1. Introduction
2. Clear description of dataset and algorithms used

The dataset used in this project is the infamous MNIST dataset[1]. The MNIST dataset consists of 70000 28x28 pixelated images of handwritten numbers. Each image can be represented as a 28x28 matrix with each element containing a value between 0 and 1 representing the darkness of the pixel. Alternatively, the 28x28 matrix can be transformed into a 784x1 vector array for easy processing. The 28x28 representation technically contains more useful information since it conserves pixel proximity, but for this project, I use 784x1.

Preprocessing is an important step to building a classifier. Thankfully, the MNIST dataset is well known for requiring little-to-no preprocessing to test machine learning algorithms https://medium.com/tebs-lab/how-to-classify-mnist-digits-with-different-neural-network-architectures-39c75a0f03e3. For this reason, preprocessing was avoided in order to focus on algorithm implementation. Similarly, cross-validation can result in a more-robust assessment but is not necessary for achieving tangible results with the MNIST dataset.

MNIST designates 60,0000 of the 70,000 images as “training” images and 10,000 images as “testing” images. For this project, the 60,000 “training” images are used as my training dataset and the 10,000 “testing” images are used as my validation dataset.

Various algorithms were tested for classification. The algorithms are split into two broad categories: 1) Regression and support-vector machine algorithms (summarized in table BLAH), and 2) Neural network algorithms (summarized in table BLAH). For the regression algorithms, regularization was necessary since number of images, or data samples, outnumbered features. I used both L1 and L2 regularizations for regressions involving least squares, hinge, and squared hinge loss functions. Additionally, all regression and SVM algorithms used each of 0th, 1st, and 2nd – degree polynomial fits for the data. The fits above 0th degree were achieved by expanding the feature matrix and weight matrix accordingly.

Regression and support-vector machines



Neural Networks



(DESCRIPTION OF ALGORITHMS)

Algorithms were employed by binary classification…

1. Results (tables, figures, discussion)
2. Results (tables figures and discussion)

Regression

Least-squares loss, l1 regularized, (0, 1, 2 degree)

Least-squares loss, l2 regularized (0, 1, 2 degree)

Hinge loss, l2 regularized (0, 1, 2 degree)

Squared Hinge loss, l1 regularized (0, 1, 2 degree)

Squared Hinge loss, l2 regularized (0, 1, 2 degree)

Neural Network

Least squares loss

relu activation functions

2 layers, 3 layers

sigmoid activation function

2 layers, 3 layers

Categorical cross-entropy loss

relu activation functions

2 layers, 3 layers

sigmoid activation function

2 layers, 3 layers

1. Results (tables figures and discussion)
2. Strengths and limitations

Performance could likely be improved by centering and scaling images or by splitting the “7” class into two separate classes, one where class contains 7’s with the central cross (7“ink free” font) and one class is without the central cross (7).

|  |  |  |
| --- | --- | --- |
| **Loss Function** | **Activation** | **No. Hidden Layers** |
| Least squares | relu | 1, 2 |
| sigmoid | 1, 2 |
| Categorical cross-entropy | relu | 1, 2 |
| sigmoid | 1, 2 |

|  |  |  |
| --- | --- | --- |
| **Loss Function** | **Regularization** | **Fitting Degrees** |
| Least squares | L1 | 0, 1, 2 |
| L2 | 0, 1, 2 |
| Hinge | L2 | 0, 1, 2 |
| Squared hinge | L1 | 0, 1, 2 |
| L2 | 0, 1, 2 |

|  |  |  |
| --- | --- | --- |
| **Loss Function** | **Regularization** | **Fitting Degrees** |
| Least squares | L1 | 0, 1, 2 |
| L2 | 0, 1, 2 |
| Hinge | L2 | 0, 1, 2 |
| Squared hinge | L1 | 0, 1, 2 |
| L2 | 0, 1, 2 |

1. conclusion