 class output logistic regression classification cannot be an efficient solution, so an algorithm which produces unified probability distribution is available called Softmax algorithm. Softmax function turns the scores produced by different models into a probability distribution across all these models. Then these probabilities can be used to see which class gets the highest probability and thus is likely to be the output class. One hot encoding is one of the which will produce output with 1 for the highest probability class and 0 for other classes. We typically use a deep learning framework to compute gradient internally and learn the parameters for softmax regression.

We also learned the vectorized notation of logistic and softmax algorithm. As part of the assignment, we implemented logistic regression on a 2D Gaussian Data using Stochastic gradient descent.

We were also introduced to tensorflow framework. An advantage of tensorflow API is that it provides a way to use SGD with backpropagation so you do not need to implement a code to obtain gradient. Moreover, tensorflow automatically optimizes the computation using modern CPUs/GPUs parallelization techniques so we can focus more on model development. Tensorflow is used for Low and Mid Level API's, while Keras is used on top of these libraries to for high level API development which involve building powerful deep network.

For the assignment we used tensorflow framework to find weights and bias for 5 Input, 3 hidden layer Neural Network, so that output produced by the last layer is 0.3.

2.4 Going Further Deep

In the section we studied the universality theorem which said that any particular function can be represented by a neural network even with just one hidden layer. We discussed why going deep is important as we now have developed deep networks which perform much better than shallow networks in solving real-world problems like image recognition and text classification. Ex: Recognizing a human face involves identifying different parts like eyes, nose, lips etc, which require multiple layers acting like subsystems to learn the input features and provide a correct output.

Next we looked at problem of vanishing or exploding gradient which training deep network. This happens because while we are using backpropagation gradient descent method to minimize the cost function for each layer in a deep network, the way the weights and biases are calculated in each of the layer is the same, but in practice it is seen that each layer in a deep network learns from its input with a different rate. Sometimes, the initial layers learn very slow and then learn nothing, this phenomenon is called vanishing gradient problem. On the other hand, we also see cases where initial layers learn very fast and later layers get stuck, this problem is called exploding gradient problem.

We also introduced a new cost function called cross entropy cost function.

$$C = -\frac{1}{n}\sum[y \ln a + (1-y)\ln(1-a)]$$

Two important properties of this function which makes it a better candidate for a cost function in a deep network where when the model makes big mistakes, it will learn faster.
1. C is always positive.
2. If the actual output and predicted output are close, the value of cost function tends to be 0 whereas if they are far apart, the value tends to 1.

We concluded that cross-entropy is better than mean squared error whenever we are using sigmoid activation function in the output layer.

In the assignment, we implemented a model adding hidden layers to the classification on MNIST data and reporting the accuracy and time of the model run. We saw that as we increased the hidden layers the accuracy decreased or remained same as expected because of vanishing gradient problem.

2.5 Practical Techniques and Tricks

We looked at overfitting problems for deep networks and how we can solve these problems in this section. Chances of overfitting is prominent in deep network because of huge number of parameters involved in the deep networks. To deal with the overfitting problem, we have regularization techniques like L2 regularization and drop-out regularization.

We also looked at techniques on how we can speedup the training of a deep network. Input normalization is one of the most reliable technique to speedup training. Weight Initialization is another technique to speedup the training. Important consideration while initilaization of weights are - Initializing weights to very large random values does not work well. It slows down training. Also, poor initialization can lead to vanishing/exploding gradients, which also slows down the optimization algorithm. There are some useful recommendations available based on the activation functions used.
Few other techniques for optimizing algorithms are -
1. Mini-batch gradient descent(used when we have huge input size)
2. Gradient descent with momentum(when we want to smoothen up the noise and move towards the goal state of minimum cost much faster)
3. Learning rate decay(reduces the learning rate as learning goes on)
4. RMSProp(we maintain exponentially weighted averages of the square of the gradients)
5. Adam optimization algorithms(combination of both momentum and RMSprop into one algorithm).

2.6 Computational graph and Representation learning

In this section, we saw what do we mean by computational graph and how we can represent Neural or Deep Network using computational graph. Computational graph helps visualize the backpropagation and chain rule for a deep network. We also saw the difference between 2 frameworks Tensorflow(Static graph) and Pytorch(Dynamic graph). We also saw how modern day GPU's are helping in expanding the Deep Networks.

Next we saw the representation learning where Cartesian coordinates which are not separable can be automatically separated using architecture into polar coordinates. Machine Learning algorithms will automatically represent the input data into polar coordinates in case we have a very complex representation.

In the assignment, we learn a dataset having circular coordinates using logistic and Feed Forward Neural Network. The logistic regression was able to provide only 0.5 accuracy, whereas Feed Forward network gave a higher accuracy greater than 0.9.

We also went through an example of representation learning for words which is based on linguistic theory called distribution hypothesis. The hypothesis is effectively combined with a neural network model called skip-gram, which is also known as word2vec. In this we studied the definition of distribution hypothesis, skip-gram model intuition and finally the skip-gram or word2vec model. We implemented the word2vec model using tensorflow and came up with the most adjacent words for a given word in the document dictionary.

3. CONCLUSIONS

This study in Deep Learning provided an opportunity to understand the basic concepts on which various components of a Deep Network are built. This course was very thorough in terms of theory and practical examples and was a very good learning experience. The level of details and content of the course is very well put forth and I found out very structured to meet needs of an individual who wants to start taking steps in the realm of Deep Learning.

4. ACKNOWLEDGMENTS

I truly acknowledge the support from Satoshi Tsutsui in guiding me with the assignments and the course content. I also thank Prof. Ying Ding in providing me the opportunity to work on this independent Study of Deep Learning.