Keras

Machine Learning II

Lecture 6 - b

THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

- Keras : Francois Chollet
- PyTorch – Facebook AI research
- TensorFlow – Google
- CNTK - Microsoft
- Caffe - Berkeley Vision and Learning
- Mxnet: Amzoon

- Why we need deep learning frameworks?
- What is a computational graphs?
- Easily build big computational graphs
- Easily compute gradients in computational graphs
- Run it all efficiently on GPU
- Why we need Keras?

What is Keras ?

- Deep neural network library in Python
- Neural networks API.
- Modular.
- Useful for fast prototyping, ignoring the details of implementing backprop.
- Almost any architecture can be designed using this framework.
- Open Source code and community support.

1- Preparing the inputs and targets - specify the dimension and sizes.

2- Define the NN model architecture and build the computational graph.

3- Setting optimizer and the learning process.

4- Specify the Inputs, Outputs of the computational graph in the model and configure loss function.

5- Train and test the model on the dataset.

Three API styles

1- The Sequential Model

It is simple – Only for single-input, single-output; Limited Cases

2- The functional API

Multi input multi-output – Can play with models – Most for cases

3- Model subclassing

Maximum flexibility

- Keras has a number of pre-built layers.

- Regular dense, MLP type (Sample Code).
- Recurrent layers, LSTM, GRU, etc(Sample Code).
- 1D Convolutional layers (Sample Code).
- 2D Convolutional layers (Sample Code).
- Convolutional LSTM (Sample Code).

Other Layers

- Other types of layer include:

- Dropout (Sample Code).
- Normalization (Sample Code).
- Pooling (Sample Code).
- Embedding (Sample Code).
- Noise (Sample Code).

- Most your activations functions are available:

- **Regular activations functions:** *Sigmoid, tanh, ReLu, softplus, hard sigmoid, linear*
- **Advanced activations functions:** *LeakyReLu, PReLu, ELU, Parametric Softplus, Thresholded linear and Thresholded Relu*
- **Custom activations functions**

Performance Index and Optimizers

- Performance Index:
- **Error loss**: *rmse, mse, mae, mape, msle*
- **Class loss**: *binary crossentropy, categorical crossentropy*

- Optimization:
- **First Order:** *SGD, Adagrad, Adadelta, Rmsprop and Adam*
- All optimizers has its own hyper parameters

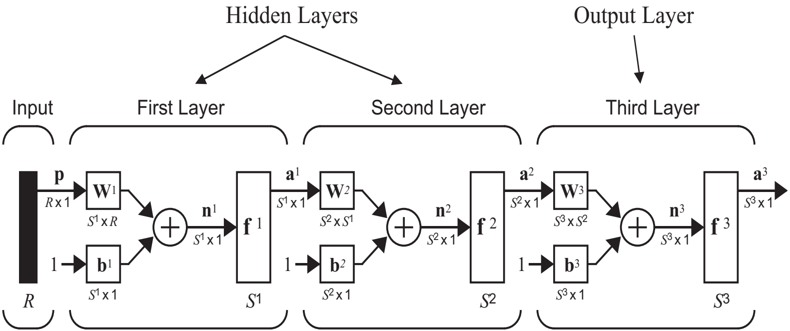
```python
import keras
from keras import layers
import numpy as np
x = np.linspace(-4,4,500)
y = np.sin(x)

model = keras.Sequential()

model.add(layers.Dense(10, activation='relu' , input_shape=(1,)))
model.add(layers.Dense(1, activation='linear' , input_shape=(10,)))

model.compile(loss='mse',optimizer='sgd',metrics=['accuracy'])
model.fit(x, y, epochs=10)
```

```
import keras
from keras import layers
import numpy as np
x = np.linspace(-4,4,500)
y = np.sin(x)

inputs = keras.Input(shape=(1,))
x1 =layers.Dense(10,activation='relu')(inputs)
ouputs = layers.Dense(1, activation='relu')(x1)

model = keras.Model(inputs, ouputs)
model.compile(loss='mse',optimizer='sgd',metrics=['accuracy'])
model.fit(x, y, epochs=10)
```

```python
import keras
from keras import layers
import numpy as np
x = np.linspace(-4,4,500).reshape(-1,1)
y = np.sin(x)

class Mymodel(keras.Model):
    def __init__(self):
        super(Mymodel,self).__init__()
        self.dense1 = layers.Dense(1, activation='relu')
        self.dense2 = layers.Dense(10, activation='relu')
        self.dense3 = layers.Dense(1, activation='linear')
    def call(self, inputs):
        x = self.dense1(inputs)
        x = self.dense2(x)
        output = self.dense3(x)
        return output

model = Mymodel()

model.compile(loss='mse',optimizer='sgd',metrics=['accuracy'])
model.fit(x, y, epochs=10)
```

- The Mixed National Institute of Standards and Technology (MNIST) database is a large database of handwritten digits.
- Source: Modified National Institute of Standards and Technology database (NIST).
- The MNIST database contains 60,000 training images and 10,000 testing images.
- 60k training images from American Census Bureau employees and 10k testing images from American high school students
- 28 x 28 grayscale images, 10 labels (0-9)

- How we can use MLP to classify digits?



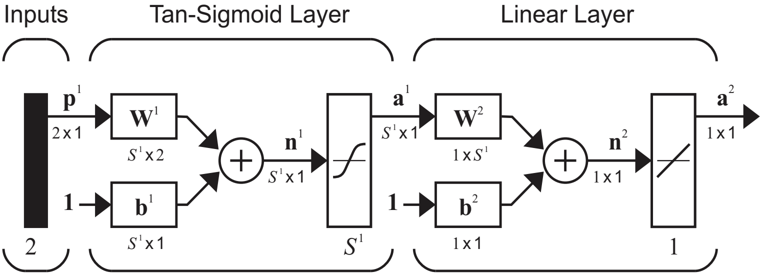
$$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1) \qquad \mathbf{a}^2 = \mathbf{f}^2(\mathbf{W}^2\mathbf{a}^1+\mathbf{b}^2) \qquad \mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{a}^2+\mathbf{b}^3)$$

$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1)+\mathbf{b}^2)+\mathbf{b}^3)$$

```
from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.optimizers import RMSprop

batch_size = 128
num_classes = 10
epochs = 20

(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.reshape(60000, 784)
x_test = x_test.reshape(10000, 784)
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
```

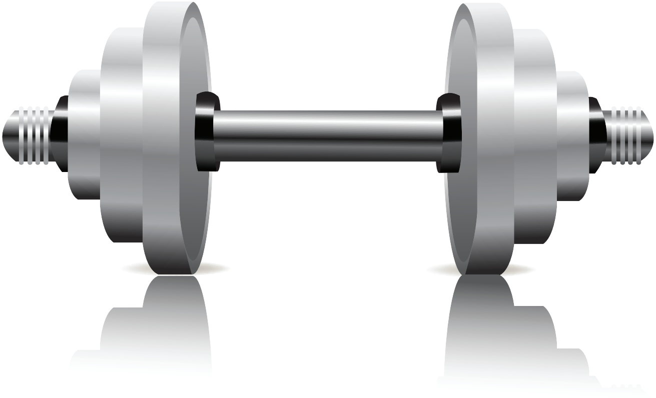
```
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))

model.summary()

model.compile(loss='categorical_crossentropy',
              optimizer=RMSprop(),
              metrics=['accuracy'])

history = model.fit(x_train, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

- How we can use two layer network to approximate underlying function?

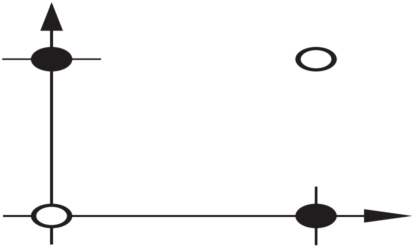


$$\mathbf{a}^1 = \mathbf{tansig}(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1) \qquad \mathbf{a}^2 = \mathbf{purelin}(\mathbf{W}^2\mathbf{a}^1+\mathbf{b}^2)$$

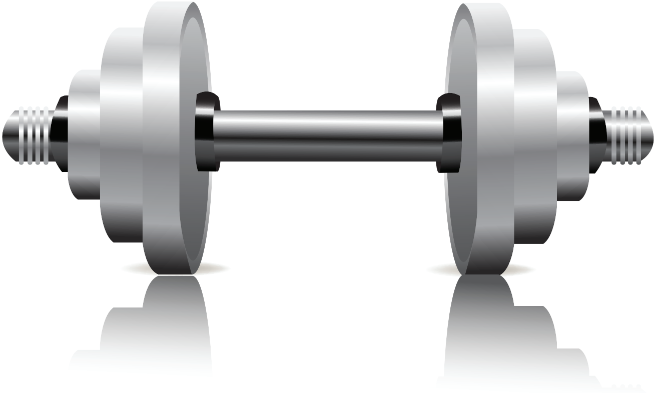
- Class-Ex-Keras.py



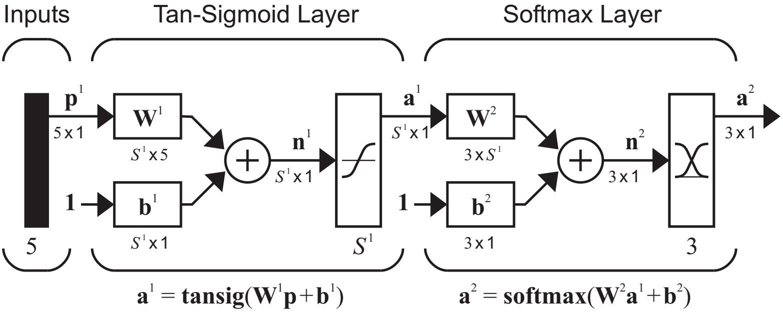
- How we can use MLP to classify XOR problem?



- Class-Ex-Keras.py



- How we can use MLP to classify images?



- Class-Ex-Keras.py