Deep Learning Package: Tensorflow

Theory and Interfacing

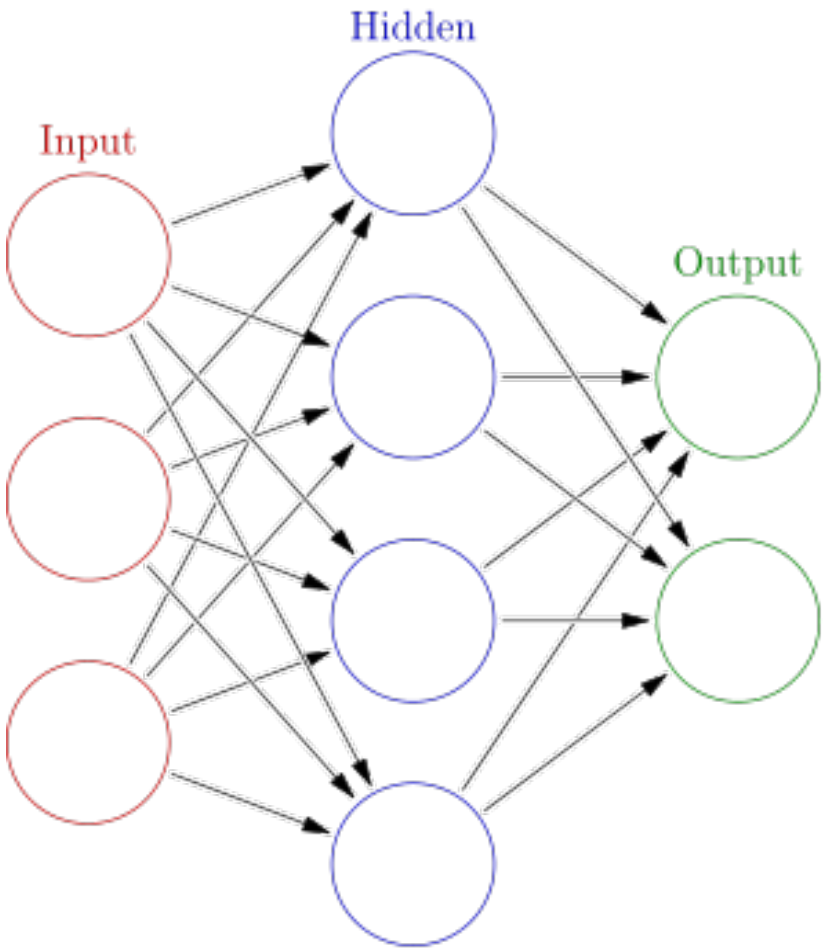
Vishal Kumar Jaiswal

IIEST Shibpur

June 18, 2016

Table of Contents

Artificial Neural Network



Pictorial representation of neural network

Artificial Neural Network (aka ANNs)

- Contains sets of adaptive weights, i.e. numerical parameters that are tuned by a learning algorithm,
- Weight is connection strength between neuron, which are activated during training and prediction.
- e.g. For handwriting recognition,a set of input neurons may be activated by the pixels of an input image. After being weighted and transformed by a function, it is passed on to other neurons. This process is repeated until finally, the output neuron that determines which character was read is activated.
- Other tasks include computer vision, speech recognition etc.

Terminology

Score Function

Maps the raw image pixels to class scores(e.g. linear function Wx_i).

Loss Function

Measures the quality of a particular set of parameters based on how well the induced scores agreed with the ground truth labels in the training data.

Optimization

The process of finding the set of parameters W and b that minimize the loss function.

Logistic Classifier

- A linear classfier.
- Based on $WX + b = Y$ where X is input vector (e.g. image pixel), W is weight matrix, b is the bias term, Y is score for a particular class.
- Tuning weight and bias is where the machine learning comes in.
- Score Y will be changed into probabilities.
- An input can belong to at most one class.

SOFTMAX

Softmax function

Takes any kind of score and turns them into proper probabilities.

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

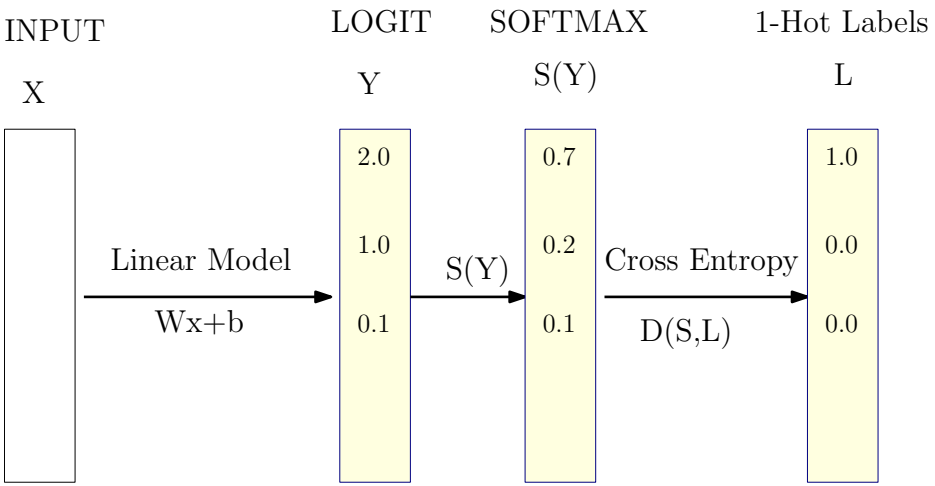
Note:

- Proper probabilities will sum to one.
- Probabilities will be large when scores are large and will be small when the scores are comparatively smaller.
- Scores are also known as logits.

Softmax: Code

```
scores=[3.0,1.0,0.2]
import numpy as np
def softmax(x):
        return np.exp(x)/np.sum(np.exp(x),axis=0)
print(softmax(np.array(scores)))
import matplotlib.pyplot as plt
x=np.arange(-2.0,6.0,0.1)
scores=np.vstack([x,np.ones_like(x),0.2*np.ones_like(x)])
plt.plot(x,softmax(np.array(scores)/10).T,linewidth=2)
plt.show()
```

Multinominal Logistic Classification



Workflow of deep learning classification, $D(S(WX + b), L)$

One hot encoding

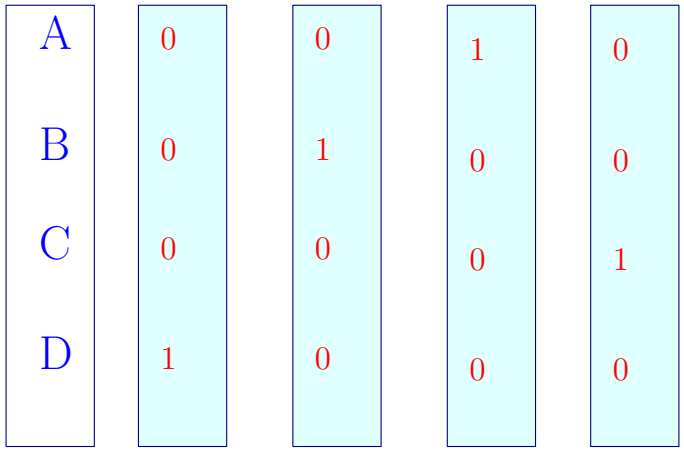
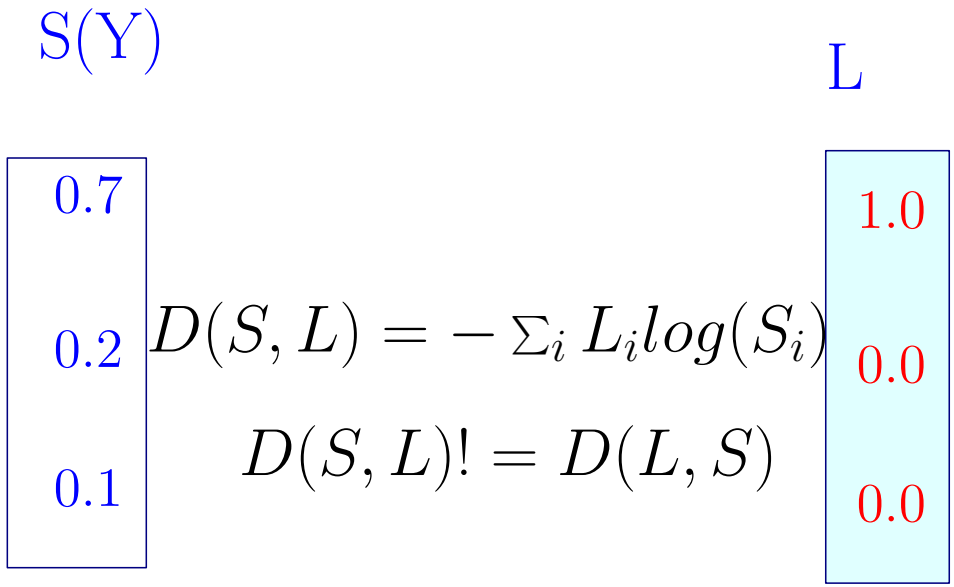


Illustration of one hot encoding

Cross entropy

- Useful when you have millions of classes,
- Vector bacomes large and sparse in one hot encoding matrix,
- Cross Entropy is not symmetric.



Cross Entropy

Numerical Stability

It's about adding very large value to very small value.

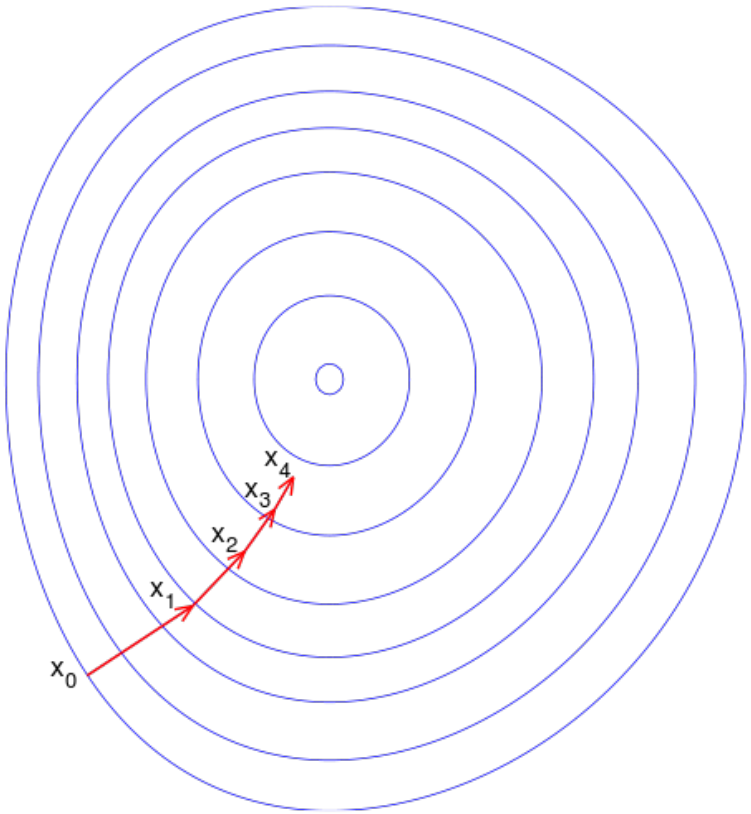
```
#a=1000000000
a=1
for i in xrange(1000000):
        a=a+1e-6
print a-1
#print a-1000000000
```

Numerical Stability

- Variables should have zero mean and equal variance $\sigma(X_i) = \sigma(X_j)$.
- A lot of searching required for optimized solution when above case is not satisfied.
- Pixel values X_i range from 0 to 255 in image.
  To fix problem, use

$$R = \frac{R - 128}{128} \qquad G = \frac{G - 128}{128} \qquad B = \frac{B - 128}{128}$$

Gradient Descent



Optimization using Gradient Descent

Gradient Descent

- Most popular algorithm for optimizing objective function in Neural network.
- The learning rate η determines the size of steps we take to reach a (local) minimum.
- In hiking analogy, feeling the slope of the hill below our feet and stepping down the direction that feels steepest.
- The slope is the instantaneous rate of change of the function at any point.

Gradient

Gradient

The gradient is a generalization of slope for functions that takes a number of vector of numbers.So basically, it is just a vector of slopes for each dimension in the input space.

$$\frac{df(x)}{dx} = lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Calculating Gradient: Code

```
# Numerical gradient
def eval_numerical_gradient(f,x):
        fx=f(x)
        grad=np.zeros(x.shape)
        h=0.00001
        it=np.nditer(x,flags=['multi_index'],op_flags=['readwrite']
        while not it.finished:
                ix=it.multi_index
                old_value=x[ix]
                x[ix]=old_value+h
                fxh=f(x)
                x[ix]=old_value
                grad[ix]=(fxh-fx)/h
                it.iternext()
        return grad
```

Initialization of logistic classifier

Avg. cross entropy loss

$$L = \frac{1}{N} \sum_i D\left(S\left(WX_i + b\right), L_i\right)$$

Where, S - Softmax function,

D - Cross entropy loss,

W - Weight(initially random),

b - Bias.

Backpropagation optimization

Optimization

$W = W - \alpha \Delta_W L$

$b = b - \alpha \Delta_b L$

Repeat until minimum of loss.

Stochastic Gradient Descent (SGD)

Why use SGD ?

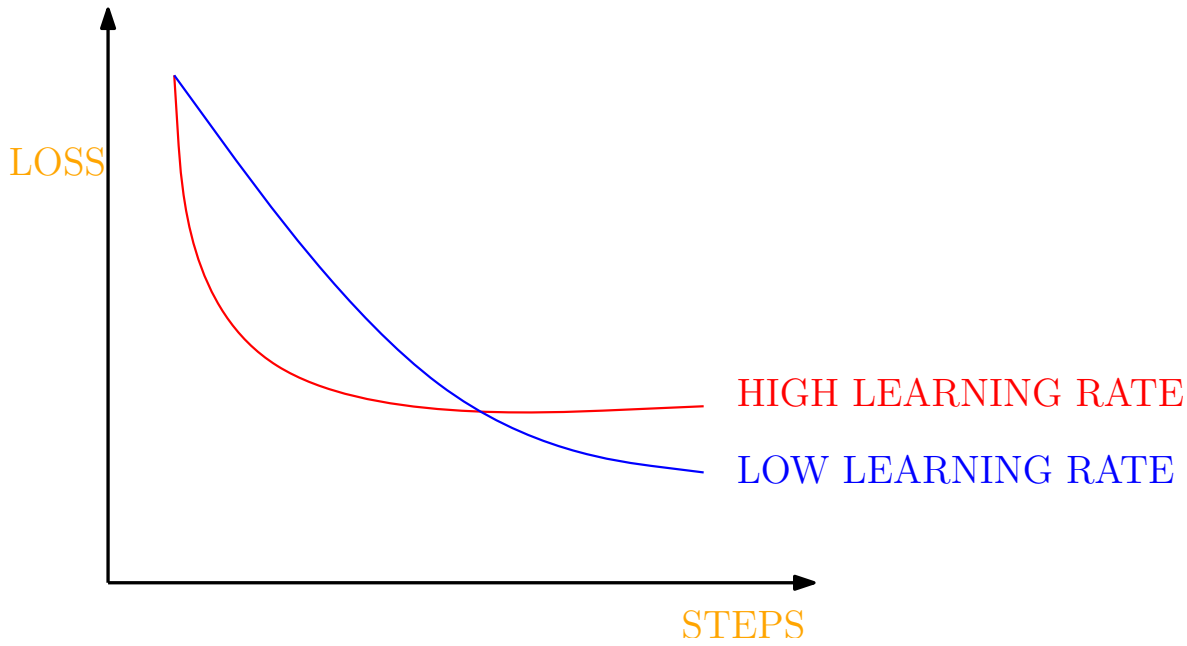
To calculate loss for every element in big data is very inefficient process.So,just pick random sample of datapoints and repeat it several times.(maybe thousand of datapoints in dataset of millions datapoints).

Hyper parameters

Configure following hyper parameters:

- Momentum: running average of gradient $0.9M + \Delta L$. Using running average in backpropagation formulae instead of loss factor, leads to better convergence.
- Learning Curve Decay: learning step size is an area of research. lower it over time is a key thing to remember.

Learning Rate Tuning



Learning rate convergence

Data Flow Graphs

- Describes computation with a digraph.
- Nodes represent the mathematical operations or endpoints to feed in data, push out results, or read/write persistent variables.
- Edges describe the input/output relationships between nodes.
- These edges carry dynamically-sized multidimensional data arrays or tensors.
- Flow of tensors through graph assignes Tensorflow its name.

Tensorflow

- Open source library for machine intelligence developed by Google Brain team,
- Performs numerical computation using data flow graphs,
- Play with tensorflow: http://playground.tensorflow.org/

DSSTNE: Deep Scalable Sparse Tensor Network Engine

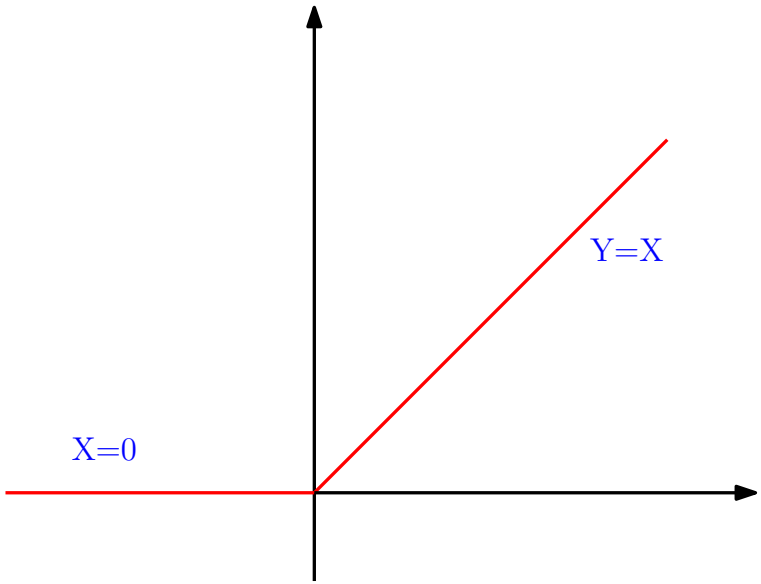
- Open source software library developed by Amazon,
- Used for training and deploying deep neural networks using GPUs,
- Built for real-world deep learning applications,
- Can perform model-parallel training on Multiple GPUs,
- Suitable for sparse datasets,
- Optimized for fast performance in production applications,
- Automatically distributes each computation across all available GPUs,
- 2.1x speedup relative to Tensorflow,
- Can work for millions of products or natural language processing with very large vocabulary.

Recurrent Neural Network

- Sharing of weights over time,
- Creates an internal state of the network,
- Applicable to tasks such as unsegmented connected handwriting recognition or speech recognition.

Rectified linear Units

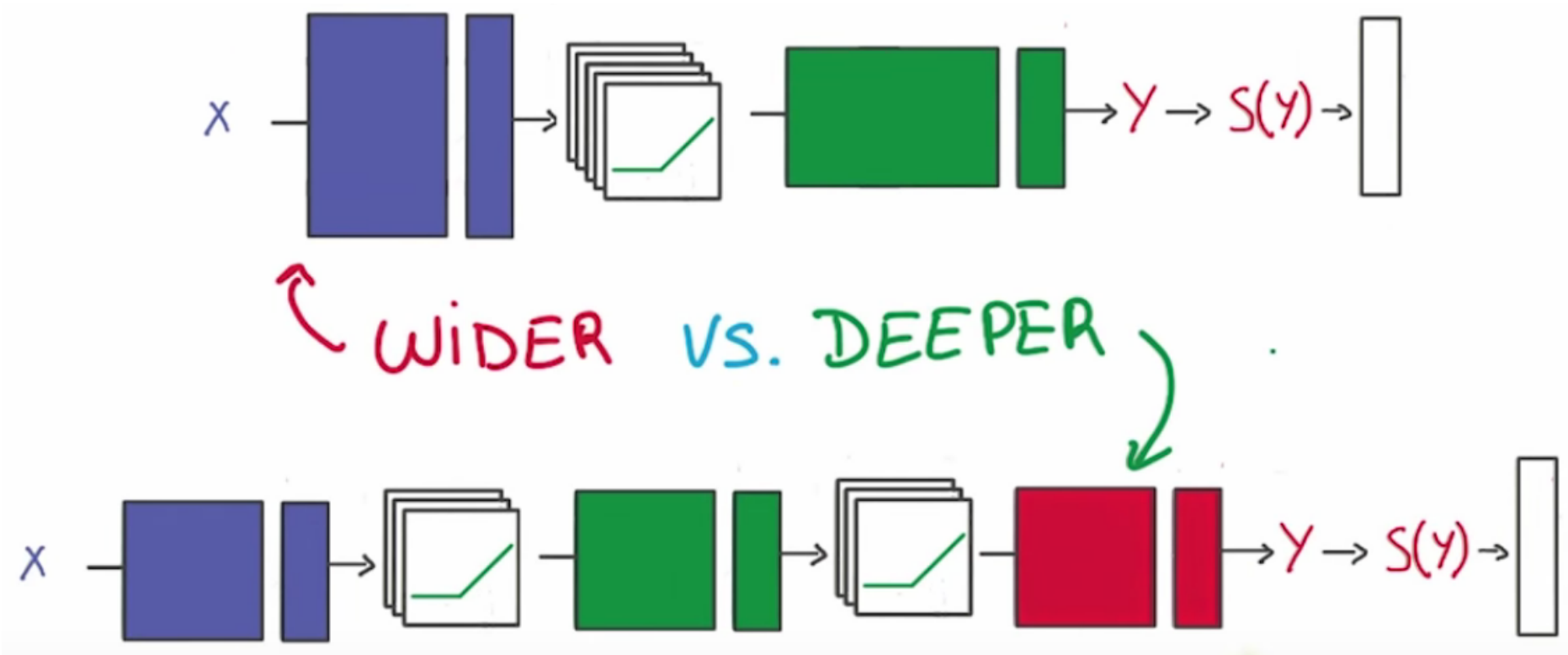
Simplest non-linear functions.



Rectified linear Units

What is Deep learning Network?

Making a logistic classifier non-linear



Deep learning Network

References

Vincent Vanhoucke Arpan Chakraborty.

Ud730: Deep learning.
Udacity.

Andrej Karpathy Justin Johnson.

Cs231n: Convolutional neural networks for visual recognition.
Stanford CS.

Sebastian Ruder.

An overview of gradient descent optimization algorithms.
AYLIEN.

Thank You