In this project, we will explore the MNIST handwritten digits dataset, available at: http://yann.lecun.com/exdb/mnist/

The dataset included in this project was originally downloaded using TensorFlow: https://www.tensorflow.org/datasets/catalog/mnist

[1]:

# install numpy and scikit-learn, if needed

import numpy as np  
from sklearn.linear\_model import LogisticRegression  
# I included a helper module for the project, for loading MNIST

from mnist import Mnist

[2]:

# load the training dataset (used to train/learn the logistic regression model)

dataset = Mnist("train")  
# load the testing dataset (used to test/evaluate the logistic regression model) test\_dataset = Mnist("test")

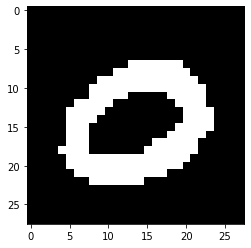
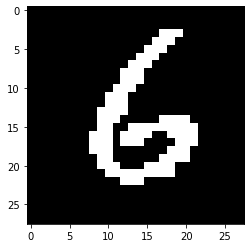
MNIST is a dataset of handwritten image.  
The dataset contains 55,000 images, and each image has 28x28 = 784 pixels.  
For the purposes of this project, we are dealing with pure black and white images (not grayscale). Below, we plot three example digits from the dataset.

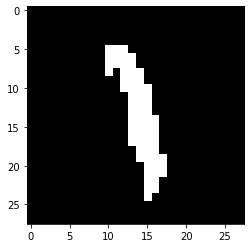
[3]:

# plot images with index 100, index 101 and index 102

for image in dataset.images()[1000:1003]:

dataset.plot\_image(image)





We will consider classification between two different digits at a time (this is simpler, and will help us try to understand how logistic regression works better). First, we shall extract all images of ones and twos.

[4]:

# pick two digits to compare  
# the first class is the negative label, the second class is the positive label

class\_neg,class\_pos = 1,7  
# get all images/labels of the above two digits  
images,labels = dataset.image\_labels\_of\_digits([class\_neg,class\_pos])  
# convert the labels from (1/7) to (0/1) (or True/False), for logistic regression

labels = labels == class\_pos

# get all images/labels of the two digits from the test dataset

test\_images,test\_labels = test\_dataset. ,→image\_labels\_of\_digits([class\_neg,class\_pos])

test\_labels = test\_labels == class\_pos

We will initialize a logistic regression model, and learn its parameters from the images and the labels, using the call to “fit”. Below we print the resulting accuracy of the model, based on the dataset that we used to train it.

[5]:

# learn the classifier from data

classifier = LogisticRegression(max\_iter=1000)  
classifier.fit(images,labels)  
# use the classifier to predict the labels of the training set, and then compute␣

,→the accuracy

predictions = classifier.predict(test\_images)

correct = sum(predictions == test\_labels)

print("(%d vs %d) accuracy %.4f%% (%d/%d)" % (class\_neg,class\_pos,

100\*correct/len(test\_images),

correct,len(test\_images)))

(1 vs 7) accuracy 99.3990% (2150/2163)

Let **x** be an image, represented as a vector of 784 input pixels, where *xi* denotes the *i*-th pixel. Our logistic regression model has a weight vector **w** composed of weights *wi*, one for each pixel

*xi*. We also have a bias *b*; these weights are learned from a dataset.

Once we have learned the weights, our logistic regression model predicts the probability that a new input image **x** has a positive label or a negative label. This probability is computed using the following formula:

Pr(positive) = *σ*(*w*1*x*1 +···+*wnxn* +*b*) = *σ*(**w***T***x**+*b*)

*where*

*σ*(*x*) = 1 /1 + exp{−*x*}

is the logistic (or sigmoid) function. Remember that the logistic function maps a number from negative infinity to positive infinity. So the more positive that *x* is, the closer to one that *σ*(*x*) is, and the more negative that *x* is, the closer to zero that *σ*(*x*) is.

[6]:

sigmoid = lambda x: 1.0/(1.0+np.exp(-x)) # the logistic function

w = classifier.coef\_.T # weights that we learned from data  
b = classifier.intercept\_ # bias that we learned from data

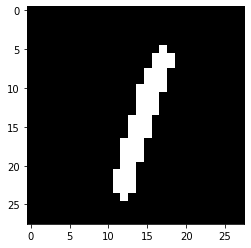
[7]:

# below, we pick an image, and print out the probability of being class\_a vs␣ ,→class\_b, as computed by:

#  
# (a) the model  
# (b) manually by ourselves, using the weights and bias that we fetched above #  
# to verify that the computations are the same (this is just a sanity check).  
x = test\_images[2]  
dataset.plot\_image(x)  
pr\_classifier = classifier.predict\_proba([x])[0]  
pr\_manual = sigmoid( w.T@x + b )  
print("probability in classifier: Pr(digit-%d) = %.6f, Pr(digit-%d) = %.6f" %␣

,→(class\_neg,pr\_classifier[0],class\_pos,pr\_classifier[1]))  
print("probability we calculated: Pr(digit-%d) = %.6f, Pr(digit-%d) = %.6f" %␣

,→(class\_neg,1.0-pr\_manual,class\_pos,pr\_manual))



probability in classifier: Pr(digit-1) = 0.997212, Pr(digit-7) = 0.002788

probability we calculated: Pr(digit-1) = 0.997212, Pr(digit-7) = 0.002788

Below, we visualize the weights **w** of our linear regression model.

Remember that the positive weights in the logistic regression will push the model right (towards a positive label), and that the negative weights will push the model left (towards a negative label). Weights that are close to zero correspond to inputs that do not affect the output much.

Below, the red pixels correspond to positive weights, and the blue pixels correspond to negative weights. White pixels have a value close to zero.

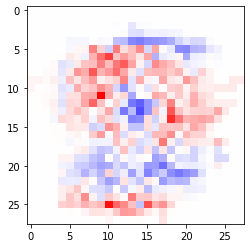
[8]:

# plot the probability of a pixel being white, given the image was of a zero

limit = max(abs(w))

weight\_image = w

dataset.plot\_image(weight\_image,cmap="bwr",limit=limit)



Next, we look for the image that had the highest probability of being labeled positively (as a digit seven), and then overlay it on the above image, to see why it was classified as a positive label.

[9]:

pr = classifier.predict\_proba(test\_images)

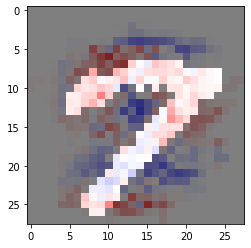
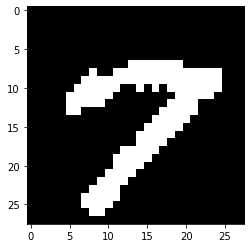
diff\_pr = pr[:,1]-pr[:,0]

max\_index = np.argmax(diff\_pr)

image,label = test\_images[max\_index],test\_labels[max\_index]

dataset.plot\_image(image)

dataset.plot\_image(weight\_image,cmap="bwr",limit=limit,second=image)



When we overlay the image of the two onto the red/blue image. Note that since our inputs (the pixels) are either black (with value 0) or white (with value 1). Hence, the black pixels contribute zero weight in the logistic regression model (in the summation, inside the logistic function). Thus, the aggregate sum of the weights of the white pixels determine the output of the logistic regression model.

In this case, we see that the white pixels of the digit seven hit a lot more red pixels than they do blue pixels.

We can do the same for the image that had the highest probability of being a negative label (a digit one), next.

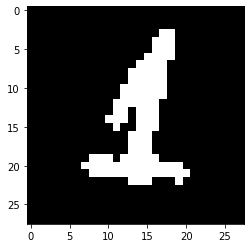
[10]:

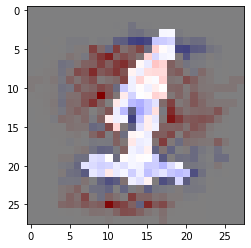
min\_index = np.argmin(diff\_pr)

image,label = test\_images[min\_index],test\_labels[min\_index]

dataset.plot\_image(image)

dataset.plot\_image(weight\_image,cmap="bwr",limit=limit,second=image)





In this case, the white pixels of the digit one hit more blue pixels than red pixels.

In general, for logistic regression, the more red pixels an image hits the more likely the image will be labeled positively (as a digit seven, in this case). The more blue pixels an image hits, the more likely the image will be labeled negatively (as a digit one, in this case).

Answer the following questions about this logistic regression model learned from the given dataset.

1. Some images that were labeled as a digit seven in the training set were misclassified as being a digit one by the classifier. Find the image of a seven that had the highest probability of being a one (include a picture). Explain why the digit was classified as a one instead of as a seven (include a visualization, like we did above).
2. Some images that were labeled as a digit one in the training set were misclassified as being a digit seven by the classifier. Find the image of a one that had the highest probability of being a seven (include a picture). Explain why the digit was classified as a seven instead of as a one (include a visualization, like we did above).
3. Train another logistic regression model using any other pair of digits *i* and *j*, besides 1 and 7. Provide a single example of a digit labeled *i* being misclassified as *j*, with a visualization of why it was misclassified (like we did above). (Note that this is only possible if some image of a digit was misclassified, i.e., if it did not get 100% accuracy.)

**Turn in** a pdf containing the answers to the above questions onto the course website under *Assignments* and *Homework 3.* Assignments are due Monday, October 2 by 11:59pm. Please start early in case you encounter any unexpected difficulties.