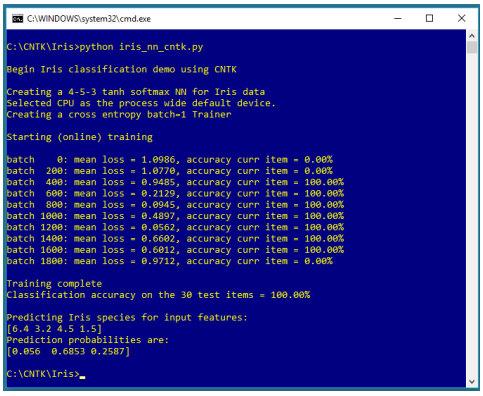
Predictive Analytics World / Deep Learning World
Exercises - CNTK

1. Locate the .whl file to install CNTK v2.4 (CPU only). Download the .whl file to your machine and install CNTK using pip.

2. Create training and test files of the Iris Dataset in a format that is friendly to CNTK data readers:

```
|attribs 5.1 3.5 1.4 0.2 |species 1 0 0
|attribs 4.9 3.0 1.4 0.2 |species 1 0 0
. . .
```

3. Write a CNTK program to create and train a 4-5-3 simple neural network. Run your program.



4. Which statement is most accurate?

a.) CNTK operates at a slightly higher level of abstraction than TensorFlow, but slightly lower than Keras.
b.) CNTK clearly operates at a higher level of abstraction than both TensorFlow and Keras.
c.) CNTK clearly operates at a lower level of abstraction than both TensorFlow and Keras.

5. Which statement is most accurate?

a.) A distinguishing characteristic of CNTK is the ability to create custom data readers relatively easily.
b.) A distinguishing characteristic of CNTK is the ability to create custom trainers relatively easily.
c.) A distinguishing characteristic of CNTK is the ability to create custom loss functions relatively easily.

6. Which statement is most accurate?

a.) A quirk of CNTK is that there is no categorical_cross_entropy loss function.
b.) A quirk of CNTK is that there is no mean_squared_error loss function
c.) A quirk of CNTK is that there is no softmax activation function.

```
# iris_nn_cntk.py
# CNTK 2.4 - Anaconda 4.1.1 (Python 3.5, NumPy 1.11.1)

# data resembles:
# |attribs 5.1 3.5 1.4 0.2 |species 1 0 0
# . . .

import numpy as np
import cntk as C

def main():
  print("\nBegin Iris classification demo using CNTK  \n")
  input_dim = 4; hidden_dim = 5; output_dim = 3
  train_file = ".\\Data\\iris_train_cntk.txt"
  test_file = ".\\Data\\iris_test_cntk.txt"

  train_x = np.loadtxt(train_file, delimiter=" ", usecols=[1,2,3,4], dtype=np.float32)
  train_y = np.loadtxt(train_file, delimiter=" ", usecols=[6,7,8], dtype=np.float32)
  test_x = np.loadtxt(test_file, delimiter=" ", usecols=[1,2,3,4], dtype=np.float32)
  test_y = np.loadtxt(test_file, delimiter=" ", usecols=[6,7,8], dtype=np.float32)

  X = C.ops.input_variable(input_dim, np.float32)
  Y = C.ops.input_variable(output_dim, np.float32)

  print("Creating a 4-5-3 tanh softmax NN for Iris data ")
  with C.layers.default_options(init=C.initializer.uniform(scale=0.01,
    seed=1)):
    h_layer = C.layers.Dense(hidden_dim, activation=C.ops.tanh, name='hidLayer')(X)
    o_layer = C.layers.Dense(output_dim, activation=None, name='outLayer')(h_layer)
  nnet = o_layer
  model = C.ops.softmax(nnet)

  print("Creating a cross entropy batch=1 Trainer \n")
  tr_loss = C.cross_entropy_with_softmax(nnet, Y)
  tr_clas = C.classification_error(nnet, Y)

  learn_rate = 0.01
  learner = C.sgd(nnet.parameters, learn_rate)
  trainer = C.Trainer(nnet, (tr_loss, tr_clas), [learner])

  max_iter = 2000  # maximum training iterations
  np.random.seed(1); N = len(train_x)

  print("Starting (online) training \n")
  for i in range(0, int(max_iter)):
    rnd_row = np.random.choice(N,1)
    trainer.train_minibatch({X:train_x[rnd_row], Y:train_y[rnd_row]})
    if i % 200 == 0:
      mcee = trainer.previous_minibatch_loss_average
      macc = (1.0 - trainer.previous_minibatch_evaluation_average) * 100
      print("batch %4d: mean loss = %0.4f, mean accuracy = %0.2f%% " % (i,mcee, macc))
  print("\nTraining complete")

  acc = (1.0 - trainer.test_minibatch({X:test_x, Y:test_y })) * 100
  print("Classification accuracy on the 30 test items = %0.2f%%" % acc)

  # mp = ".\\Models\\iris_nn.model"  # path to model file
  # model.save(mp, format=C.ModelFormat.CNTKv2)  # or ONNX

  np.set_printoptions(precision = 4)
  unknown = np.array([[6.4, 3.2, 4.5, 1.5]], dtype=np.float32)
  print("\nPredicting Iris species for input features: ");  print(unknown[0])

  pred_prob = model.eval(unknown)  # simple form works
  print("Prediction probabilities are: ")
  print(pred_prob[0])

if __name__ == "__main__":
  main()
```