Keras: Handwritten Digit Recognition using MNIST Dataset

IIT PATNA

January 31, 2018

OUTLINE

Architecture of a Neural Network

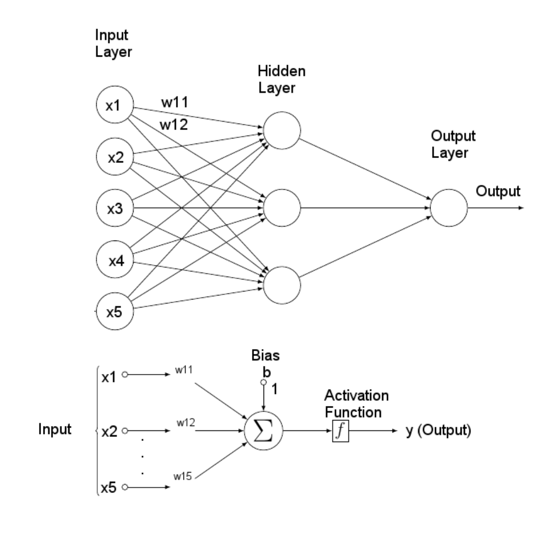

Figure : A Neural Network

Keras

Keras is

- high-level neural networks library
- written in Python
- capable of running on top of
  - TensorFlow (open source software library for numerical computation)
  - Theano (numerical computation library for Python)
  - CNTK (Microsoft Cognitive Toolkit): Deep learning framework
- developed with a focus on enabling fast experimentation (through user friendliness, modularity, and extensibility)

Guiding principles

- Modularity
  - configurable modules
    - neural layers, cost functions, optimizers, initialization schemes, activation functions, regularization schemes are all standalone modules that you can combine to create new models
- Minimalism
  - Each module should be kept short and simple
- Easy extensibility
  - New modules are simple to add (as new classes and functions)
  - suitable for advanced research
- Work with Python
  - Models are described in Python code, which is compact, easier to debug, and allows for ease of extensibility

Dependencies

- Python 2.7+
- numpy: fundamental package for scientific computing with Python
- scipy: library used for scientific computing and technical computing
- Matplotlib (Optional, recommended for exploratory analysis)
- HDF5 and h5py (Optional, required if you use model saving/loading functions)
- Theano

Installation

- Follow instructions provided in "keras installation" file

- Alternatively you may visit Keras Installation Page

Keras provides

- Models
- Layers
- Preprocessing
- Metrics
- Optimizers
- Activations
- Datasets
- Constraints
- and many more...

Model

- Model
  - core data structure of Keras
  - a way to organize layers
- Two types:
  - Sequential
  - Model class API
- Sequential Model: a linear stack of layers
- functional API: for defining complex models, such as models with shared layers

Layers

- Core Layers
  - Dense
  - Activation
  - Dropout
  - Flatten
  - many more ...
- Convolutional Layers
- Pooling Layers
- Recurrent Layers
- Your own Keras layers
- and many more ...

Core Layers

Dense

- fully connected NN layer: connection to all activation in previous layer

Activation

- Applies an activation function
  - softmax: usually used on the output layer to turn the outputs into probability-like values
  - relu: rectified linear unit (ReLU), most popular activation function, $f(x) = max(x, 0)$
  - linear
  - and many more...

Dropout

- Applies Dropout to the input
- randomly setting a fraction p of input units to 0
- prevent overfitting

Keras provides

Optimizer

- the specific algorithm used to update weights while we train our model
- such as *sgd* (Stochastic gradient descent optimizer)

objective function or loss function

- used by the optimizer to navigate the space of weights
- such as *mse* (mean squared error)

metrics

- used to judge the performance of your model
- such as *accuracy*

Building a Simple Deep Learning Network Using Keras

Steps

- Import libraries and modules
- Load image data
- Preprocess data
- Define model architecture
- Compile model
- Fit and evaluate Model
- Improvements

- Henceforth, the example file complements the (incomplete) slides

Building a Simple Deep Learning Network Using Keras

Import libraries and modules

- as in example file

Dataset

- Keras provides in-built support to many datasets
- such as MNIST
  - database of handwritten digits
  - used extensively in optical character recognition and machine learning research
  - training set of 60,000 examples, and a test set of 10,000 examples
  - digits have been size-normalized and centered in a fixed-size image
  - black and white digits
  - 28×28 pixels
  - Keras provides method to load MNIST data set (example file)

Building a Simple Deep Learning Network Using Keras

Preprocessing input data for Keras

- With Theano backend, the depth of the input image must be declared explicitly
- MNIST images have a depth of 1
- Also, convert data type to *float32* and normalize values
- as in example

Preprocessing class labels for Keras

- 10 different classes, one for each digit
- as in example

Model Architecture

- Usually most time consuming
- Use sequential model
- a Sequential model is declared as
  model = Sequential()
- Adding layers
  - The model needs to know what input shape it should expect
  - first layer in a Sequential model (and only the first, because following layers can do automatic shape inference) needs to receive information about its input shape
  - Dense(32, input_dim=784) specifies that it is
    - first (input) layer
    - output dimension is 32 (1^{st} argument
    - input dimension is 784
    - If no activation function specified, no activation is applied (ie. "linear" activation: $a(x) = x$).

Model architecture

- one hidden layer with the same number of neurons as there are inputs (784)
- init: name of initialization function for the weights of the layer. normal for values randomly drawn from normal distribution.
- there are many other initializations available in Keras
- rectifier activation function is used for the neurons in the hidden layer
- softmax activation function is used on the output layer to turn the outputs into probability-like values and allow one class of the 10 to be selected as the models output prediction

Compile model

- Before training, configure the learning process, using compile() method. Three argements:
  - optimizer: ANN training process is an optimization task with the aim of finding a set of weights to minimize some objective function
  - loss function: the objective function that model try to minimize
  - list of metrics: used to judge performance of model, similar to objective function however not used for training purpose
- Logarithmic loss is used as the loss function
- ADAM gradient descent algorithm is used to learn the weights

Train and Evaluate model

Train model

using fit() function

Evaluate model on test data

using evaluate() function

Performance of Simple Network

temp : bash – Konsole

File Edit View Bookmarks Settings Help

```
niraj@niraj-Veriton-M200-Q87:~/temp$ python 1.py
Using Theano backend.
(60000, 28, 28)
(60000,)
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
dense_1 (Dense)                  (None, 784)           615440      dense_input_1[0][0]
____________________________________________________________________________________________________
dense_2 (Dense)                  (None, 10)            7850        dense_1[0][0]
====================================================================================================
Total params: 623,290
Trainable params: 623,290
Non-trainable params: 0
____________________________________________________________________________________________________
None
Train on 60000 samples, validate on 10000 samples
Epoch 1/2
60000/60000 [==============================] - 4s - loss: 0.2744 - acc: 0.9221 - val_loss: 0.1356 - val_acc: 0.9601
Epoch 2/2
60000/60000 [==============================] - 4s - loss: 0.1078 - acc: 0.9688 - val_loss: 0.0957 - val_acc: 0.9707
 9600/10000 [===========================>..] - ETA: 0s Error: 2.93%
niraj@niraj-Veriton-M200-Q87:~/temp$ 
```

temp : bash

Improving Performance of Simple Network: additional hidden layers

temp : bash – Konsole

File Edit View Bookmarks Settings Help

```
niraj@niraj-Veriton-M200-Q87:~/temp$ python 1.py
Using Theano backend.
(60000, 28, 28)
(60000,)
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
dense_1 (Dense)                  (None, 784)           615440      dense_input_1[0][0]
____________________________________________________________________________________________________
dense_2 (Dense)                  (None, 784)           615440      dense_1[0][0]
____________________________________________________________________________________________________
dense_3 (Dense)                  (None, 10)            7850        dense_2[0][0]
====================================================================================================
Total params: 1,238,730
Trainable params: 1,238,730
Non-trainable params: 0
____________________________________________________________________________________________________
None
Train on 60000 samples, validate on 10000 samples
Epoch 1/2
60000/60000 [==============================] - 8s - loss: 0.2184 - acc: 0.9354 - val_loss: 0.1094 - val_acc: 0.9639
Epoch 2/2
60000/60000 [==============================] - 8s - loss: 0.0755 - acc: 0.9767 - val_loss: 0.0852 - val_acc: 0.9720
 9824/10000 [============================>.] - ETA: 0s Error: 2.80%
niraj@niraj-Veriton-M200-Q87:~/temp$
```

temp : bash

Improving Performance of Simple Network: additional hidden layers

temp : bash – Konsole

File Edit View Bookmarks Settings Help

```
niraj@niraj-Veriton-M200-Q87:~/temp$ python 1.py
Using Theano backend.
(60000, 28, 28)
(60000,)
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
dense_1 (Dense)                  (None, 784)           615440      dense_input_1[0][0]
____________________________________________________________________________________________________
dense_2 (Dense)                  (None, 784)           615440      dense_1[0][0]
____________________________________________________________________________________________________
dense_3 (Dense)                  (None, 784)           615440      dense_2[0][0]
____________________________________________________________________________________________________
dense_4 (Dense)                  (None, 10)            7850        dense_3[0][0]
====================================================================================================
Total params: 1,854,170
Trainable params: 1,854,170
Non-trainable params: 0
____________________________________________________________________________________________________
None
Train on 60000 samples, validate on 10000 samples
Epoch 1/2
60000/60000 [==============================] - 11s - loss: 0.1999 - acc: 0.9388 - val_loss: 0.0950 - val_acc: 0.9712
Epoch 2/2
60000/60000 [==============================] - 12s - loss: 0.0751 - acc: 0.9770 - val_loss: 0.0914 - val_acc: 0.9738
 9920/10000 [============================>.] - ETA: 0s Error: 2.62%
niraj@niraj-Veriton-M200-Q87:~/temp$
```

temp : bash

Improving Performance of Simple Network: introducing dropout layer

temp : bash – Konsole

File Edit View Bookmarks Settings Help

```
niraj@niraj-Veriton-M200-Q87:~/temp$ python 1.py
Using Theano backend.
(60000, 28, 28)
(60000,)
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
dense_1 (Dense)                  (None, 784)           615440      dense_input_1[0][0]
____________________________________________________________________________________________________
dense_2 (Dense)                  (None, 784)           615440      dense_1[0][0]
____________________________________________________________________________________________________
dense_3 (Dense)                  (None, 784)           615440      dense_2[0][0]
____________________________________________________________________________________________________
dropout_1 (Dropout)              (None, 784)           0           dense_3[0][0]
____________________________________________________________________________________________________
dense_4 (Dense)                  (None, 10)            7850        dropout_1[0][0]
====================================================================================================
Total params: 1,854,170
Trainable params: 1,854,170
Non-trainable params: 0
____________________________________________________________________________________________________
None
Train on 60000 samples, validate on 10000 samples
Epoch 1/2
60000/60000 [==============================] - 13s - loss: 0.2014 - acc: 0.9386 - val_loss: 0.1017 - val_acc: 0.9697
Epoch 2/2
60000/60000 [==============================] - 14s - loss: 0.0771 - acc: 0.9760 - val_loss: 0.0811 - val_acc: 0.9740
10000/10000 [==============================] - 1s
 Error: 2.60%
niraj@niraj-Veriton-M200-Q87:~/temp$
```

temp : bash

Improving Performance of Simple Network: using different optimizers

```
niraj@niraj-Veriton-M200-Q87:~/temp$ python 1.py
Using Theano backend.
(60000, 28, 28)
(60000,)
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
dense_1 (Dense)                  (None, 784)           615440      dense_input_1[0][0]
____________________________________________________________________________________________________
dense_2 (Dense)                  (None, 784)           615440      dense_1[0][0]
____________________________________________________________________________________________________
dense_3 (Dense)                  (None, 784)           615440      dense_2[0][0]
____________________________________________________________________________________________________
dropout_1 (Dropout)              (None, 784)           0           dense_3[0][0]
____________________________________________________________________________________________________
dense_4 (Dense)                  (None, 10)            7850        dropout_1[0][0]
====================================================================================================
Total params: 1,854,170
Trainable params: 1,854,170
Non-trainable params: 0
____________________________________________________________________________________________________
None
Train on 60000 samples, validate on 10000 samples
Epoch 1/2
60000/60000 [==============================] - 9s - loss: 1.0352 - acc: 0.7382 - val_loss: 0.4882 - val_acc: 0.8791
Epoch 2/2
60000/60000 [==============================] - 10s - loss: 0.4422 - acc: 0.8784 - val_loss: 0.3497 - val_acc: 0.9051
 9984/10000 [============================>.] - ETA: 0s Error: 9.49%
niraj@niraj-Veriton-M200-Q87:~/temp$
```

Improving Performance of Simple Network: training for more number of epochs

temp : bash – Konsole

File Edit View Bookmarks Settings Help

```
niraj@niraj-Veriton-M200-Q87:~/temp$ python 1.py
Using Theano backend.
(60000, 28, 28)
(60000,)
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
dense_1 (Dense)                  (None, 784)           615440      dense_input_1[0][0]
____________________________________________________________________________________________________
dense_2 (Dense)                  (None, 784)           615440      dense_1[0][0]
____________________________________________________________________________________________________
dense_3 (Dense)                  (None, 784)           615440      dense_2[0][0]
____________________________________________________________________________________________________
dropout_1 (Dropout)              (None, 784)           0           dense_3[0][0]
____________________________________________________________________________________________________
dense_4 (Dense)                  (None, 10)            7850        dropout_1[0][0]
====================================================================================================
Total params: 1,854,170
Trainable params: 1,854,170
Non-trainable params: 0
____________________________________________________________________________________________________
None
Train on 60000 samples, validate on 10000 samples
Epoch 1/20
60000/60000 [==============================] - 10s - loss: 1.0537 - acc: 0.7378 - val_loss: 0.4933 - val_acc: 0.8815
Epoch 2/20
60000/60000 [==============================] - 9s - loss: 0.4407 - acc: 0.8813 - val_loss: 0.3505 - val_acc: 0.9054
Epoch 3/20
60000/60000 [==============================] - 10s - loss: 0.3512 - acc: 0.9016 - val_loss: 0.3008 - val_acc: 0.9187
Epoch 4/20
```

temp : bash

Improving Performance of Simple Network: training for more number of epochs

temp : bash – Konsole

File Edit View Bookmarks Settings Help

```
60000/60000 [==============================] - 9s - loss: 0.2415 - acc: 0.9314 - val_loss: 0.2227 - val_acc: 0.9377
Epoch 8/20
60000/60000 [==============================] - 10s - loss: 0.2280 - acc: 0.9348 - val_loss: 0.2114 - val_acc: 0.9404
Epoch 9/20
60000/60000 [==============================] - 9s - loss: 0.2150 - acc: 0.9386 - val_loss: 0.2007 - val_acc: 0.9428
Epoch 10/20
60000/60000 [==============================] - 9s - loss: 0.2036 - acc: 0.9420 - val_loss: 0.1931 - val_acc: 0.9454
Epoch 11/20
60000/60000 [==============================] - 10s - loss: 0.1934 - acc: 0.9446 - val_loss: 0.1835 - val_acc: 0.9477
Epoch 12/20
60000/60000 [==============================] - 10s - loss: 0.1845 - acc: 0.9476 - val_loss: 0.1775 - val_acc: 0.9497
Epoch 13/20
60000/60000 [==============================] - 10s - loss: 0.1757 - acc: 0.9500 - val_loss: 0.1714 - val_acc: 0.9508
Epoch 14/20
60000/60000 [==============================] - 9s - loss: 0.1689 - acc: 0.9516 - val_loss: 0.1649 - val_acc: 0.9525
Epoch 15/20
60000/60000 [==============================] - 10s - loss: 0.1614 - acc: 0.9541 - val_loss: 0.1584 - val_acc: 0.9532
Epoch 16/20
60000/60000 [==============================] - 10s - loss: 0.1546 - acc: 0.9556 - val_loss: 0.1549 - val_acc: 0.9547
Epoch 17/20
60000/60000 [==============================] - 9s - loss: 0.1484 - acc: 0.9583 - val_loss: 0.1491 - val_acc: 0.9564
Epoch 18/20
60000/60000 [==============================] - 10s - loss: 0.1429 - acc: 0.9593 - val_loss: 0.1455 - val_acc: 0.9565
Epoch 19/20
60000/60000 [==============================] - 10s - loss: 0.1373 - acc: 0.9611 - val_loss: 0.1412 - val_acc: 0.9579
Epoch 20/20
60000/60000 [==============================] - 10s - loss: 0.1324 - acc: 0.9623 - val_loss: 0.1381 - val_acc: 0.9583
10000/10000 [==============================] - 1s
 Error: 4.17%
niraj@niraj-Veriton-M200-Q87:~/temp$
```

temp : bash

Improving Performance of Simple Network

other options to explore

- different learning rate for optimizer
- number of neurons in hidden layer
- batch size

Constructing the Right Network

steps to follow to make an efficient image classifier?

- lot of experimentation and testing to get the optimal structure and parameters

Important Links

Links

1. Keras Official Documentation Page
2. Keras GitHub page
3. Another GitHub Page
4. GitHub Page MNIST example
5. Keras Tutorial
6. An Example
7. Another Example
8. Deep Learning with Keras (Book)

The End