Machine Learning

Theano, Keras, Tensorflow

Dom Huh

What is Theano?

Mathematical symbolic expression compiler

Makes it easy to define, manipulate, and compute expressions with NumPy syntax

This serves as the base for many higher level APIs modules, like sklearn, keras, block, and many more.

Has GPU capabilities with CUDA (NVIDIA…)

THEANO_FLAGS = mode=FAST_RUN, device =gpu, floatX=float32 python net_name.py

Uses theano vectors, matrices, tensors, scalar

Theano

What is Theano?

Mathematical symbolic expression compiler

Makes it easy to define, manipulate, and compute expressions with NumPy syntax

This serves as the base for many higher level APIs modules, like sklearn, keras, block, and many more.

How to use Theano

Initialize inputs:

x = T.vector(“symbolic name of featureset’’)

Ndim, dtype(ie. ivector, fmatrix, dtensor4), patterns, device are specified

Shape, memory layout are not

Shared variables:

Values are persistent and shared throughout functions.

They can be updated

W= theano.shared(weights_values)

How to use Theano

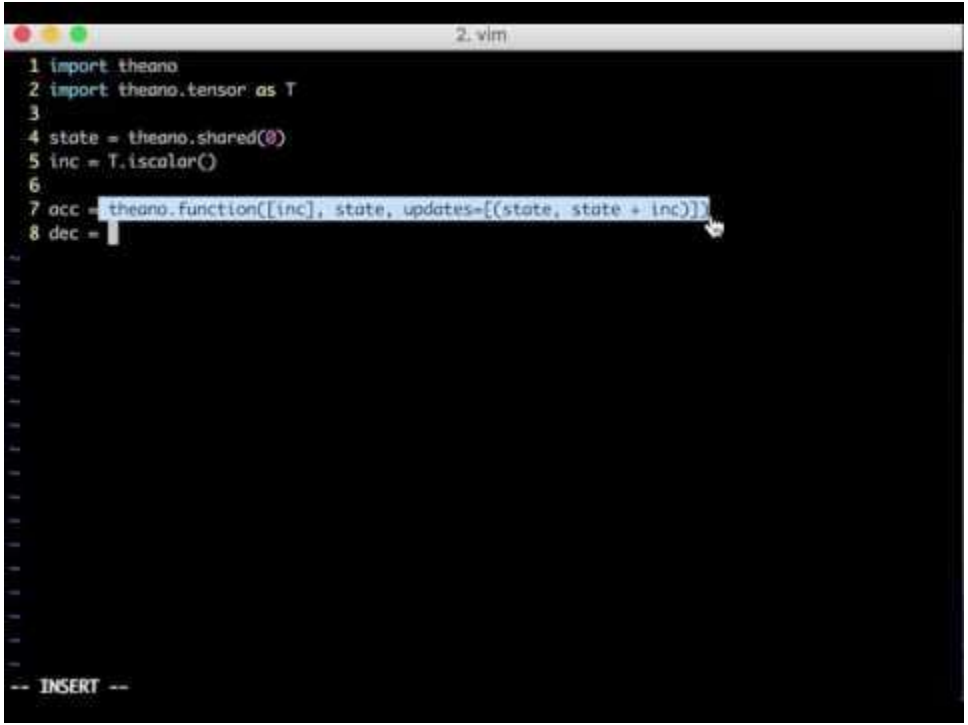
Math APIs:

```
Dot = T.dot(x,W) #dot product

weight_Cost = theano.grad(C,W) #gradient
```

Theano functions:

```
Funct_name = theano.function([input], s_v,
                                        update=[(s_v, <action on s_v with input>)])

Funct_name(input)
```

Example: Shared variables + Functions



Optimization with Theano

Mode = “FAST_RUN”

Other modes:

- “FAST_COMPILE”
- “DEBUG_MODE”

Optimizer = ‘fast_compile”

Can be done either locally or globally

Linear Regression with Theano

$$MSE = 1N\sum_{i=1n}(y_i-(mx_i+b))2$$

Logistic Regression with Theano

Code on:

http://deeplearning.net/tutorial/code/logistic_sgd.py

ConvNet and LSTM with Theano

ConvNet:

http://deeplearning.net/tutorial/code/convolutional_mlp.py

LSTM:

http://deeplearning.net/tutorial/code/lstm.py

Keras

NN library, built on top of either Theano or TF (use as backend)

High level APIs

Very simple

General structure:

- I/O tensors
- Layers

keras.layers.core.Dense(output_dim, activation_funct, bias, inits*, regularizers*, constraints*, input_dim)

Core layers: https://github.com/keras-team/keras/blob/master/keras/layers/core.py#L762

keras.layers.recurrent.GRU(...)

Example:

encoder,decoder =keras.layers.containers.Sequential(...)

Autoencoder = Sequential()

Autoencoder.add(encoder,decoder,...)

Additional Information about Keras

Layers:

Core, convolutional, pooling, noise, dropout, normalization, embedding, recurrent...

More details on : https://github.com/keras-team/keras/tree/master/keras/layers

Activations:

Sigmoid, tanh, ReLU, softplus, linear, LeakyReLU...

Optimizers:

SGD, Adagrad, Adadelta, RMSprop, ADAM

Objective Function:

RMSE, MSE, hinge, binary_crossentropy, categorical_crossentropy

Additional Information about Keras

Save architectures (json, yaml):

```
json_string = model.to_json()

model = model_from_json(json_string)
```

Save parameters:

```
model.save_weights("file_name")

model.load_weights("file_name")
```

Callbacks:

...

Model Types:

Sequential, graph

To train:

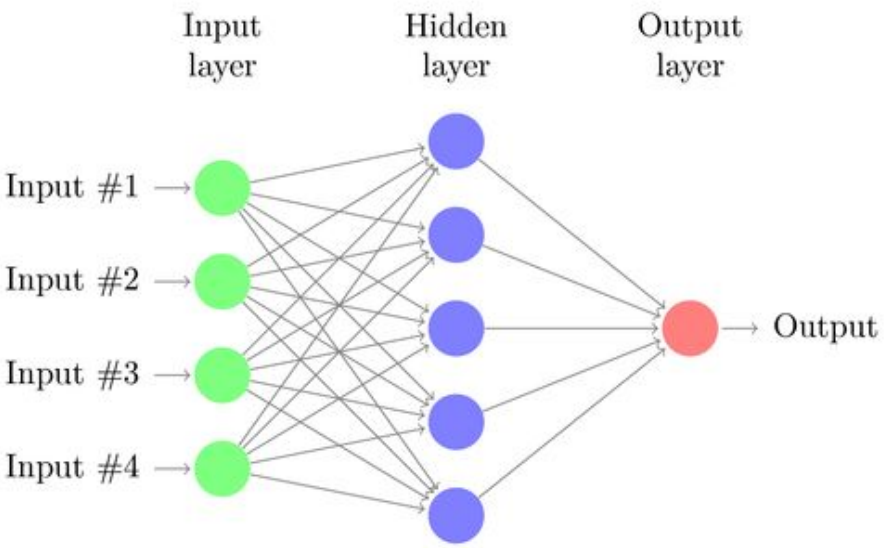
```
compile(loss,optimizer)

fit(features,labels,batch_size,n_epochs,...)
```

Sequential Model Type

Linear stack of layers

Treat each layer as object that feeds into the next

Ie: Autoencoder



Keras Implementation

https://github.com/keras-team/keras/blob/master/examples/mnist_dataset_api.py

PyTorch

Imperative programming

Dynamic computational graphs

Debug friendly

Easy to read

Useful for research (DCG)

Linear regression w/ pytorch

https://github.com/pytorch/examples/blob/master/regression/main.py

https://colab.research.google.com/drive/1k0n4wdnubEgI-KnQhLrFBwmTGU-xmCNP

Tensorflow Implementation of MLP

Tensorflow Implementation

https://colab.research.google.com/github/tensorflow/models/blob/master/samples/core/tutorials/keras/basic_classification.ipynb