

Homework 4: Neural Networks and Reinforcement Learning

EE 459/559 2020 Spring

Discussion is encouraged, but homework and codes shall be written on your own

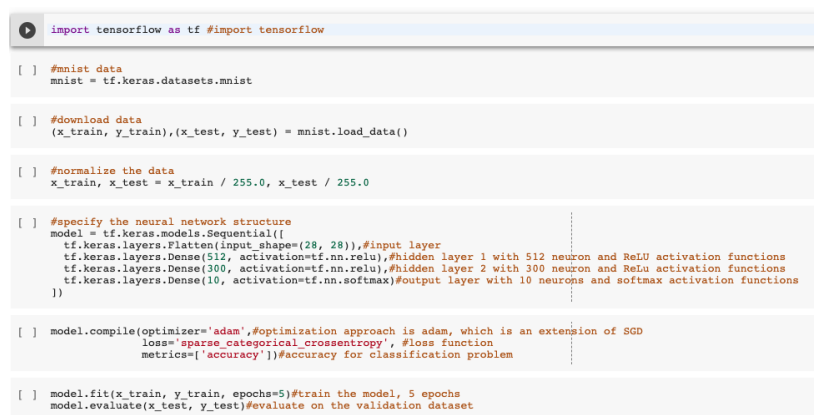
Due: 11:59pm, April 28 2019

Late submissions will not accepted

1 Neural Networks (40 points)

Understand the following example code for classification on MNIST dataset (Fig. 1).

The help doc: https://www.tensorflow.org/api_docs/python/tf/keras



```
import tensorflow as tf #import tensorflow

[ ] #mnist data
mnist = tf.keras.datasets.mnist

[ ] #download data
(x_train, y_train), (x_test, y_test) = mnist.load_data()

[ ] #normalize the data
x_train, x_test = x_train / 255.0, x_test / 255.0

[ ] #specify the neural network structure
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)), #input layer
    tf.keras.layers.Dense(512, activation=tf.nn.relu), #hidden layer 1 with 512 neuron and ReLU activation functions
    tf.keras.layers.Dense(300, activation=tf.nn.relu), #hidden layer 2 with 300 neuron and ReLU activation functions
    tf.keras.layers.Dense(10, activation=tf.nn.softmax) #output layer with 10 neurons and softmax activation functions
])

[ ] model.compile(optimizer='adam', #optimization approach is adam, which is an extension of SGD
    loss='sparse_categorical_crossentropy', #loss function
    metrics=['accuracy']) #accuracy for classification problem

[ ] model.fit(x_train, y_train, epochs=5) #train the model, 5 epochs
model.evaluate(x_test, y_test) #evaluate on the validation dataset
```

Figure 1: Example code for classification on MNIST dataset.

Complete the steps as listed below.

1. Import the fashion mnist dataset:

```
fashion_mnist = tf.keras.datasets.fashion_mnist
```

2. load the training and test data from fashion_mnist dataset:

```
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```

3. training data size and dimension:

```
train_images.shape
```

4. Process the data

```
train_images = train_images / 255.0
```

```
test_images = test_images / 255.0
```

5. There are in total 10 classes. Set class name:

```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker',  
'Bag', 'Ankle boot']
```

6. Image show: (see Fig. 2)

7. Follow the example in Fig. 1, implement a neural network with *two* hidden layers, the number of neurons in hidden layer 1 is 128, the number of neurons in hidden layer 2 is 64. In the hidden layers, the activation functions are ReLu.

The number of neurons in the output layer is 10 (because we have ten classes in this problem).

The activation function in the output layer is softmax.

8. Train the neural network

9. Evaluate the performance on the test dataset.

```
plt.figure(figsize=(10,10))  
for i in range(25):  
    plt.subplot(5,5,i+1)  
    plt.xticks([])  
    plt.yticks([])  
    plt.grid(False)  
    plt.imshow(train_images[i], cmap=plt.cm.binary)  
    plt.xlabel(class_names[train_labels[i]])  
plt.show()
```

Figure 2: Image show

In your submission, please save the output of each step in your code. Submit the code together with the output.

2 Reinforcement Learning (60 points)

Consider the following problem. The agent is in a 4X4 maze. Its initial position is the top-left corner. The goal of the agent is to escape the maze by moving to the bottom right corner. The agent has no preliminary knowledge of the environment or the goal to achieve. At each time step, it can perform 4 possible actions: up, down, left or right (*unless it is on the edge of the maze*). *Once the agent is in the bottom-right corner, then no action can be taken and the game ends.* After performing

an action, it receives a positive or a negative reward indicated by a score in each cell. There are 16 possible states, shown by a red number.

Set the discount factor $\gamma = 0.9$ and step size $\alpha = 0.1$. For each problem below, run for 5×10^5 iterations.

| | | | |
|----------|----------|----------|-----------|
| 1 0 | 2 -1 | 3 -5 | 4 +3 |
| 5 -3 | 6 -2 | 7 -6 | 8 -2 |
| 9 -2 | 10 -4 | 11 -8 | 12 -10 |
| 13 +2 | 14 -3 | 15 -9 | 16 +20 |

Figure 3: 4×4 grid-maze

1. Consider a randomized policy π , where the probabilities of taking all available actions are the same, e.g., in grid 1, the probabilities of taking right and down are all $1/2$, and in grid 6, the probabilities of taking up, down, left and right are all $1/4$. Implement the temporal difference (TD) algorithm and evaluate the value function of this randomized policy π .

Hint: This problem has a finite-horizon, i.e., the game will end once the agent reaches state 16. Once the agent reaches the terminate state 16, then restart the algorithm from state 1. A modified TD algorithm shall be followed, see Algorithm 1.

2. The Q function table is composed of 16 rows (one for each state) and 4 columns (one for each action) and its usually initialised with zeros. Adapt the Q-learning algorithm to this finite-

horizon setting as in problem 1. Implement the Q-learning algorithm using the randomized policy described above as the behavior policy. Derive a policy which is greedy with respect to the obtained Q-function, and compute its value function in state 1. Note that for a greedy policy, it is actually deterministic, and its value function can be directly computed.

Result: $V(s)$

```

Initialization; for  $i = 1, \dots, 5 \times 10^5$  do
    if  $s$  is not the absorbing state (state 16) then
        Get action  $a$  according to  $\pi$ .
        Observe reward  $R(s, a)$  and next state  $s'$ .
        Update:  $V(s) \leftarrow V(s) + \alpha(R(s, a) + \gamma V(s') - V(s))$ .
         $s \leftarrow s'$ ;
    end
    else
         $s = 1$ ; %restart from state 1.
    end
end

```

Algorithm 1: TD algorithm for finite-horizon problem.

3. Adapt the SARSA algorithm to this finite-horizon problem as in Algorithm 1, and implement the adapted SARSA algorithm using ϵ -greedy approach, where $\epsilon = 0.05, 0.1, 0.2, 0.5$. Derive policies which are greedy with respect to the obtained Q-functions for different ϵ 's, and compute their value function in state 1.
4. Compare the obtained policies from task 2 and task 3 in terms of the cumulative discounted rewards starting from state 1. Discuss the performance of the SARSA algorithm for different values of ϵ .

Hint: The value of ϵ measures the tradeoff between exploitation and exploration.