Language Translation with Deep Learning

Using TensorFlow on FloydHub

About me

Machine Learning and Big Data Architect

MS Computer Science, Machine Learning, Georgia Tech
Deep Learning Nanodegree, Udacity
Artificial Intelligence Nanodegree, Udacity

Connect

Twitter : @techmazed
LinkedIn : https://www.linkedin.com/in/knownasar

About Nashville AI Society Meetup

- This is a group for anyone interested in AI, Machine Learning, Deep Learning, NLP, Deep RL, Image Recognition, Speech Recognition etc.
- All backgrounds are welcome!
- This group is created in order to bring AI enthusiasts together, build the leading AI community and move forward the research and technology advancement in the AI related subfields.

Join the conversations at

Meetup Group: https://www.meetup.com/Nashville-Artificial-Intelligence-Society/

LinkedIn Group: https://www.linkedin.com/groups/12043828

Twitter: @AINashville (https://twitter.com/AInashville)

p.s. We are currently looking for venue Sponsors!

What we'll cover

- The Language Translation challenge
- Introducing Deep Learning
- Introducing RNN
- Introducing NLP concepts
- Brief introduction to TensorFlow & FloydHub

- The solution approach
- Demo time and code walkthrough
- What next!
- Question time

The Language Translation challenge

Language translation challenge

We will be training a **sequence to sequence** model on a dataset of English and French sentences that can translate new sentences from English to French

We use a small portion of the English & French corpus

English:

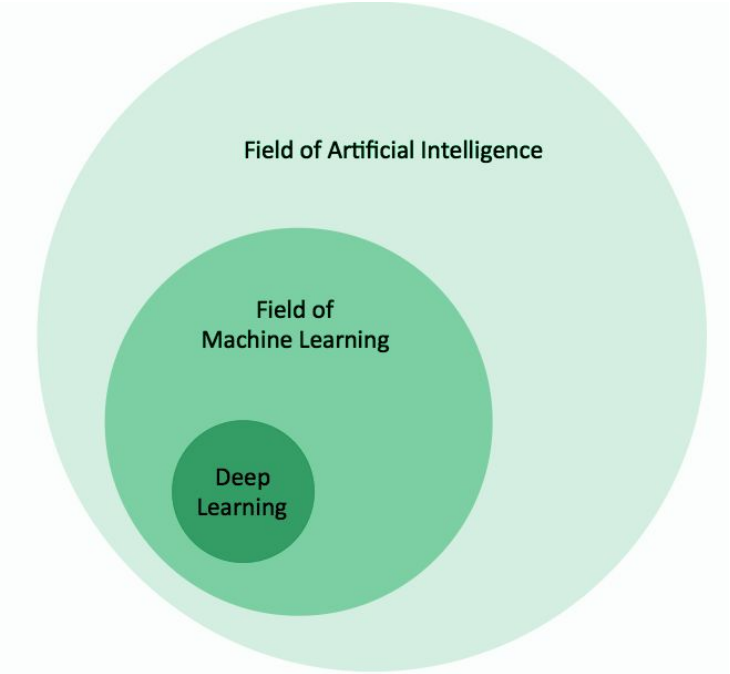
new jersey is sometimes quiet during autumn , and it is snowy in april .

French:

new jersey est parfois calme pendant l' automne , et il est neigeux en avril .

Introducing Deep Learning

Field of Deep Learning



Buckets of DL

General DL

Sequence Models

RNN/LSTM/GRU

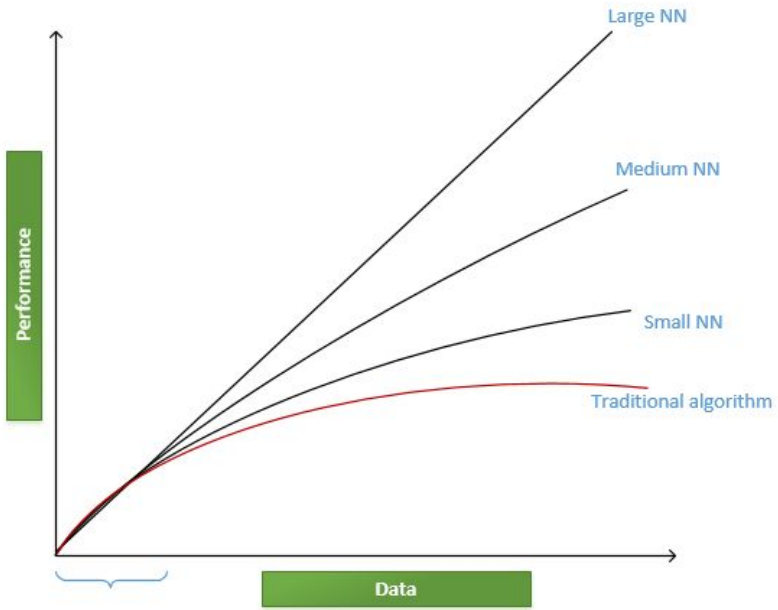
Image

2D / 3D
CNN

Other

Unsupervised
Reinforcement
Learning

Deep Learning vs Traditional Algorithms



Data vs Performance

Source: https://www.youtube.com/watch?v=F1ka6a13S9I

What is Deep Learning

Neural networks with large number of parameters and layers.

The layers belong to one of four fundamental network architectures:

- Unsupervised Pre-Trained Networks
- Convolutional Neural Networks
- Recurrent Neural Networks
- Recursive Neural Networks

A neural Network is an algorithm that learns to identify patterns in data

Backpropagation is a technique to train a Neural Net by updating weights via gradient descent

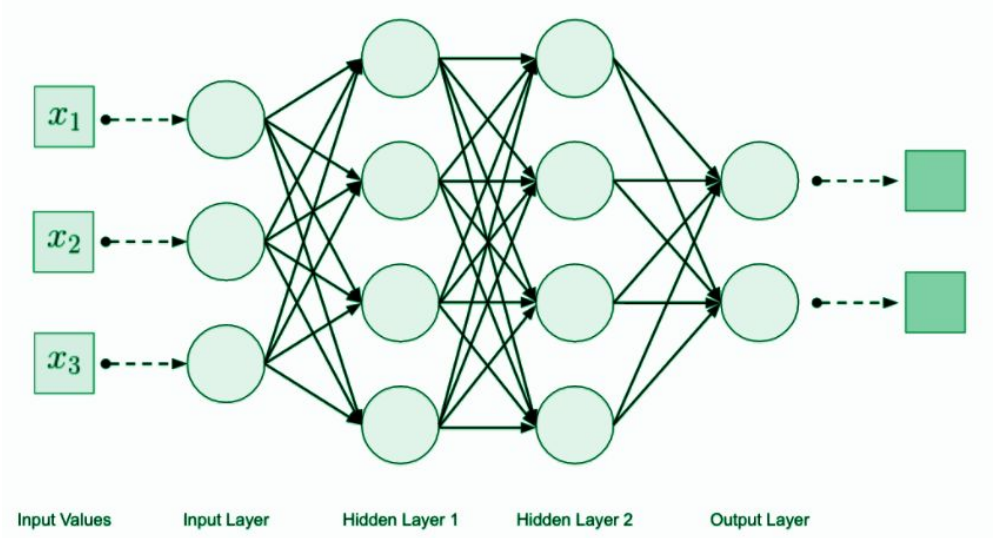
Deep Learning :

many layer neural net + massive data (big data) + massive compute (GPU)

Multi-Layer Neural Network topology

The behavior of neural networks is shaped by its network architecture. A network's architecture can be defined, in part, by:

- number of neurons
- number of layers
- types of connections between layers



Backpropagation Learning

Backpropagation is an important part of reducing error in a neural network model.

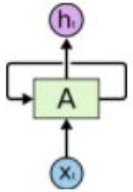
General Neural Network Training Pseudocode

```
function neural-network-learning( training-records ) returns network
    network <- initialize weights (randomly)
    start loop
        for each example in training-records do
            network-output = neural-network-output( network, example )
            actual-output = observed outcome associated with example
            update weights in network based on { example, network-output, actual-output }
        end for
    end loop when all examples correctly predicted or hit stopping conditions
    return network
```

Introducing Recurrent Neural Network

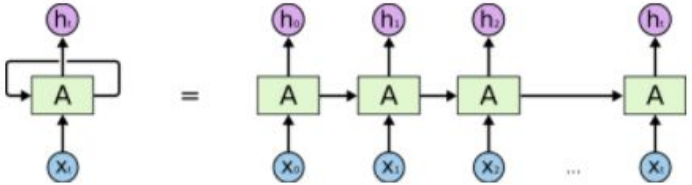
What is RNN?

They are networks with loops in them, allowing information to persist. Recurrent Neural Networks have loops.

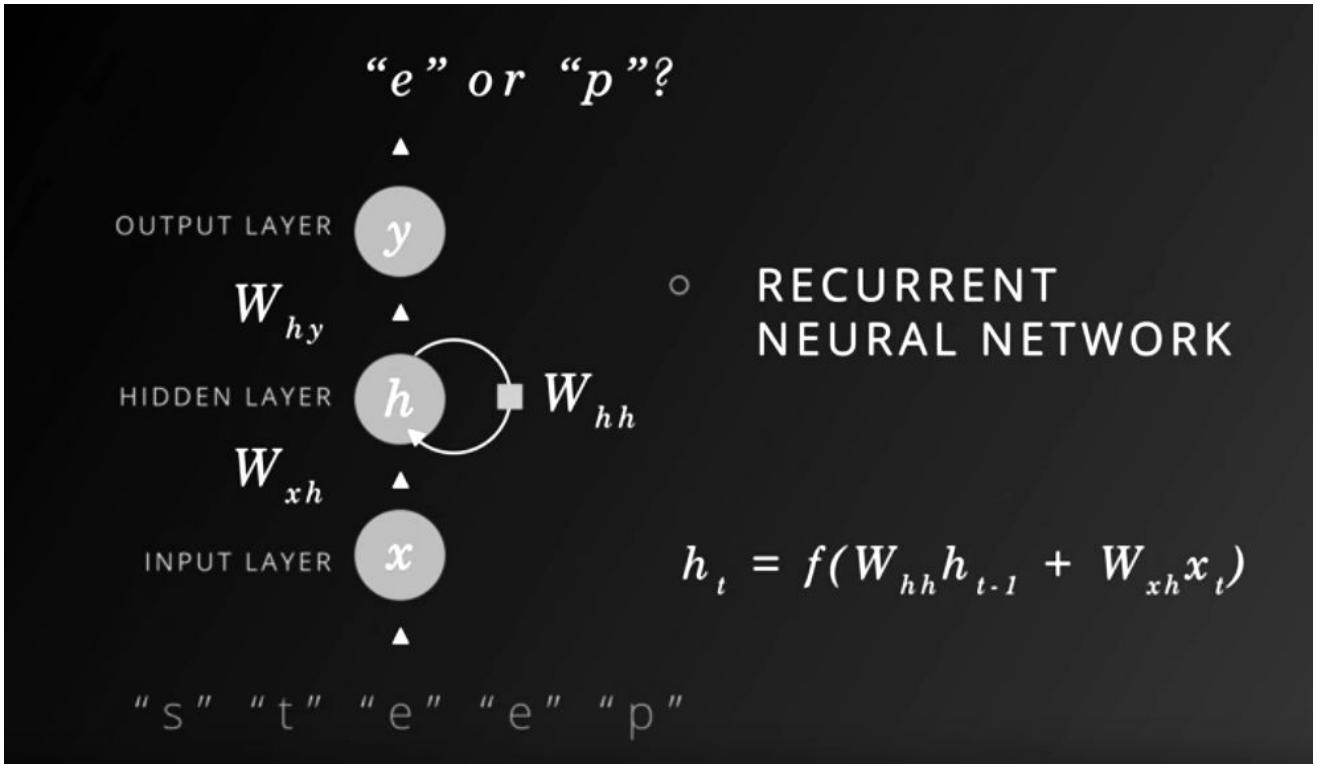


In the above diagram, a chunk of neural network, A, looks at some input x_t and outputs a value h_t.
A loop allows information to be passed from one step of the network to the next.

A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor. Consider what happens if we unroll the loop:



Recurrent Neural Network - single character example



W = weight

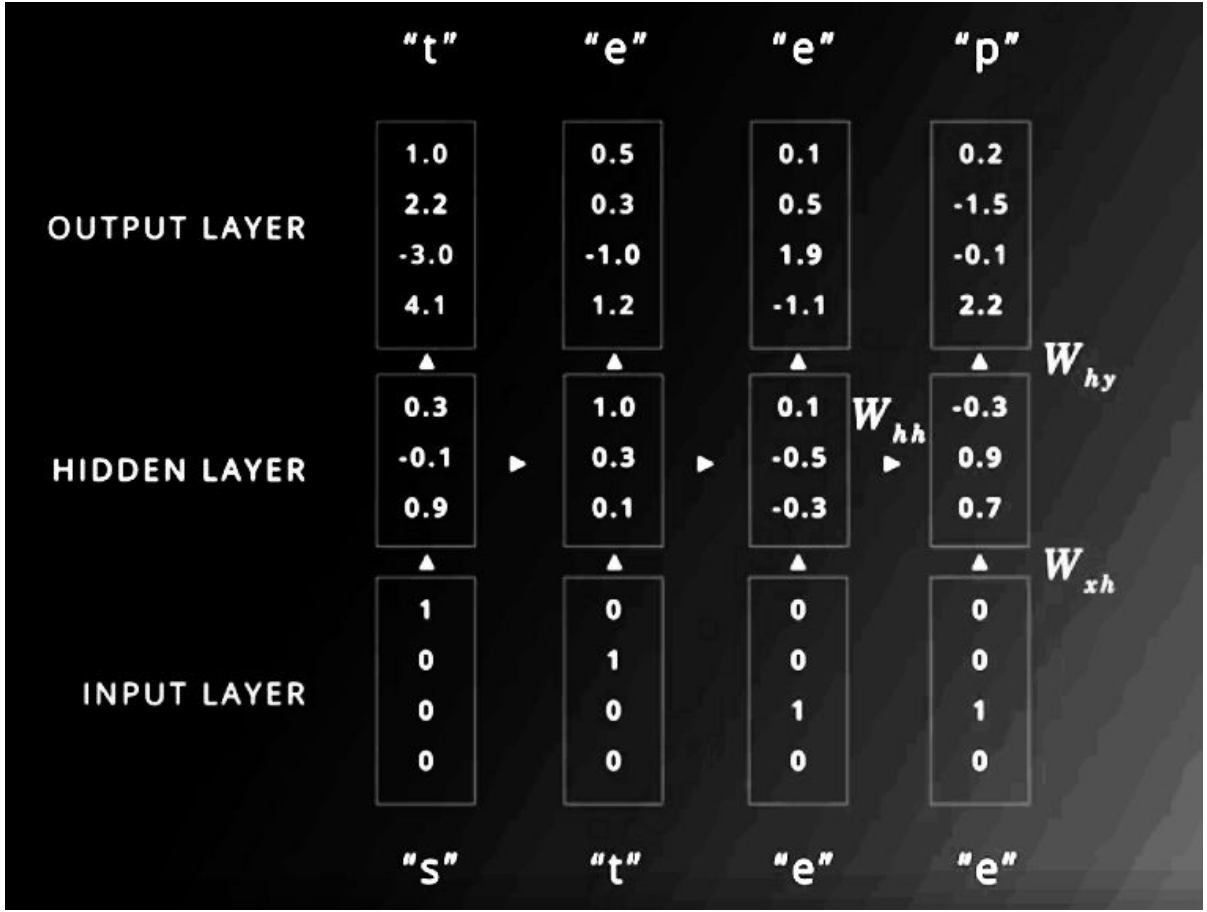
H = hidden layer

x = input

y = output

t = time step

Recurrent Neural Network - the flow



Introducing Natural Language Processing (NLP) concepts

One-Hot Encoding

One hot encoding technique is used to encode categorical integer features using a one-hot (also called one-of-K) scheme.

Example: Suppose you have 'color' feature which can take values 'white', 'red', and 'black' etc.

Original data:

id	Color
1	White
2	Red
3	Black
4	Purple
5	Gold

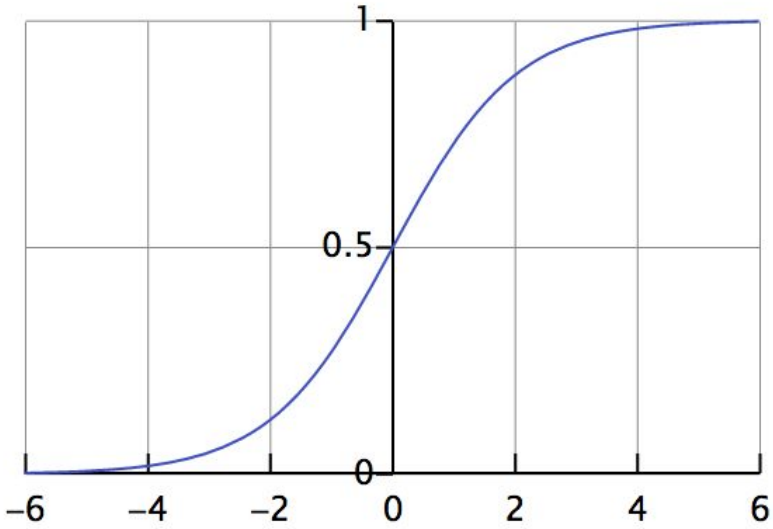
One-hot encoding format:

id	White	Red	Black	Purple	Gold
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	0	0	0	1	0
5	0	0	0	0	1

Logits

Logistic Function:

$$f(x) = \frac{1}{1 + e^{-\theta x}}$$

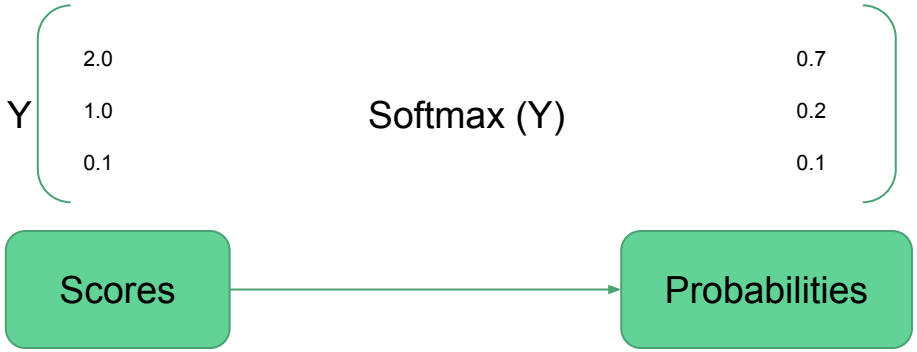


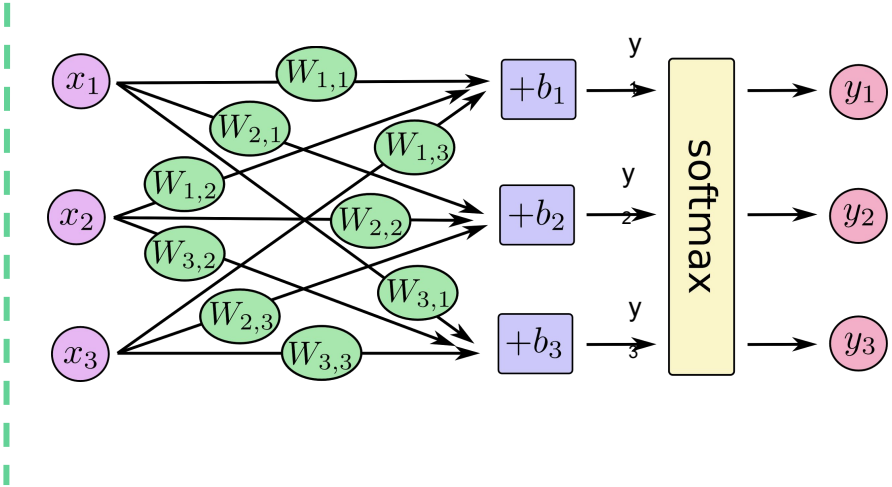
The logit function is the inverse of the logistic function.

SoftMax

Multiclass Classification Model Output Layer.

The softmax activation function returns the probability distribution over mutually exclusive output classes. Softmax is the function you will often find at the output layer of a classifier.



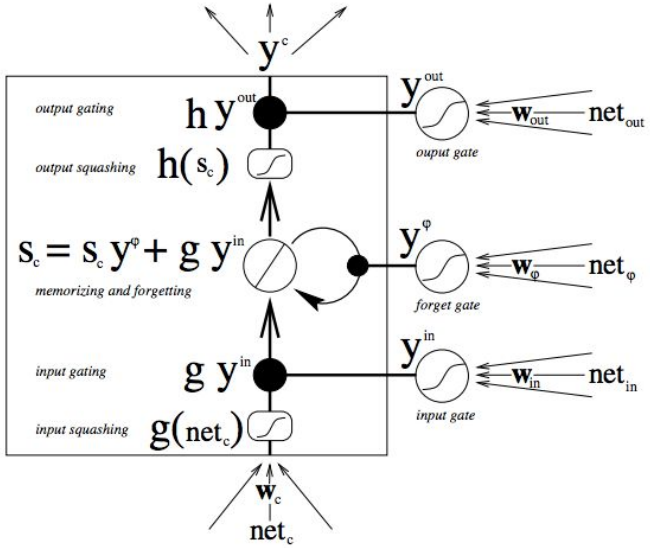


LSTMs

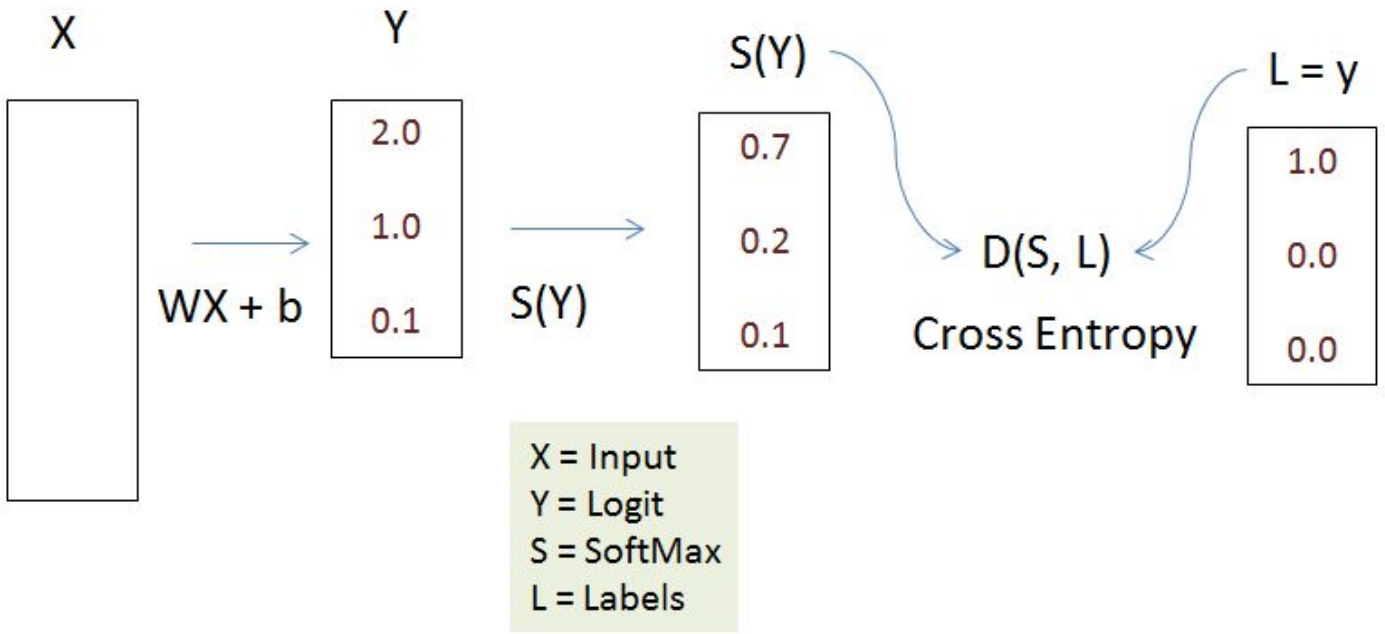
Stands for Long Short-Term Memory Units (LSTMs).

This basically handles the problem of vanishing and exploding gradients. LSTMs help preserve the error that can be backpropagated through time and layers.

Diagram (right): Illustrates how data flows through a memory cell and is controlled by its gates:



Cross Entropy & Embeddings



Brief introduction to technologies used

TensorFlow

What is TensorFlow?

TensorFlow is an open source software library released in 2015 by Google to make it easier for developers to design, build, and train deep learning models.

TensorFlow originated as an internal library that Google developers used to build models in house, and we expect additional functionality to be added to the open source version as they are tested and vetted in the internal flavor.

Why use: Thoughtful design & Ease of use

Some TensorFlow alternatives are: Theano, Torch, Caffe, Neon, and Keras

Variables in TensorFlow

TensorFlow variables are in-memory buffers that contain tensors, but unlike normal tensors that are only instantiated when a graph is run and are immediately wiped clean afterwards, variables survive across multiple executions of a graph. TensorFlow variables have the following three properties:

- Variables must be **explicitly initialized** before a graph is used for the first time
- We can use gradient methods to **modify variables after each iteration** as we search for a model's optimal parameter settings
- We can **save the values stored in variables** to disk and restore them for later use.

```
weights = tf.Variable(tf.random_normal([300, 200], stddev=0.5), name="weights", trainable=False)

#other methods to initialize a TensorFlow variable:
tf.zeros(shape, dtype=tf.float32, name=None)
tf.ones(shape, dtype=tf.float32, name=None)
tf.random_normal(shape, mean=0.0, stddev=1.0, dtype=tf.float32, seed=None, name=None)
tf.truncated_normal(shape, mean=0.0, stddev=1.0, dtype=tf.float32, seed=None, name=None)
tf.random_uniform(shape, minval=0, maxval=None, dtype=tf.float32, seed=None, name=None)
```

Operations in TensorFlow

- TensorFlow operations represent abstract transformations that are applied to tensors in the computation graph.
- Operations may have attributes that may be supplied a priori or are inferred at runtime.

Category	Examples
Element-wise mathematical operations	Add, Sub, Mul, Div, Exp, Log, Greater, Less, Equal, ...
Array operations	Concat, Slice, Split, Constant, Rank, Shape, Shuffle, ...
Matrix operations	MatMul, MatrixInverse, MatrixDeterminant, ...
Stateful operations	Variable, Assign, AssignAdd, ...
Neural network building blocks	SoftMax, Sigmoid, ReLU, Convolution2D, MaxPool, ...
Checkpointing operations	Save, Restore
Queue and synchronization operations	Enqueue, Dequeue, MutexAcquire, MutexRelease, ...
Control flow operations	Merge, Switch, Enter, Leave, NextIteration

Placeholder Tensors in TensorFlow

- A placeholder can be initialized every time, unlike variables
- This can be populated every single time the computation graph is run

```
x = tf.placeholder(tf.float32, name="x", shape=[None, 784])
W = tf.Variable(tf.random_uniform([784,10], -1, 1), name="W")
multiply = tf.matmul(x, W)
```

Sessions in TensorFlow

- A TensorFlow program interacts with a computation graph using a session.
- The TensorFlow session is responsible for building the initial graph, can be used to initialize all variables appropriately, and to run the computational graph.

```
import tensorflow as tf
from read_data import get_minibatch()

x = tf.placeholder(tf.float32, name="x", shape=[None, 784])
W = tf.Variable(tf.random_uniform([784, 10], -1, 1), name="W")
b = tf.Variable(tf.zeros([10]), name="biases")
output = tf.matmul(x, W) + b

init_op = tf.initialize_all_variables()

sess = tf.Session()
sess.run(init_op)
feed_dict = {"x" : get_minibatch()}
sess.run(output, feed_dict=feed_dict)
```

FloydHub

What is FloydHub?

“Floydhub is Heroku for Deep Learning. A Platform-as-a-Service for training and deploying your models in the cloud with a few simple commands.”

```
# Install locally and login
$pip install -U floyd-cli
$floyd login

# Run a Jupyter notebook on FloydHub
$ floyd run --mode jupyter --gpu

#Run a python code on GPU
$ floyd run --gpu "python mnist_cnn.py"
```

http://docs.floydhub.com/home/getting_started/

The solution: Language translation approach

Language translation approach

Preprocess

Convert text into a number (ID)

Build the Network

Let us build the components necessary to build a Sequence-to-Sequence model by implementing the following functions below:

- Model inputs
- Process decoding on the inputs
- Encoding layer
- Decoding layer
- Sequence to Sequence model

Continued...

Hyperparameters

Tuning the following hyperparameters:

- Number of epochs
- Batch size
- Size of the RNNs
- Number of layers
- Size of the embedding for the encoder
- Size of the embedding for the decoder
- Learning rate
- Dropout keep probability

DEMO TIME

What next !

References

Books:

1. TensorFlow for Deep Learning (Nikhil Buduma, 8/2016), OReilly
2. Grokking Deep Learning (Andrew Trask, early 2018), Manning
3. Python: Deeper Insights into Machine Learning (Sebastian Raschka, 8/2016), Packt

Other:

1. Foundations of Deep Learning (Hugo Larochelle, Twitter) - https://youtu.be/zij_FTbJHsk
2. Deep Learning for Computer Vision (Andrej Karpathy, OpenAI) - https://youtu.be/u6aEYuemt0M
3. Deep Learning for Natural Language Processing (Richard Socher, Salesforce) - https://youtu.be/oGk1v1jQITw
4. TensorFlow Tutorial (Sherry Moore, Google Brain) - https://youtu.be/Ejec3ID_h0w
5. Foundations of Unsupervised Deep Learning (Ruslan Salakhutdinov, CMU) - https://youtu.be/rK6bchqeaN8
6. Nuts and Bolts of Applying Deep Learning (Andrew Ng) - https://youtu.be/F1ka6a13S9I
7. Deep Reinforcement Learning (John Schulman, OpenAI) - https://youtu.be/PtAIh9KSnjo
8. Theano Tutorial (Pascal Lamblin, MILA) - https://youtu.be/OU8I1oJ9HhI
9. Deep Learning for Speech Recognition (Adam Coates, Baidu) - https://youtu.be/g-sndkf7mCs
10. Torch Tutorial (Alex Wiltschko, Twitter) - https://youtu.be/L1sHcj3qDNc
11. Sequence to Sequence Deep Learning (Quoc Le, Google) - https://youtu.be/G5RY_SUJih4
12. Foundations and Challenges of Deep Learning (Yoshua Bengio) - https://youtu.be/11rsu_WwZTc

Questions