I am expecting to build a Neural Network, which will have NLP capabilities, does Information Retrieval from a corpus, which can be linked to big-data and can be deployed into real-time production environment. Also wants to understand Text Classification.

How to prepare a real time system which will take feedback, and learns from the feedback.

AI learns patterns from data, in the form a formula, and relates two or more variables.

Supervised: which has input, and generates expected output. E.g. Prediction of house prices

Unsupervised: we will have input data and we won’t necessarily have output.

Classification: Bucket Images.

Auto Encoders:

Deep Learning is data driven, more the data major it is. It requires more data, it is hungry to data. The more data the efficient data it is given. It is best to do Feature Extraction.

Most of the heavy weight games handled by GPUS. All games are series of Images. Matrix Transformations.

CUDA is something enables parallel computing, So if we use Nvidia GPUs which have cuda cores , DL can leverage parallel computing feature of CUDA.

Speech Processing, Natural language Processing and

Where shld we conider to use basic machine learning over dl:

XGBOOST is better than Deep Learning. Tree Based Algorithms many times performed better than Deep Learning.

Where Deep Learning Beats other systems? Compute Vision

Payroll Translation Model from the raw data.

Can I to How can I

Regularization is used to avoid overfitting. We use (L1 , L2 Regularization techniques)

Overfitting is a scenario where the performance (error) is less on trained data and high on test data.

Increase the dataset size will decrease overfitting.

Model memories when the data is small size, and fails to generalize on unseen data. This is called Overfitting.

Instead of increasing Hidden layers, can we increase the nodes in a Hidden layer and can get better output?

How to decide the optimal Number of Hidden Layers? Any Thumb rule?

Neuron == Perceptron == Node

Input layer is features, and the hidden layers and output layers are neurons.

ANN –uses fully connected network

CNN—Images

RNN – TEXT, SPEECH PROCESSING

DNN – DEEP NEURAL NETWORK

DBN – DEEP BELIEF NETWORK

What Comprises of NN?

================================day2========================================

Weights and coefficients are equal in neural nets. Both are same.

We learnt coefficients in linear regression.

We can use these words interchangeably.

How to decide weights?

During the training, the perceptron will learn the optimum weights.

Perceptron takes a dot production between input and weights and passed to a summation function.

….The output of a dot product is a scalar value.

Input is a vector X1,X2…..XN. {features or columns of dataset}

W1,W2……WN are weights.

Bias: apart from the inputs and corresponding weights we have additional bias. W0 .

2 types of perceptron: single and Multi-layer.

A Single Neuron can do only Binary Classification. Where the decision boundary is linear.

Error is the difference between computed output and the actual output.

We have to reduce the error, to reduce the error the weights are recalculated .

The key difference between linear regression and neureon is neruron will have additional activagion function.

Y = mx+c is linear regression

m = weight

c = bias

dot product is multiplication and addition.

Perceptron equation is :

In the equation given above:

• “w” = vector of real valued weights

• “b” = bias (element that adjusts the boundary away from origin without any dependence on the input value)

• “x” = vector of input x values

• “m” = number of inputs to the Perceptron

Step Funtion: there will be threshold t. (step function takes a decimal as input and rounds it off and outputs a integer)

Sign function:

Sigmoid function:

Taking input and computing output is called forward propagation.

when there is an error, do all weights get adjusted or is each weight adjusted one at a time and adjusted until the error is resolved?

Ans: we compute partial derivate of each weight and update weight. All the weights are updated at once.

The weights decides whether the neruron fires or not

If the output of Activation Function is 1 then it will fire otherwise it wont fire.

In the first run of perceptron how to decide weights?

Usually people start with 0. People start with random values.

Weight are helpful in “Feature Engineering and Extraction”. They onlyl will decide if the feature is important or not.

Logical Gates:

XOR: is a non linear problem

To solve a non-linear problem we need a hidden layers. The purpose of a hidden layer is a ability to train a non-linear classifier.

Sigmoid : Range 0 to +1.

Softmax ---- Multi Class Classification Problem .

TanH: Range -1 to +1 .

Tensorflow is low level api which is used to create and train neural networks models.

Keras is a high level api , wrapper around tensorfow, which is used to create and train neural networks models.

The hyperparametersof the perceptron and Adaline learning algorithms are:

• Learning rate “η” (eta) and

• Number of epochs (n\_iter)

=========================class 3 =================================================

Weights are also called parameters or coefficients. Parameters are in network control and Hyper Parameters are on our control. We decide Hyper Parameters.

Activation Function is also a Hyper Parameters.

Weights are learnt by network. So these are parameters.

Weights are updated by the following formula

W=W-ng

n = eta (Learning rate will be between 0 and 1)

g = gradient

We usually try starting with 0.001 (10 power -3)

Higher Learning Rate leads to

How to decide Number of Neurons in a layer?

How many Layers? To decide?

**Cost Function** and **Error Function** are same. We use the words interchangeably.

**Convergence** is performed so that **cost functions** gets minimized and preferably reaches the **global minima**. It is also done to find the best possible weights to minimize the classification problem.

For a three layer network with n input and m output neurons, the hidden layer would have square root of n∗m neurons.

n = 5 inputs

m = 10 outputs

Hidden layer will have square root of 5\*10 = square root of 50 neurons.

If the network is trained on all training samples once it is called one Epoch.

In Adaline we use a linear Activation Function and in Perceptron we use non linear Activation Function.

1. while adjusting weight are we going to reduce it only?

Ans: Not neccesarily, if the gradient is positive w value will decrease, if gradient is negative w value will increase.

# Binary Classification : sigmoid output with binary cross entropy

# Multi Class Classification : softmax output with categorical cross entropy

Target for multi class is a one hot vector

Error Function (J) we have to minimize. Partial diff of J w.r.t J hence J differentiable.

=============== Day 4 ============================

MLP : Multi layer perceptron

Hyper Parameters: Number of Layers, Number of Neurons, Learning Rate, Number of Epochs, Batch Size, Regularization Lambda value

Parameters: weights( Coefficients ) ,

In MLP weight adjustments are done using back propagation,

Deep Layers can produce Vanishing Gradient Problems.

Learn More about Vanishing Gradient Problems caused by Activation Functions:

<https://www.quora.com/What-is-the-vanishing-gradient-problem>

MLP Notations:

m = number of input features .. this is alwasys number of input featurs + 1 bias

d = number of neurons in Hidden layer. It always number of neurons + 1 bias

t = number of units in output layer

z = dot product of input and weights (Net Input Function)

a = output of Activation Function

y^ = output (Predicted Class Label)

I = bias Unit

X = input

W = weights

MLP is a feedforward example. Weights are updated using back propagation.

activation function notation is φ

Cost Function in a Neural Network:

cost function is called the Cross-entropycost function J(w).

Regularization: L2

To avoid overfitting we use L2 regularization.

We add Regularization function to the cost function to avoid the overfitting proble,

During the training we compute cost function , gradient and update the weights

There are 2 trainign methods. Online and Batch.

Suppose we have 1000 samples in training, and

Mini Batch size should start with 32. Training for 1 Mini Batch is called 1 step.

When we cover all samples in the training then it’s called 1 Epoch.

Dropout is typically done during training to prevent overfitting but not during testing. This is to test the model precisely on the entire test data set.

How to Handle Bias and Variance Tradeoff :?

1. By minimizing the total error:

.

#### Using Bagging and Resampling techniques:

1. **Tuning the hyperparameters that can give an impactful performance**

================ class 5 ========================

Under fitting:

When the model is generalizable means the expected error is similar in unseen data is similar in training data.

Tensor Board : is used to visualize loss etc.,

Tensorflow:

It is built for general large scale machine learning.

Tensors will flow in a diagram.

Different Deep learning libraries:

Facebook: pytorch

Microsoft: Amexnet

Google: Tensorflow

Which API is apt for which scenario?

Tensorflow :

Python API to access tensorflow : tf.learn (tensorflow.cotrib.learn)

High level: tf.layers, tf.keras, tf.slim

Low level:

Tensor: multi dimensional data arrays. Arrays are also referred as Tensors.

How many maximum dimensions tensorflow will support ?

It depends on the RAM.

Pip list will listo out all installed packages.

How Tensor Flow became more popular. Any advantages compared with others?

Placeholders are objects which have values during Training process. To pass a bunch of objects to network mostly we use Hyper parameters .

Placeholders are like Arguments and not parameters.

A **parameter** is a variable **in a** method definition. When a method is called, the **arguments** are the data you pass into the method's **parameters**. **Parameter** is variable **in the** declaration of function. **Argument** is the actual value of this variable that gets passed to function.

A placeholder is simply a variable that we will assign data to at a later date.

Whenever we see the loss function is not converging, (if it is increasing to infinity) then first thing we have to do is change the learning Rate.

By default we use 0.0001 as Learning Rate. Since it is a hyper parameter we have to experiment and change accordingly. 10 power -3 is default. We used 10 power -4 also. We actually trail and error hyper parameters.

In neural networks, We always travel in the direction of global minima.

Tensor flow bullet points:

Read data

Build graph

Train the graph

Predict

Links:

Calculus : <https://arxiv.org/abs/1802.01528>

Back propagation: <https://medium.com/datathings/neural-networks-and-backpropagation-explained-in-a-simple-way-f540a3611f5e>

<https://medium.com/@14prakash/back-propagation-is-very-simple-who-made-it-complicated-97b794c97e5c>

Image classification : CNN

Text Classification : RNN

To compute loss we need a target.

sparse\_softmax\_cross\_entropy\_with\_logits :

* it takes y\_hat (activation of output layer)
* applies softmax
* converts target y to one hot encoding
* and then compute Cross Entropy loss (soft max scores, one hot encoding)

================================== DAY 6 ======================================

All the computations in the neural network we usually define in graph. Graph is something like a class.

We define place holders, variables, loss function and optimizer of choice to optimize the loss function, and then

We initialize the hyper parameters, and then we declare parameters how we should sample them and we manually made all the connectios, the dot product and all.

We define the trainer object which minimizes the loss function.

High level API:

We use tf.layers API to define layers. In high level API we need not to use W1 and B1. When we use softmax cross entropy as loss function we pass target labels, and y-hat as parameters.

When do we use low-level API, High Level API and when do we use Keras?

We generally go for high level API. If we use MLP then there is more reason to use High level API.

When you want to develop a new algorithm then a low level api is good start.

For standard tasks then High Level API and for research purpose we use Low Level API.

when to use keras and when to use tf.layers?

What is the optimal Batch\_Size? 32,64,128 works fine. Any thing greater than 128 and less than 32 will have no optimized solutions.

When it comes to Hyper Parameters:

No of Layers

No of Neurons

Learning Rate

Regularization

Epochs matters a lot in Hyper Parameters.

Optimizers:

Tf.train.GradinetDescentOptimizer

tf.train.AdamOptimizer

Saving and Restoring data models:

Whjat if we have new data:

Scenario 1: combine old data with new data and train the new model

Scenario 2: Take previously trained model, train on the new data only

Scenario 3: Take previously trained model, train on combined old and new data with small learning rate.

How to figure out the best Model?