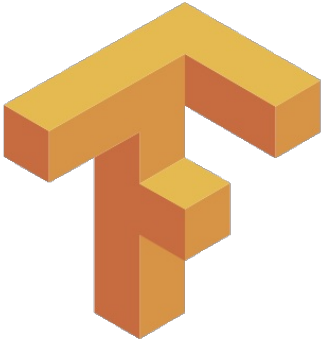


TensorFlow

Jake S. Choi (shchoi@diotek.com)
2015.12.17

차례

TensorFlow?

- Open source software library for **numerical computation using data flow graphs**
  - Deep learning framework이 아니네?
- Google Research Blog에 2015년 11월 9일 발표
- Apache 2.0 라이선스

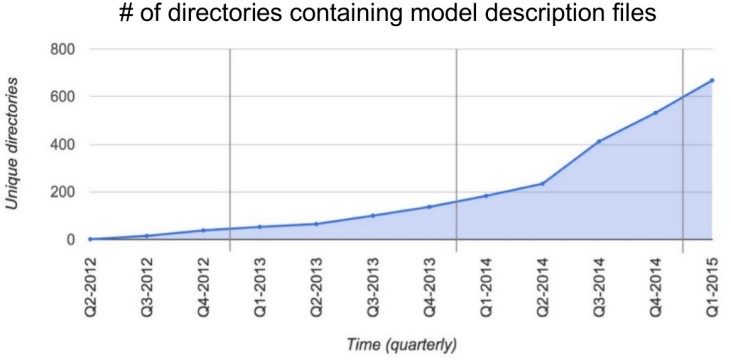
Tensor - Flow?

- **Tensor**: N-dimensional array
  - 1-dimension: Vector
  - 2-dimension: Matrix
  - Represent many dimensional data flowing through the graph
    - e.g. Image represented as 3-d tensor rows, cols, color
- **Flow**: Computation based on data flow graphs
  - Lots of operations (nodes in the graph) applied to data flowing through
- Tensors flow through the graph → "***TensorFlow***"
  - Edges represent the tensors (data)
  - Nodes represent the processing

[1] Large-Scale Deep Learning for Intelligent Computer Systems

배경

- 2012년 기준으로 구글 내부적으로 machine learning, deep learning에 대한 수요가 증가함

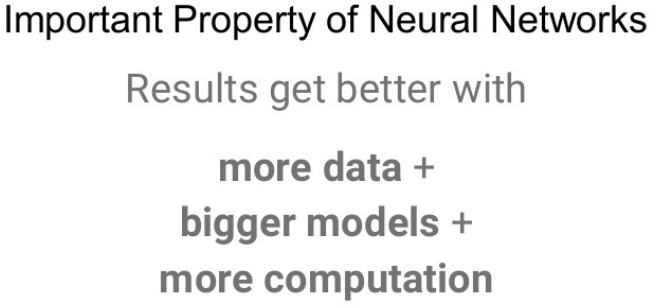


[1] Large-Scale Deep Learning for Intelligent Computer Systems

  - GMail, Photos, Speech 등 어러 분야에 적용

배경

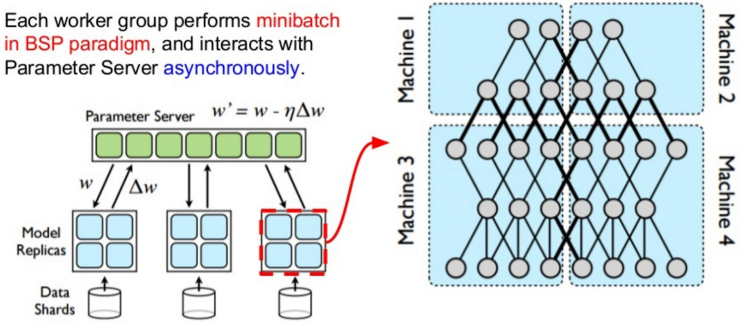
- 2011년 Google Brain project의 일환으로 대용량 데이터와 대규모 계산을 중점으로 개발을 시작



[1] Large-Scale Deep Learning for Intelligent Computer Systems

DistBelief

- 2011년에 개발된 Google의 Deep learning infrastructure
- 기본 설계 방향
  - 처리할 데이터가 많다 → 데이터 병렬화
  - 모델이 너무 커서 메모리에 한번에 안 올라감 → 모델 병렬화
- 구글은 학습에 천 단위 코어 수를 사용함

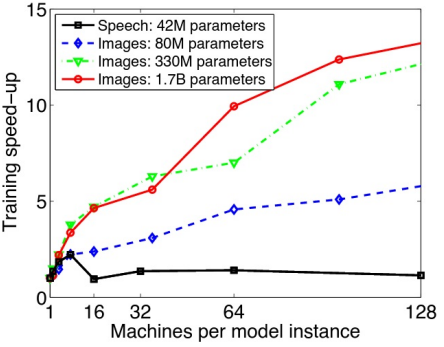


[2] Introduction to apache horn (incubating)

DistBelief

Model parallelism benchmarks

- 실험 결과
  - HWR은 일반적으로 40K, 최대 10M까지 테스트
  - Speech의 경우 8대에서 2.2배의 성능 향상, 8대 이상은 데이터 교환 비용으로 성능 하락이 확인됨

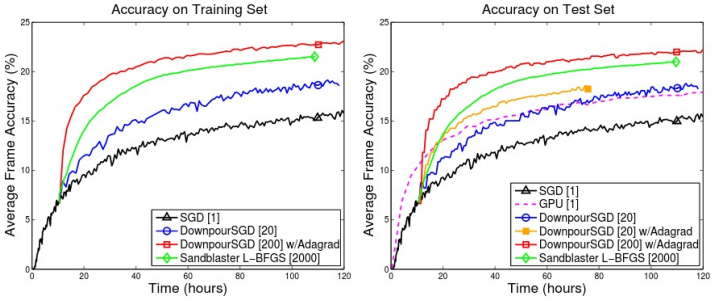


[4] Large Scale Distributed Deep Networks

DistBelief

Data parallelism benchmarks

- 실험 결과



[4] Large Scale Distributed Deep Networks

TensorFlow

- 기존 DistBelief를 개선
- DistBelief은 규모에 대한 확장성은 뛰어나지만 유연성이 떨어졌음
- TensorFlow는 DistBelief 보다는 2배 정도 빠름다고 함
  - 아직 white paper에 benchmarking 자료는 미포함 됨 [3]
- 구글 내부적으로는 DistBelief에서 TensorFlow로 이전을 완료한 상태

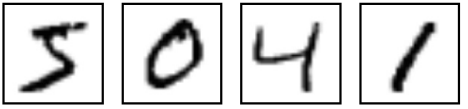
Tutorial - Logistic regression

0 ~ 9 숫자 하나를 인식하는 문제, 문제의 해결을 위한 모델은 logistic regression을 사용

MNIST 파일을 로딩, `input_data`는 구글에서 미리 구현해 둔 코드

```
# Import MINST data
import input_data
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
```

데이터는 아래와 같은 형태의 이미지

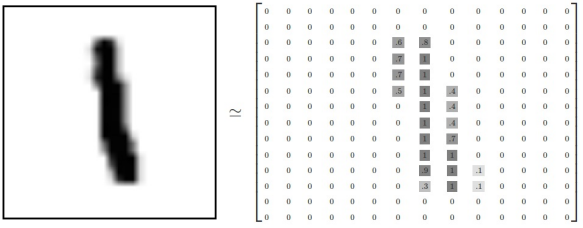


[6] MNIST For ML Beginners

Tutorial - Logistic regression

mnist.train, mnist.validation, mnist.test로 구성

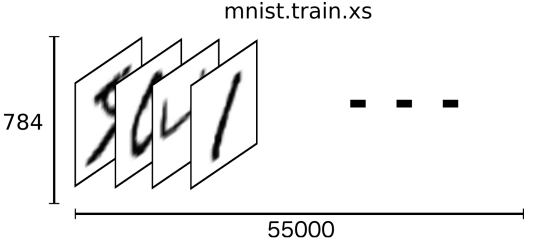
`mnist.*.xs`는 아래와 같은 형태의 28 x 28 x 1(Gray scale) 이미지



[6] MNIST For ML Beginners

Tutorial - Logistic regression

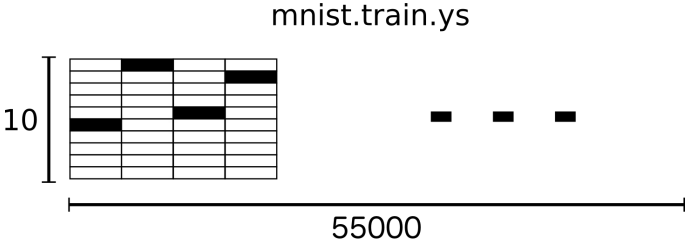
총 60,000의 이미지로 구성



[6] MNIST For ML Beginners

Tutorial - Logistic regression

`mnist.*.ys`는 0 ~ 9까지 하나만 1을 가지는 one-hot vector로 목표/정답 데이터



[6] MNIST For ML Beginners

Tutorial - Logistic regression

```
import tensorflow as tf
# Parameters
learning_rate = 0.01
training_epochs = 25
batch_size = 100
display_step = 1
# tf Graph Input
x = tf.placeholder("float", [None, 784]) # mnist data image of shape 28*28=784
y = tf.placeholder("float", [None, 10]) # 0-9 digits recognition => 10 classes
# Create model
# Set model weights
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
# Construct model
activation = tf.nn.softmax(tf.matmul(x, W) + b) # Softmax
# Minimize error using cross entropy
cost = -tf.reduce_sum(y*tf.log(activation)) # Cross entropy
# Gradient Descent
optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)
```

Tutorial - Logistic regression

None x 784 (28x28)으로 x라는 matrix를 정의, placeholder로 정의된 내용은 그래프 계산 전 전달 되어야 함, 이 값은 외부에서 그래프로 입력을 다룰 때 사용 y는 정답 데이터가 입력됨, 여기서 None으로 정의된 값은 mini-batch size가 됨

```
x = tf.placeholder("float", [None, 784])
y = tf.placeholder("float", [None, 10]) # 0-9 digits recognition => 10 classes
```

W는 input x output 크기의 matrix, b는 bias로 output 크기와 동일한 변수를 정의

```
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
```

Tutorial - Logistic regression

logistic regression 모델을 정의

$$\text{evidence}_i = \sum_j W_{i,j} x_j + b_i$$

$$y = \text{softmax}(\text{evidence})$$

tf.matmul은 matrix multiply 함수

```
activation = tf.nn.softmax(tf.matmul(x, W) + b) # Softmax
```

Tutorial - Logistic regression

cross entropy 함수

$$H_{y'}(y) = -\sum_i y'_i \log(y_i)$$

cost 함수를 위의 수식과 같이 cross entropy 함수와 동일하게 정의

optimizier는 아래와 같이 gradient decent를 사용하고 cost가 최소화 되도록 최적화를 실행

```
cost = -tf.reduce_sum(y*tf.log(activation)) # Cross entropy
# Gradient Descent
optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)
```

Tutorial - Logistic regression

```
# Initializing the variables
init = tf.initialize_all_variables()
# Launch the graph
with tf.Session() as sess:
    sess.run(init)
    # Training cycle
    for epoch in range(training_epochs):
        avg_cost = 0.
        total_batch = int(mnist.train.num_examples/batch_size)
        # Loop over all batches
        for i in range(total_batch):
            batch_xs, batch_ys = mnist.train.next_batch(batch_size)
            # Fit training using batch data
            sess.run(optimizer, feed_dict={x: batch_xs, y: batch_ys})
            # Compute average loss
            avg_cost += sess.run(cost, feed_dict={x: batch_xs, y: batch_ys})/total_batch
        # Display logs per epoch step
        if epoch % display_step == 0:
            print "Epoch:", '%04d' % (epoch+1), "cost=", "{:.9f}".format(avg_cost)
```

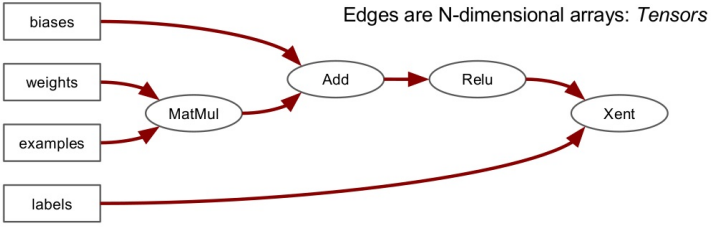
Tutorial - Logistic regression

```
# Initializing the variables
init = tf.initialize_all_variables()
# Launch the graph
with tf.Session() as sess:

    ...

    print "Optimization Finished!"
    # Test model
    correct_prediction = tf.equal(tf.argmax(activation, 1), tf.argmax(y, 1))
    # Calculate accuracy
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
    print "Accuracy:", accuracy.eval({x: mnist.test.images, y: mnist.test.labels})
```

Tutorial - Logistic regression

코드를 실행하면 아래와 같이 dataflow graph를 생성함

Computation is a dataflow graph

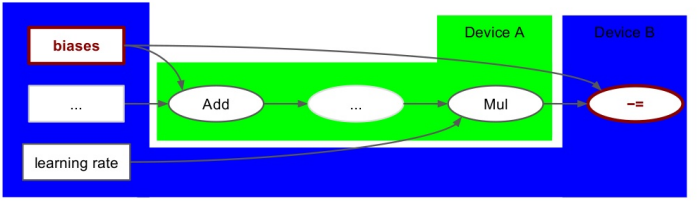


[1] Large-Scale Deep Learning for Intelligent Computer Systems

Tutorial - Logistic regression

이렇게 만들어진 그래프는 분산 처리가 가능함

Computation is a dataflow graph



Devices: Processes, Machines, GPUs, etc

[1] Large-Scale Deep Learning for Intelligent Computer Systems

Tutorial - Logistic regression

- 코드 상에는 forward pass만 있고 backward pass에 대한 내용이 없음 (Optimizer 정의를 제외하고)
- TensorFlow에서는 x, y, W, b 등을 각각의 심볼로 다루고 심볼 계산 (Symbolic computation)을 지원함
- 내부적으로 자동적으로 미분 함수를 구하고 gradient를 계산함

Tutorial - Logistic regression

Graph Visualization

아래와 같이 summary에 관한 코드를 추가

```
# Initializing the variables
init = tf.initialize_all_variables()
tf.scalar_summary("loss", cost)
merged_summary_op = tf.merge_all_summaries()
# Launch the graph
with tf.Session() as sess:
    sess.run(init)
    summary_writer = tf.train.SummaryWriter('/tmp/tf_l_regression.log', graph_def=
sess.graph_def)

    ...
```

Tutorial - Logistic regression

Graph Visualization

```
    ...
    # Training cycle
    for epoch in range(training_epochs):
        avg_cost = 0.
        total_batch = int(mnist.train.num_examples / batch_size)
        # Loop over all batches
        for i in range(total_batch):
            batch_xs, batch_ys = mnist.train.next_batch(batch_size)
            # Fit training using batch data
            sess.run(optimizer, feed_dict={x: batch_xs, y: batch_ys})
            # Compute average loss
            avg_cost += sess.run(cost, feed_dict={x: batch_xs, y: batch_ys}) / total_batch
            summary_str = sess.run(merged_summary_op, feed_dict={x: batch_xs, y:bat
ch_ys})
            summary_writer.add_summary(summary_str, epoch*total_batch + i)

            ...
```

Tutorial - Logistic regression

Graph Visualization

Tensorflow가 동작 중 로그를 지정한 경로에 남김

그런 후 아래와 같이 실행하면 http://localhost:6006 에 접속이 가능함

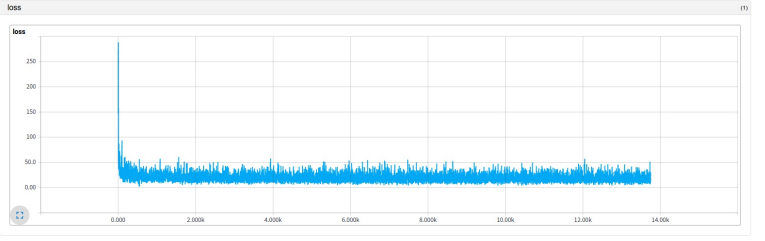
```
tensorboard --logdir=/tmp/tf_l_regression.log
```

Tutorial - Logistic regression

Graph Visualization

아래와 같이 남겨진 scalar 값은 그림과 같이 값의 변화를 확인할 수 있음

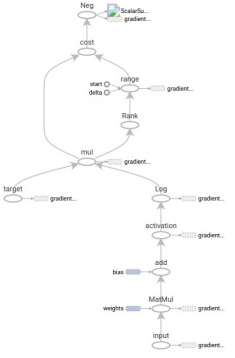
```
tf.scalar_summary("loss", cost)
```



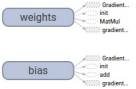
Tutorial - Logistic regression

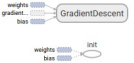
Graph Visualization

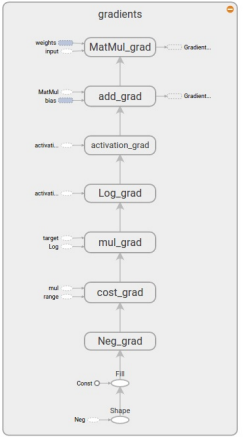
Main Graph



Auxiliary nodes

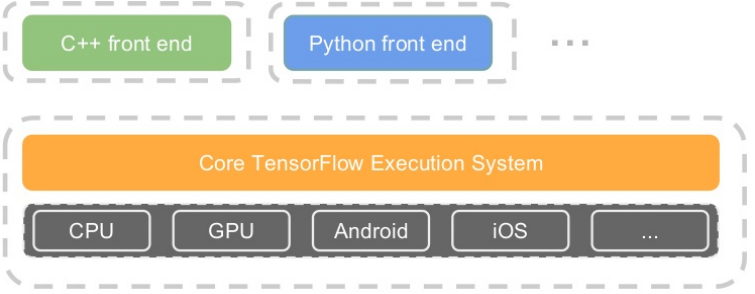






TensorFlow - 돌아와서

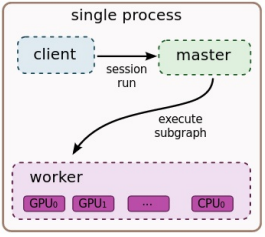
- DNN 뿐 아니라 다른 machine learing등 대규모 matrix 연산이 필요한 곳에 적용이 가능함
- 하드웨어에 대한 이해가 필요 없음
- Deep learning을 위한 다양한 최적화된 함수를 지원
- 자동 미분을 계산하여 backward pass에 대한 코드 작성이 불필요함
- 다양한 모델과 결합이 쉬움
- C++, Python front end를 지원

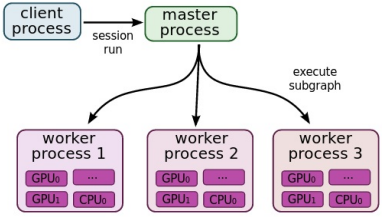


[1] Large-Scale Deep Learning for Intelligent Computer Systems

TensorFlow - 내부적으로는

- 분산, 이종/다중 디바이스를 위한 고려, 현재는 single machine 코드만 공개된 상태
- 구글에서도 해당 기능에 대해서 중요하게 생각하고 있으며 향후 공개 가능성이 높을 것으로 판단됨 [7]
- Multi-device(GPU/CPU)에 대한 동작은 지원하고 있음 [8]

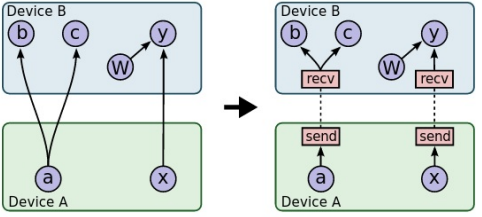




[3] TensorFlow: Large-scale machine learning on heterogeneous systems

TensorFlow - 내부적으로는

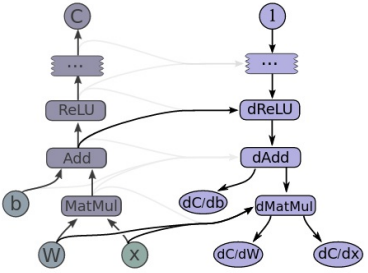
- Scheduling과 리소스(CPU, GPU, 메모리 등)별 node 배치가 핵심일 것으로 판단됨
- 노드 배치는 cost model을 사용함
  - Cost는 input, output의 크기와 operation으로 통계적으로 추정
  - 혹은 이미 실행된 배치에서 측정된 비용으로 계산됨
  - 이 비용에는 계산, 데이터 교환 비용 등이 포함됨
- 사용자에 의해 특정 작업을 특정 디바이스에 할당할 수도 있음
- 디바이스간 간선은 동적으로 send/receive node가 추가되어 디바이스로부터 추상화 시킴



[3] TensorFlow: Large-scale machine learning on heterogeneous systems

TensorFlow - 내부적으로는

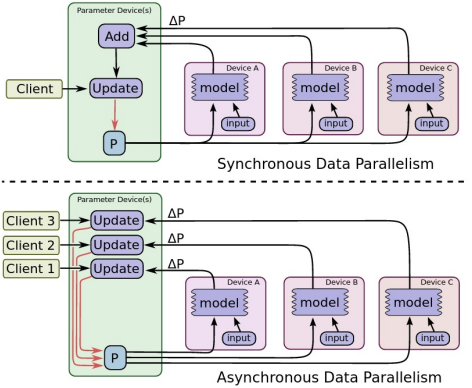
- 특별히 Gradient 기반의 그래프 계산에는 몇 가지 힌트를 사용할 수 있음
  - forward pass를 통해 계산된 결과는 backward pass에서 다시 사용됨
  - 따라서 노드 할당에 이를 고려하여 할당
  - 단, 메모리 한계가 있기 때문에 `long-lived tensor`에 대한 swapping도 같이 고려됨



[3] TensorFlow: Large-scale machine learning on heterogeneous systems

TensorFlow - 내부적으로는

- 커널 구현은 cuBLAS, cuDNN, Eigen 등을 사용
- 데이터 병렬화에 대한 내용에 대한 고려가 되어 있음
  - 그래프 생성후 디바이스별 노드 배치는 모델 병렬화의 내용
  - 아직 이 부분도 API상에서 확인되지 않음



[3] TensorFlow: Large-scale machine learning on heterogeneous systems

Tutorial - CNN

```
import tensorflow as tf
import input_data

def weight_variable(shape):
    # mean = 0.0
    initial = tf.truncated_normal(shape, stddev=0.1)
    return tf.Variable(initial)

def bias_variable(shape):
    initial = tf.constant(0.1, shape=shape)
    return tf.Variable(initial)

def conv2d(x, W):
    return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')

def max_pool_2x2(x):
    return tf.nn.max_pool(x, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
```

Tutorial - CNN

```
mnist = input_data.read_data_sets('MNIST_data/', one_hot=True)

x = tf.placeholder('float', [None, 784])
# -1 == batch size
x_image = tf.reshape(x, [-1, 28, 28, 1])

# 5x5 patch size of 1 channel, 32 features
W_conv1 = weight_variable([5, 5, 1, 32])
b_conv1 = bias_variable([32])
h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)
h_pool1 = max_pool_2x2(h_conv1)

# 5x5 patch size of 32 channel, 64 features
W_conv2 = weight_variable([5, 5, 32, 64])
b_conv2 = bias_variable([64])
h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
h_pool2 = max_pool_2x2(h_conv2)
```

Tutorial - CNN

```
# image size reduced to 7x7, full connected
W_fc1 = weight_variable([7 * 7 * 64, 1024])
b_fc1 = bias_variable([1024])

h_pool2_flat = tf.reshape(h_pool2, [-1, 7 * 7 * 64])
h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)

keep_prob = tf.placeholder('float')
h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)

W_fc2 = weight_variable([1024, 10])
b_fc2 = bias_variable([10])

y_conv = tf.nn.softmax(tf.matmul(h_fc1_drop, W_fc2) + b_fc2)
y_ = tf.placeholder('float', [None, 10])
cross_entropy = -tf.reduce_sum(y_ * tf.log(y_conv))
```

Tutorial - CNN

```
train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
correct_prediction = tf.equal(tf.argmax(y_conv, 1), tf.argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, 'float'))
```

Tutorial - RNN

Jürgen Schmidhuber[10]가 제안한 문제

time step	0	1	2	3	4	...	20	21	...	53	54
x[0]	0.5	0.7	0.3	0.1	0.2	...	0.5	0.9	...	0.8	0.2
x[1]	0	0	1	0	0		0	1		0	0

이 입력은 `0.3 + .9 = 1.2`가 정답으로 출력되어야 함

[9] rnn_example.ipynb

Tutorial - RNN

아래 코드는 이 문제를 생성하는 코드

```
def gen_data(min_length=51, max_length=55, n_batch=5):
    X = np.concatenate([np.random.uniform(size=(n_batch, max_length, 1)),
                        np.zeros((n_batch, max_length, 1))],
                       axis=-1)
    y = np.zeros((n_batch,))
    # Compute masks and correct values
    for n in range(n_batch):
        # Randomly choose the sequence length
        length = np.random.randint(min_length, max_length)
        # i changed this to a constant
        # length=55

        # Zero out X after the end of the sequence
        X[n, length:, 0] = 0
        # Set the second dimension to 1 at the indices to add
        X[n, np.random.randint(length / 2 - 1), 1] = 1
        X[n, np.random.randint(length / 2, length), 1] = 1
        # Multiply and sum the dimensions of X to get the target value
        y[n] = np.sum(X[n, :, 0] * X[n, :, 1])
    return (X, y)
```

Tutorial - RNN

```
import tensorflow as tf
import numpy as np
from tensorflow.models.rnn import rnn_cell
from tensorflow.models.rnn import rnn

# Defining some hyper-params
num_units = 2  # this is the parameter for input_size in the basic LSTM cell
input_size = 2  # num_units and input_size will be the same

batch_size = 50
seq_len = 55
num_epochs = 100
```

Tutorial - RNN

```
### Model Construction
# we use the basic LSTM cell provided in TensorFlow
cell = rnn_cell.BasicLSTMCell(num_units)
# num units is the input-size for this cell

# create placeholders for X and y
inputs = [tf.placeholder(tf.float32, shape=[batch_size, input_size]) for _ in range(se
q_len)]
result = tf.placeholder(tf.float32, shape=[batch_size])
# note that outputs is a list of seq_len
outputs, states = rnn.rnn(cell, inputs, dtype=tf.float32)
# each element is a tensor of size [batch_size,num_units]
# we actually only need the last output from the model, ie: last element of outputs
```

Tutorial - RNN

```
# We actually want the output to be size [batch_size, 1]
# So we will implement a linear layer to do this
W_o = tf.Variable(tf.random_normal([2, 1], stddev=0.01))
b_o = tf.Variable(tf.random_normal([1], stddev=0.01))

outputs2 = outputs[-1]
outputs3 = tf.matmul(outputs2, W_o) + b_o
# compute the cost for this batch of data
cost = tf.reduce_mean(tf.pow(outputs3 - result, 2))

# compute updates to parameters in order to minimize cost
train_op = tf.train.RMSPropOptimizer(0.005, 0.2).minimize(cost)
```

Tutorial - RNN

- control flow 중 loop control에 대한 지원이 없어 Python loop로 동작하여 성능 이슈가 있을 것으로 판단됨 [11]
- BLSTM, CTC은 미구현, 가변 길이의 RNN 입력이 불가

Benchmarks

convnet-benchmarks의 자료 [12]

AlexNet - Input 128x3x224x224

Library	Class	Time (ms)	forward (ms)	backward (ms)
Nervana-fp16	ConvLayer	92	29	62
CuDNN[R3]-fp16	cudnn.SpatialConvolution	96	30	66
CuDNN[R3]-fp32	cudnn.SpatialConvolution	96	32	64
Nervana-fp32	ConvLayer	101	32	69
fbfft	fbnn.SpatialConvolution	104	31	72
cudaconvnet2*	ConvLayer	177	42	135
CuDNN[R2] *	cudnn.SpatialConvolution	231	70	161
Caffe (native)	ConvolutionLayer	324	121	203
TensorFlow	conv2d	326	96	230
Torch-7 (native)	SpatialConvolutionMM	342	132	210
CL-nn (Torch)	SpatialConvolutionMM	963	388	574
Caffe-CLGreenTea	ConvolutionLayer	1442	210	123

다른 오픈 소스들

Theano

- 2009년부터 시작
- Symbolic graph 처리를 TensorFlow보다 먼저 구현한 라이브러리
- scan이라 불리는 looping control을 지원하여 RNN 구현에 용이함
- Block, Keras와 같은 Theano를 기반으로한 상위 레벨의 라이브러리가 존재함
  - Keras는 backend로 TensorFlow를 지원하고 있음
- Multi-GPU에 대한 지원이 없음
- Python 기반

Touch

- 2002년부터 시작
- RNN에 대한 지원이 없음
  - 비공식적 브랜치에서는 지원
- Yann LeCun이 contributor로 있는 프로젝트
- Lua 기반

다른 오픈 소스들

기타

- Caffe: 2013년 UC Berkeley, Yangqing jia 개발
- Deeplearning4J: 2014년 Adam Gibson이 개발
- DeepDist: 2014년 Facebook, Dirk neumann이 개발
- Apache Horn: 2012년 분산 ANN을 기반한 Hama에서 파생됨, DistBelief 오픈소스화를 위한 국내 개발자 프로젝트

[13] Evaluation of Deep Learning Toolkits

다른 오픈 소스들

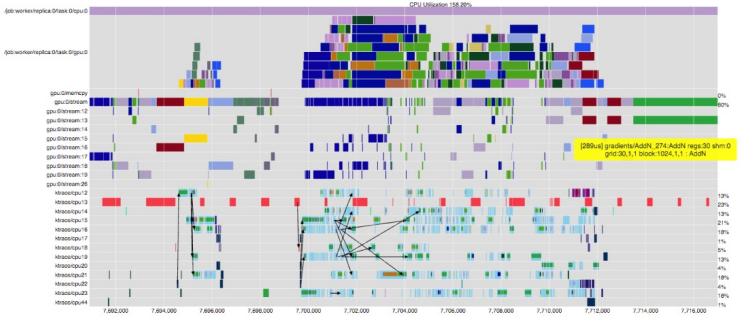
Keras

지원하는 기능이라면 구조 명세 수준

```
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation
from keras.layers.embeddings import Embedding
from keras.layers.recurrent import LSTM

model = Sequential()
model.add(Embedding(max_features, 256, input_length=maxlen))
model.add(LSTM(output_dim=128, activation='sigmoid', inner_activation='hard_si
gmoid'))
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation('sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='rmsprop')

model.fit(X_train, Y_train, batch_size=16, nb_epoch=10)
score = model.evaluate(X_test, Y_test, batch_size=16)
```

TensorFlow v0.5

- **분산 처리 관련 구현** 미공개
- C++ front end를 지원하나 Python쪽에 집중되어 있음
- C++에서는 그래프 생성, 자동 미분 계산 등이 누락되어 있음 [5]
- C++ 예제 코드는 Python에서 그래프 생성하여 파일(`protocal buffer` 포맷)로 저장 후 로딩하도록 되어 있음
- 그래프 생성 코드는 Python으로 작성된 것으로 추정됨
- EEG라 불리는 성능 프로파일링 툴은 미공개 상태



[3] TensorFlow: Large-scale machine learning on heterogeneous systems

TensorFlow를 고려한다면

- 작은 규모의 파라미터에서도 나쁘지 않은 성능
- 아직 분산처리는 공개되지 않았지만 단일 GPU에서 모델이 다 올라가지 않는다면 대안을 가진 라이브러리는 많지 않음
- RNN쪽 지원은 아직 더 기다려야 할 것 같음
- v0.5에 비해 나쁘지 않는 완성도
  - 구느님의 V8 엔진 같은 신적화를 기대해 볼 수 있을지도
- TensorFlow뿐 아니라 다른 라이브러리도 구현 코드가 줄어 들어 모델 자체에 집중할 수 있을 것으로 보임
  - Keras같은 high-level 라이브러리를 선택하여 backend를 변경해도 될 것 같음
  - 이제 WYSIWYG 툴만 나오면 됨

설치

- GPU 요구 사항
  - **CUDA 7.0, cuDNN 6.5**
  - CUDA(Compute capability) 버전 3.5 이상, 비공식 적으로는 CUDA 3.0 GPU도 지원
  - Titan, Tesla(K20, K40) 권장
  - **Multiprocessor 최소 8개 이상**
- OS: Linux, OSX
  - **Python 2.7**
  - Windows는 아직 지원하지 않음
  - Docker Toolbox를 통해 시도는 가능

버전이 다른 라이브러리는 시도하지 않는 것이 정신 건강에 좋음

설치

TensorFlow를 동작하는 방법은 3가지 방법이 있음

- Docker 이미지
  - 가장 편안한 방법
  - Docker 사용법을 익혀야 함
- Python 패키지
  - 로컬에서 작업 가능
  - 소스 컴파일이 필요한 작업은 확인 불가능, 몇몇 예제는 bazel 빌드로 동작
- 소스 빌드
  - 가장 어려운 방법
  - 약간은 bazel을 익혀야 함

설치 - Docker

tensorflow, tensorflow-full 두 가지 이미지가 있으며 tensorflow-full 은 소스가 포함되어 있음

```
docker run -p 127.0.0.1:8888:8888 -it b.gcr.io/tensorflow/tensorflow-full
```

`-p 127.0.0.1:8888:8888`는 jupyter(iPython) 포트를 포워딩 하기 위함

아래와 같이 jupyter 실행이 가능함

```
root@bacf85f6aea3:~# /run_jupyter.sh
```

설치 - Docker (GPU)

Docker tensorflow, tensorflow-full 이미지는 GPU를 지원하지 않음
Docker 내부에서 GPU를 동작하게 하려면 소스를 받아야 함

```
git clone --recurse-submodules https://github.com/tensorflow/tensorflow
```

GPU 사용시 아래와 같은 환경 변수가 등록되어야 함

```
export LD_LIBRARY_PATH="$LD_LIBRARY_PATH:/usr/local/cuda/lib64"
export CUDA_HOME=/usr/local/cuda
```

`tensorflow/tensorflow/tools/docker`에 `docker_run_gpu.sh`를 실행하면 local의 환경 변수를 읽어 docker 이미지를 생성함

설치 - Python

Python 2.7을 사용함 아직 **3.0은 지원하지 않음**, 만약 여러 버전이 공존한다면 virtualenv 사용을 권장

`python-pip`, `python-dev`, `python-virtualenv`는 설치되어 있어야 함, 아래 예시는 GPU 지원 pip 패키지를 설치

```
$ virtualenv --system-site-packages -p /usr/bin/python2.7 tensorflow
$ source ~/tensorflow/bin/activate

(tensorflow)$ pip install --upgrade https://storage.googleapis.com/tensorflow/linux
/gpu/tensorflow-0.5.0-cp27-none-linux_x86_64.whl
```

GPU 사용시 아래와 같은 환경 변수가 등록되어야 함

```
export LD_LIBRARY_PATH="$LD_LIBRARY_PATH:/usr/local/cuda/lib64"
export CUDA_HOME=/usr/local/cuda
```

설치 - Python

정상 설치 되었다면 아래 명령이 오류를 발생하지 않음

```
(tensorflow)$ python -c 'import tensorflow'
```

만약 아래와 같은 오류가 발생한다면 `pip install 'protobuf>=3.0.0a3'`로 protobuf 버전을 일치시켜야 함

```
  File "/home/shchoi/pyenv/tensorflow/local/lib/python2.7/site-packages/tensorflow/core/framework/tensor_shape_pb2.py", line 22, in <module>
    serialized_pb=_b('₩n,tensorflow/core/framework/tensor_shape.proto₩x12₩ntensorflow₩"d₩n₩x10TensorShapeProto₩x12-₩n₩x03₩x64im₩x18₩x02 ₩x03(₩x0b₩x32 .tensorflow.TensorShapeProto.Dim₩x1a!₩n₩x03₩x44im₩x12₩x0c₩n₩x04size₩x18₩x01 ₩x01(₩x03₩x12₩x0c₩n₩x04name₩x18₩x02 ₩x01(₩tb₩x06proto3')
TypeError: __init__() got an unexpected keyword argument 'syntax'
```

설치 - 소스

소스 클론 및 관련 라이브러리 설치, 아울러 구글이 만든 빌드 툴인 Bazel 을 설치해야 함

```
$ git clone --recurse-submodules https://github.com/tensorflow/tensorflow
$ sudo apt-get install python-numpy swig python-dev
```

설치 - 소스

configure 실행

```
$ cd tensorflow
$ ./configure
Do you wish to build TensorFlow with GPU support? [y/n] y
GPU support will be enabled for TensorFlow

Please specify the location where CUDA 7.0 toolkit is installed. Refer to
README.md for more details. [default is: /usr/local/cuda]: /usr/local/cuda

Please specify the location where CUDNN 6.5 V2 library is installed. Refer to
README.md for more details. [default is: /usr/local/cuda]: /usr/local/cuda

Setting up Cuda include
Setting up Cuda lib64
Setting up Cuda bin
Setting up Cuda nvvm
Configuration finished
```

설치 - 소스

만약 CUDA 3.0 GPU라면 아래와 같이 실행하여 설정이 가능함

```
TF_UNOFFICIAL_SETTING=1 ./configure
```

설치 - 소스

예제 프로그램 빌드 및 실행

```
$ bazel build -c opt --config=cuda //tensorflow/cc:tutorials_example_trainer
$ bazel-bin/tensorflow/cc/tutorials_example_trainer
...
000008/000007 lambda = 2.000000 x = [0.894427 -0.447214] y = [1.788854 -0.8
94427]
000002/000000 lambda = 2.000000 x = [0.894427 -0.447214] y = [1.788854 -0.8
94427]
000004/000000 lambda = 2.000000 x = [0.894427 -0.447214] y = [1.788854 -0.8
94427]
000008/000007 lambda = 2.000000 x = [0.894427 -0.447214] y = [1.788854 -0.8
94427]
000002/000000 lambda = 2.000000 x = [0.894427 -0.447214] y = [1.788854 -0.8
94427]
000008/000007 lambda = 2.000000 x = [0.894427 -0.447214] y = [1.788854 -0.8
94427]
000004/000000 lambda = 2.000000 x = [0.894427 -0.447214] y = [1.788854 -0.8
94427]
$
```

설치 - 소스

아래와 같은 오류가 발생한다면 구글링 필요, 대부분 환경 설정 관련 오류

```
bazel build -c opt //tensorflow/cc:tutorials_example_trainer
.....
WARNING: Sandboxed execution is not supported on your system and thus hermeti
city of actions cannot be guaranteed. See http://bazel.io/docs/bazel-user-manual.ht
ml#sandboxing for more information. You can turn off this warning via --ignore_un
supported_sandboxing.
ERROR: Loading of target '//tools/jdk:ijar' failed; build aborted: no such package 'to
ols/jdk': BUILD file not found on package path.
ERROR: Loading failed; build aborted.
INFO: Elapsed time: 0.565s
```

이 경우는 `.bazelrc`를 설정하여 해결

```
build --package_path %workspace%:/home/shchoi/share/src/bazel-0.1.1/base_wo
rkspace
fetch --package_path %workspace%:/home/shchoi/share/src/bazel-0.1.1/base_wo
rkspace
query --package_path %workspace%:/home/shchoi/share/src/bazel-0.1.1/base_wo
rkspace
```

설치 - 소스

Python을 위한 pip package는 아래와 같이 빌드가 가능함

```
bazel build -c opt //tensorflow/tools/pip_package:build_pip_package
```

참고 자료

- [1] Large-Scale Deep Learning forIntelligent Computer Systems
- [2] Introduction to apache horn (incubating)
- [3] TensorFlow: Large-scale machine learning on heterogeneous systems
- [4] Large Scale Distributed Deep Networks
- [5] Loading a TensorFlow graph with the C++ API
- [6] MNIST For ML Beginners
- [7] Github - Distributed Version
- [8] Using GPUs
- [9] rnn_example.ipynb
- [10] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.
- [11] Symbolic loops (like "scan" in Theano)
- [12] convnet-benchmarks
- [13] Evaluation of Deep Learning Toolkits