task2

November 29, 2023

[ ]:
```
import numpy as np
from matplotlib import pyplot as plt
```

[ ]:
```
from keras.datasets import mnist
```

[ ]:
```
(Xtr, Ltr), (X_test, L_test)=mnist.load_data()
```

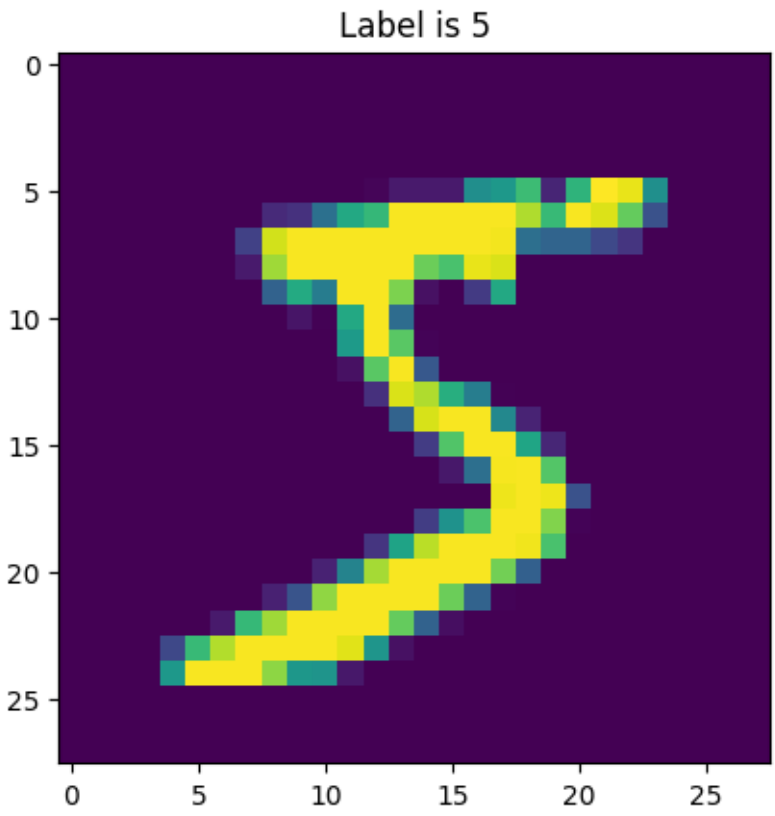
[ ]:
```
Xtr.shape
```

[ ]:
```
(60000, 28, 28)
```

[ ]:
```
Image=Xtr[0,:,:]
Label=Ltr[0]


plt.title('Label is {Label}'.format(Label=Label))
plt.imshow(Image)

plt.show()
plt.close()
```

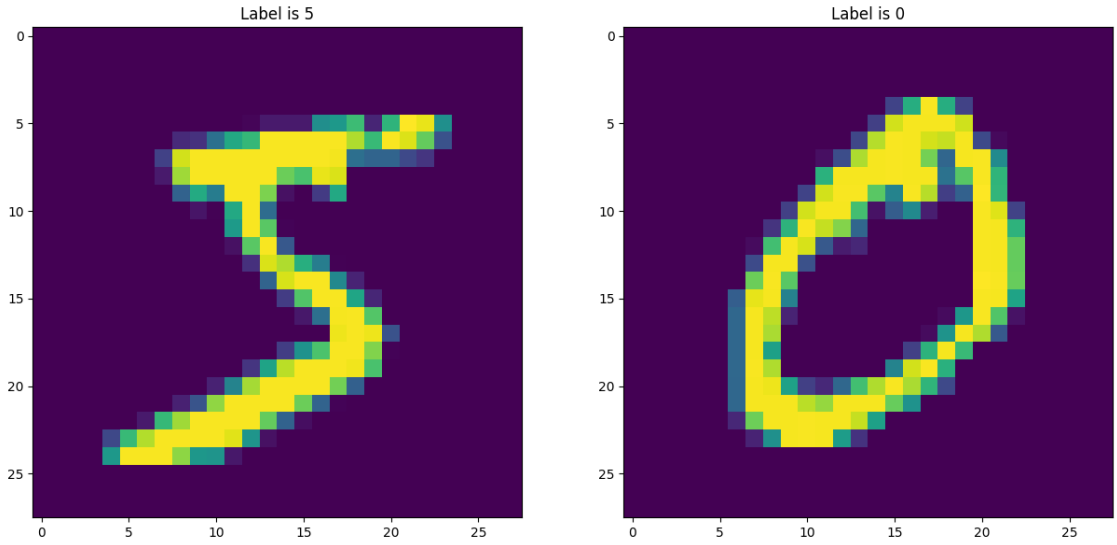


[ ]:
```
fig, ax=plt.subplots(nrows=1, ncols=2, figsize=(15,15))
ax0=plt.subplot(2,2,1)
ax1=plt.subplot(2,2,2)

Image=Xtr[0,:,:]
Label=Ltr[0]
Image1=Xtr[1,:,:]
Label1=Ltr[1]


ax0.set_title('Label is {Label}'.format(Label=Label))
ax0.imshow(Image)
ax1.set_title('Label is {Label}'.format(Label=Label1))
ax1.imshow(Image1)


plt.show()
plt.close()
```



[ ]:
```
28*28
```

[ ]: 784

[ ]:
```
#Traing phase
num_sample=500
Tr_set=Xtr[:num_sample,:,:]
Ltr_set=Ltr[:num_sample]

Tr_set=Tr_set.reshape(num_sample,Tr_set.shape[1]*Tr_set.shape[2]).astype(int)

#Tr_set=Tr_set.reshape(num_sample,Tr_set.shape[1]*Tr_set.shape[2]).astype()
Tr_set.shape
```

[ ]: (500, 784)

[ ]:
```
def predict_L1(X):
    num_test=X.shape[0]
    Lpred=np.zeros(num_test, dtype=Ltr_set.dtype)

    for i in range(num_test):
        distances=np.sum(np.abs(Tr_set-X[i,:]),axis=1)

        min_index= np.argmin(distances)
        Lpred[i]=Ltr_set[min_index]
    return Lpred
```

[ ]:
```
def predict_L2(X):
    num_test=X.shape[0]
```

```
    Lpred=np.zeros(num_test, dtype=Ltr_set.dtype)
    for i in range(num_test):
        distances=np.sqrt(np.sum(np.abs(Tr_set-X[i,:])**2, axis=1))
        min_index= np.argmin(distances)
        Lpred[i]=Ltr_set[min_index]
    return Lpred
```

[ ]:
```
def L2_norm(X, i):
    return np.sqrt(np.sum(np.abs(Tr_set-X[i,:])**2, axis=1))
```

[ ]:
```
def vote(labels):
    counts = np.bincount(labels)
    most_frequent = np.argmax(counts)
    return most_frequent
```

[ ]:
```
def k_nearest_neighbor_L2(X, k):
    num_test=X.shape[0]
    Lpred=np.zeros(num_test, dtype=Ltr_set.dtype)
    for i in range(4, num_test):
        distances=L2_norm(X, i)
        min_indicies= np.argpartition(distances, k) # Returns indicies of␣
 ↪lowest values
        labels = Ltr_set[min_indicies[:k]] # The labels corresponding to the k␣
 ↪lowest indicies
        majority_vote = vote(labels)
        Lpred[i] = majority_vote
    return Lpred
```

[ ]:
```
def k_fold(dataset, k = 3):
    return np.array_split(dataset, k)
```

[ ]:
```
def cross_validation(func, validation_set, test_set, upperbound, steps = 2):
    best_k = 0
    best_accuracy = 0
    for k in range(1, upperbound+1, steps):
        labels_predicted = func(validation_set, k)
        accuracy = np.mean(labels_predicted==test_set)
        if accuracy > best_accuracy:
            best_accuracy = accuracy
            best_k = k
        print("Cross validation - k:", k, " with accuracy:", accuracy)
    return best_k, best_accuracy
```

[ ]:
```
Test_images=X_test.reshape(X_test.shape[0],X_test.shape[1]* X_test.shape[2])

test_images = k_fold(Test_images, 3)
test_labels = k_fold(L_test, 3)
```

```
k, acc = cross_validation(k_nearest_neighbor_L2, test_images[-1],␣
 ↪test_labels[-1], 11, 2)
print("Best k:", k, " with accuracy:", acc)
Labels_predicted=k_nearest_neighbor_L2(Test_images, 1)
```

```
Cross validation - k: 1  with accuracy: 0.885988598859886
Cross validation - k: 3  with accuracy: 0.8724872487248725
Cross validation - k: 5  with accuracy: 0.8706870687068707
Cross validation - k: 7  with accuracy: 0.864986498649865
Cross validation - k: 9  with accuracy: 0.8571857185718572
Cross validation - k: 11  with accuracy: 0.8451845184518452
Best k: 1  with accuracy: 0.885988598859886
```

[ ]:
```
print("Accuracy for k =", 1, ":", np.mean(Labels_predicted==L_test))
```

```
Accuracy for k = 1 : 0.8291
```