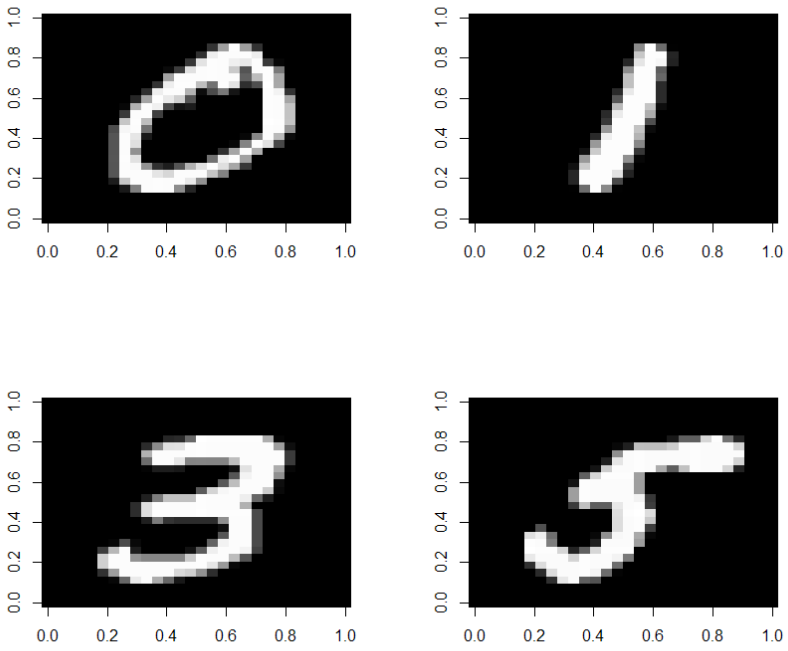
Matthew L. Miller
mmiller319

1. Data Preprocessing

After preprocessing the mnist_test and mnist_train data, the following four images were obtained from the training data. They consist of the first and last images from the train_data_0_1 dataset and the train_data_3_5 dataset. The first image of the two datasets are a 0 and 3, respectively, while the last two images are a 1 and 5. The fact that the 5 displays correctly shows that the images have the proper orientation and are not flipped, rotated, or mirrored.



2. Theory

a.) The Loss function used in Logistic Regression can be found in Dr. Lebanon's "Introduction to Logistic Regression" reading as equation 8:

$$\hat{\theta}_{MLE} = arg\,max_{\theta} \sum_{i=1}^{n} \log \frac{1}{1+\exp\left(y^{(i)}\langle\theta,x^{(i)}\rangle\right)} =$$

$$arg\,max_{\theta} \sum_{i=1}^{n} -\log\left(1+\exp\left(y^{(i)}\langle\theta,x^{(i)}\rangle\right)\right) =$$

$$arg\min_{\theta} \sum_{i=1}^{n} log\left(1+exp\left(y^{(i)}\langle\theta,x^{(i)}\rangle\right)\right)$$

b.) Using the chain rule, the equation above can be derived to find the gradient of the loss function, $\frac{dL(\theta)}{d\theta}$:

$$\frac{dL(\theta)}{d\theta} log\left(1+exp\left(y^{(i)}\langle\theta,x^{(i)}\rangle\right)\right) =$$

$$\left(\frac{1}{1+exp\left(-y^{(i)}\langle\theta,x^{(i)}\rangle\right)}\right) \frac{d}{d\theta}\left(1+exp\left(y^{(i)}\langle\theta,x^{(i)}\rangle\right)\right) =$$

$$\frac{exp\left(y^{(i)}\langle\theta,x^{(i)}\rangle\right)}{1+exp\left(-y^{(i)}\langle\theta,x^{(i)}\rangle\right)} \frac{d}{d\theta}\left(y\langle\theta,x^{(i)}\rangle\right) = \frac{exp\left(y^{(i)}\langle\theta,x^{(i)}\rangle\right)}{1+exp\left(-y^{(i)}\langle\theta,x^{(i)}\rangle\right)} \left(y^{(i)}\, x^{(i)}\right) =$$

$$\left(\frac{exp\left(y^{(i)}\langle\theta,x^{(i)}\rangle\right)\ \left(y^{(i)}\ x^{(i)}\right)}{1+exp(-y\,\langle\theta,x\rangle)}\right)$$

c.) From this derivation, the Stochastic Gradient Descent (SGD) update rule can be expressed with a sample ⟨x(i),y(i)⟩ with a step size of α:

$$\theta_j = \theta_j - \alpha \sum_{i=1}^{n} \frac{exp\left(y^{(i)}\langle\theta,x^{(i)}\rangle\right)\left(y^{(i)}\ x^{(i)}\right)}{1+exp(-y\langle\theta,x\rangle)}$$

d.) Pseudocode for implementing Logistic Regression and the SGD:

```
input: training dataset, step value alpha, number of epochs
output: training model using Logistic Regression and Stochastic Gradient Descent (SGD)

function Main(trainingData, alpha, numEpochs){
        return SGD(trainingData, alpha, numEpochs)
}

# Perform Stochastic Gradient Descent
function SGD(trainingData, alpha, numEpochs){

        #Create new array coefs with length equal to number of samples in trainingData
        coefficents = array initialized to zero with length = number of coefficients

        # Loop through each epoch, then through each of the sample in training data
        for(epoch in numEpochs){

                sumError = 0;                   # Stores sum of error squared

                for(sample in trainingData){

                        #Make prediction using current sample and coefficients
                        z = coefficients[0]

                        for(index = 0 to length of coefficients - 1){
                                z += coefficients[index] * sample[index];
                        }

                        grad = z * coefficients[index] * sample[index] / (1.0 + exp(-z))

                        # Add to error
                        sumError += grad

                        # Calculate current coefficient value
                        coefficients[0] = coefficients[0] + alpha * sumError * grad * (1 - grad)

                        # Update remaining coefficients
                        for(index in coefficients){
                                coefficients[index + 1] = coefficients[index + 1] + alpha * sumError *
                                grad * (1 - grad) * sample[i]
                        }
                }
        }

        return coefficients
}
```

e.) Number of operations per epoch

Since each epoch requires cycling through each sample in the training dataset, the loop will run at most *n* times, or once for each item in the dataset. Inside that loop there is a second for-loop that cycles through each of the coefficients, the total number of which is equal to the dimensionality, *d*, of the sample. Therefore, the total running time in Big-O notation is O(nd).