Sarah Riaz

1389209718

Homework 6

Question: 1

Training:

```
import numpy
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import LSTM
from keras.callbacks import ModelCheckpoint
from keras.utils import np_utils

# load ascii text and covert to lowercase
raw_text = open("bertrand.txt").read()
raw_text = raw_text.lower()

# create mapping of unique chars to integers
chars = sorted(list(set(raw_text)))
char_to_int = dict((c, i) for i, c in enumerate(chars))

# prepare the dataset of input to output pairs encoded as integers
seq_length = 100
dataX = []
dataY = []
for i in range(0, len(raw_text) - seq_length, 1):
   seq_in = raw_text[i:i + seq_length]
   seq_out = raw_text[i + seq_length]
   dataX.append([char_to_int[char] for char in seq_in])
   dataY.append(char_to_int[seq_out])
n_patterns = len(dataX)

# reshape X to be [samples, time steps, features] and normalize
x = numpy.reshape(dataX, (n_patterns, seq_length, 1)) / float(len(chars))

# one hot encode the output variable
y = np_utils.to_categorical(dataY)

# define the LSTM model
model = Sequential()
model.add(LSTM(256, input_shape=(x.shape[1], x.shape[2])))
model.add(Dropout(0.2))
model.add(Dense(y.shape[1], activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam')

# define the checkpoint
checkpoint = ModelCheckpoint("weights-improvement-{epoch:02d}-{loss:.2f}.hdf5",
save_best_only=True, verbose=1, mode='min', monitor='loss')
callbacks_list = [checkpoint]

# fit the model
model.fit(x, y, batch_size=128, epochs=30, callbacks=callbacks_list)
```

Testing:

```
import sys
import numpy
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import LSTM
from keras.callbacks import ModelCheckpoint
from keras.utils import np_utils

raw_text = open("bertrand.txt").read()
raw_text = raw_text.lower()

# create mapping of unique chars to integers and a reverse mapping
chars = sorted(list(set(raw_text)))
char_to_int = dict((c, i) for i, c in enumerate(chars))

# prepare the dataset of input to output pairs encoded as integers
seq_length = 100
test_x = []
test_y = []
for i in range(0, len(raw_text) - seq_length, 1):
   seq_in = raw_text[i:i + seq_length]
   seq_out = raw_text[i + seq_length]
   test_x.append([char_to_int[char] for char in seq_in])
   test_y.append(char_to_int[seq_out])

# reshape X to be [samples, time steps, features] and normalize
normalized_x = numpy.reshape(test_x, (len(test_x), seq_length, 1)) / float(len(chars))

# one hot encode the output variable
y = np_utils.to_categorical(test_y)

# define the LSTM model
model = Sequential()
model.add(LSTM(256, input_shape=(normalized_x.shape[1], normalized_x.shape[2])))
model.add(Dropout(0.2))
model.add(Dense(y.shape[1], activation='softmax'))

# load the network weights
model.load_weights("weights-improvement-29-1.9435.hdf5")
model.compile(loss='categorical_crossentropy', optimizer='adam')

start = numpy.random.randint(0, len(test_x) - 1)
for test_data in test_x:
   # generate characters
   int_to_char = dict((i, c) for i, c in enumerate(chars))
   for i in range(1000):
      x = numpy.reshape(test_data, (1, len(test_data), 1))
      x = x / float(n_vocab)
      prediction = model.predict(x, verbose=0)
      index = numpy.argmax(prediction)
      result = int_to_char[index]
      seq_in = [int_to_char[value] for value in test_data]
      sys.stdout.write(result)
      test_data.append(index)
      test_data = test_data[1:len(test_data)]
```