In [ ]:
```
tra
```

In [15]:
```
'''Trains a simple deep NN on the MNIST dataset.
Gets to 98% test accuracy after 5 epochs
'''
import pandas as pd
import numpy as np

from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential

from keras.layers import Dense
from keras.optimizers import RMSprop
from keras.optimizers import Adadelta

batch_size = 80
num_classes = 2
epochs = 5

cardio = pd.read_csv('cleveland.csv', header=None, index_col=False)

size = cardio.shape[0]

train_size = 290
test_size = 13

# train_data = cardio.iloc(0:290,0:14)
training_labels =np.asfarray(cardio.iloc[:train_size,13])
training_features = np.asfarray(cardio.iloc[:train_size,0:13])

test_labels = np.asfarray(cardio.iloc[train_size:size,13])
test_features = np.asfarray(cardio.iloc[train_size:size,0:13])

x_train = training_features
y_train = np.clip(training_labels,0,1)
x_test = test_features
y_test = np.clip(test_labels,0,1)


# the data, split between train and test sets
# (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')
```

```
# convert class vectors to binary class matrices
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=(13,)))
model.add(Dense(512, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
# model.add(Dense(num_classes, activation='relu'))



model.summary()

model.compile(loss='categorical_crossentropy',
#               optimizer=RMSprop(),
              optimizer=Adadelta(lr=0.05),
              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])
```

```
290 train samples
13 test samples
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_43 (Dense)             (None, 512)               7168
_________________________________________________________________
dense_44 (Dense)             (None, 512)               262656
_________________________________________________________________
dense_45 (Dense)             (None, 2)                 1026
=================================================================
Total params: 270,850
Trainable params: 270,850
Non-trainable params: 0
_________________________________________________________________
Train on 290 samples, validate on 13 samples
Epoch 1/5
290/290 [==============================] - 0s 2ms/step - loss: 1.5863
- acc: 0.5207 - val_loss: 0.8368 - val_acc: 0.7692
Epoch 2/5
290/290 [==============================] - 0s 74us/step - loss: 1.880
2 - acc: 0.4793 - val_loss: 0.5453 - val_acc: 0.7692
Epoch 3/5
290/290 [==============================] - 0s 74us/step - loss: 1.516
8 - acc: 0.5034 - val_loss: 0.4768 - val_acc: 0.7692
Epoch 4/5
290/290 [==============================] - 0s 71us/step - loss: 1.407
1 - acc: 0.5034 - val_loss: 0.6361 - val_acc: 0.7692
Epoch 5/5
290/290 [==============================] - 0s 72us/step - loss: 1.431
1 - acc: 0.5586 - val_loss: 0.4203 - val_acc: 0.9231
Test loss: 0.420269757509
Test accuracy: 0.923076927662
```