HM1_section_1

October 2, 2022

[1]:
```
import lr_utils as U
from matplotlib import pyplot as plt
import numpy as np
import lr_main as M
```

0.0.1 Load data using load_dataset function in lr_utils

[2]:
```
train_data_x, train_data_y, test_data_x, test_data_y = U.load_dataset()
```

0.0.2 Report the shape of the above four datasets

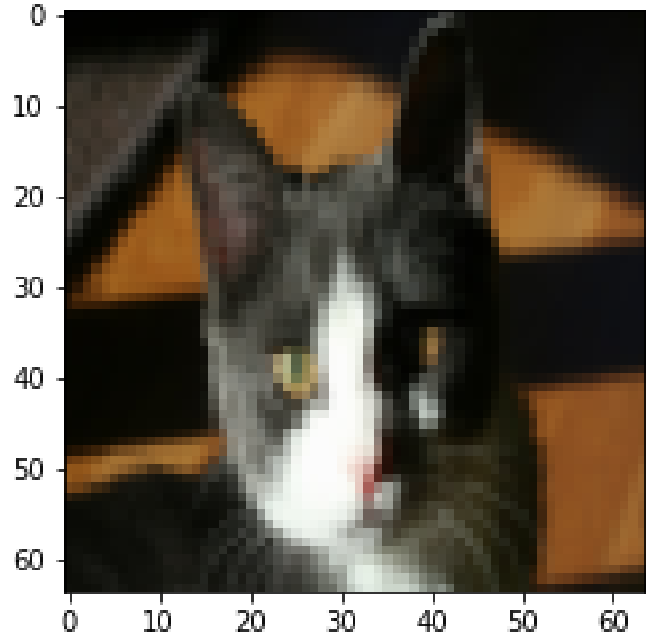
[3]:
```
print(f'The size of train_data_x is: {train_data_x.shape}')
print(f'The size of train_data_y is: {train_data_y.shape}')
print(f'The size of test_data_x is: {test_data_x.shape}')
print(f'The size of test_data_y is: {test_data_y.shape}')
```

```
The size of train_data_x is: (205, 64, 64, 3)
The size of train_data_y is: (1, 205)
The size of test_data_x is: (48, 64, 64, 3)
The size of test_data_y is: (1, 48)
```

It is clear that the image size is 64 x 64 pixels with the last 3 dimensions for color

0.0.3 Plot the image of the 20th data sample in training data

[4]:
```
data = train_data_x[19]
plt.imshow(data, interpolation='nearest')
plt.show()
```



It is a black cat.

0.0.4 Report the shape of train_data_x/test_data_x after reshaping

[5]:
```
train_data_x = np.reshape(train_data_x, (train_data_x.shape[0], train_data_x.
 ↪shape[1] * train_data_x.shape[2] * train_data_x.shape[3]))
test_data_x =  np.reshape(test_data_x, (test_data_x.shape[0], test_data_x.
 ↪shape[1] * test_data_x.shape[2] * test_data_x.shape[3]))
print(f'The shape of train_data_x after reshaping is: {train_data_x.shape}')
print(f'The shape of test_data_x after reshaping is: {test_data_x.shape}')
# normalize data
train_data_x = train_data_x / 255.0
test_data_x = test_data_x / 255.0
```

```
The shape of train_data_x after reshaping is: (205, 12288)
The shape of test_data_x after reshaping is: (48, 12288)
```

0.0.5 Run the model and report the results

[6]:
```
# run model with iter_number = 5000
iter_number = 5000
alpha = 0.01
result = M.model(train_data_x, train_data_y, test_data_x, test_data_y,␣
 ↪iter_number, alpha, print_loss = False)
```

```
Predict accurary of the test data after 5000 iterations is: 0.6458333333333334
Predict accurary of the train data after 5000 iterations with learning rate 0.01
is: 1.0
```

After 5000 iterations, the predict accuracy of test data is 0.65, whereas the accurary of the train data is about 1

0.0.6 Next we plot the learning curves with diferent learning rates

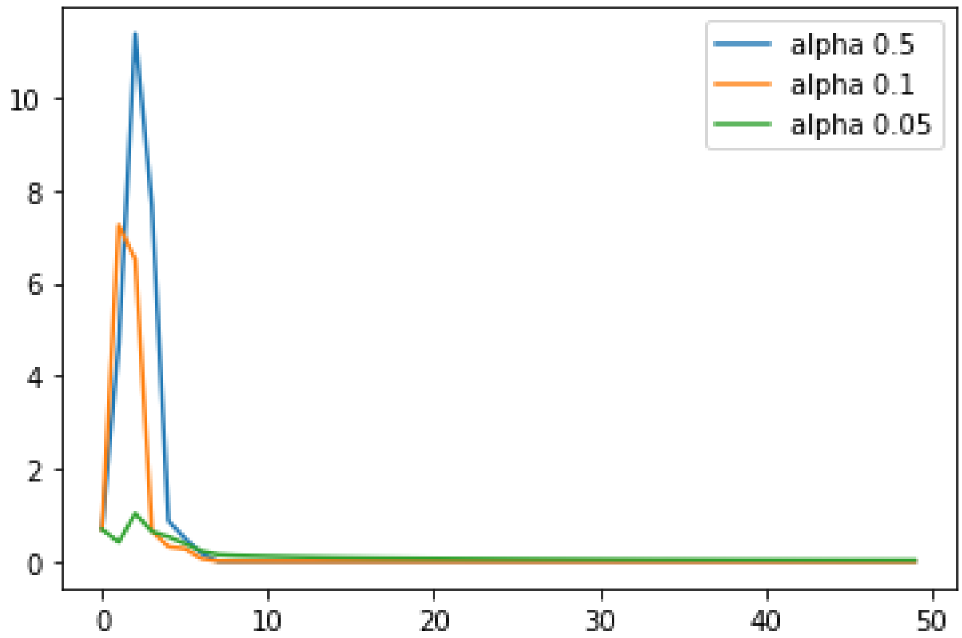
[7]:
```
loss_with_alpha = []
for alpha in [0.5, 0.1, 0.01]:
    result = M.model(train_data_x, train_data_y, test_data_x, test_data_y,␣
 ↪iter_number, alpha, print_loss = False)
    loss_with_alpha.append([x for i, x in enumerate(result['loss_all']) if i %␣
 ↪100 == 0])
```

```
/Users/zach/Downloads/cosi-165b/AS-1/code/lr_utils.py:18: RuntimeWarning:
overflow encountered in exp
  y = 1 / (1 + np.exp(- x))
```

```
Predict accurary of the test data after 5000 iterations is: 0.6875
Predict accurary of the train data after 5000 iterations with learning rate 0.5
is: 1.0
Predict accurary of the test data after 5000 iterations is: 0.6458333333333334
Predict accurary of the train data after 5000 iterations with learning rate 0.1
is: 1.0
Predict accurary of the test data after 5000 iterations is: 0.6458333333333334
Predict accurary of the train data after 5000 iterations with learning rate 0.01
is: 1.0
```

[9]:
```
data = np.array(loss_with_alpha).T
plt.plot(data)
plt.legend(["alpha 0.5", "alpha 0.1", "alpha 0.05"])
```

[9]: <matplotlib.legend.Legend at 0x7f7968bb4dc0>



0.0.7 Next we are going to train a deep network with L-layer L = 5 to solve the same problem

[2]:
```
import dnn_main as DM
```

[3]:
```
# run model with iter_number = 1000
iter_number = 1000
alpha = 0.05
layers_dims = [12288, 32, 16, 1]
train_data_x, train_data_y, test_data_x, test_data_y = U.load_dataset()
train_data_x = np.reshape(train_data_x, (train_data_x.shape[0], train_data_x.
 ↪shape[1] * train_data_x.shape[2] * train_data_x.shape[3])).T
test_data_x =  np.reshape(test_data_x, (test_data_x.shape[0], test_data_x.
 ↪shape[1] * test_data_x.shape[2] * test_data_x.shape[3])).T
train_data_x = train_data_x / 255.0
test_data_x = test_data_x / 255.0
```

[4]:
```
result = DM.model(train_data_x, train_data_y, test_data_x, test_data_y,␣
 ↪layers_dims, alpha, iter_number, False)
```

```
Predict accurary of the test data after 1000 iterations is: 0.75
Predict accurary of the train data after 1000 iterations with learning rate 0.05
is: 1.0
```

After 1000 iterations, the predict accuracy of test data is 0.75, whereas the accurary of the train data is about 1

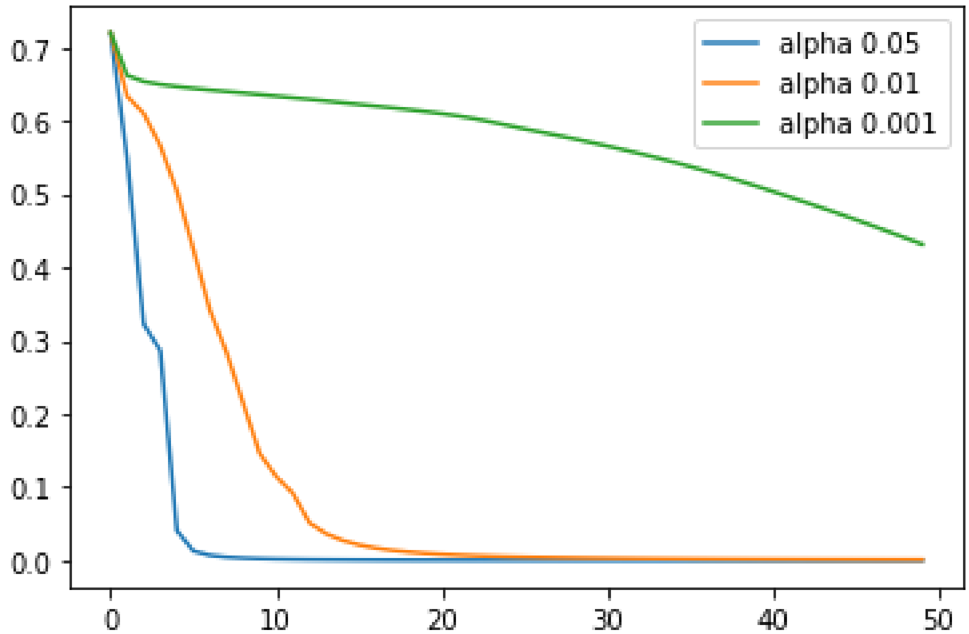
0.0.8 Next we plot the learning curves with diferent learning rates

[6]:
```
loss_with_alpha = []
iter_number = 5000
for alpha in [0.05, 0.01, 0.001]:
    result = DM.model(train_data_x, train_data_y, test_data_x, test_data_y,␣
 ↪layers_dims, alpha, iter_number, print_loss = False)
    loss_with_alpha.append([x for i, x in enumerate(result['loss_all']) if i %␣
 ↪100 == 0])
```

```
Predict accurary of the test data after 5000 iterations is: 0.75
Predict accurary of the train data after 5000 iterations with learning rate 0.05
is: 1.0
Predict accurary of the test data after 5000 iterations is: 0.6875
Predict accurary of the train data after 5000 iterations with learning rate 0.01
is: 1.0
Predict accurary of the test data after 5000 iterations is: 0.5833333333333334
Predict accurary of the train data after 5000 iterations with learning rate
0.001 is: 0.8341463414634146
```

[7]:
```
data = np.array(loss_with_alpha).T
plt.plot(data)
plt.legend(["alpha 0.05", "alpha 0.01", "alpha 0.001"])
```

[7]: <matplotlib.legend.Legend at 0x7fbd900f3f70>



Looks like what we expected but learning rate 0.001 seems not converge due to not enough iterations