Classifying Numerical Values of Handwritten Digits

Ben Wright
Department of Mathematics
UW-Madison
Madison, WI 53703
`bwright8@wisc.edu`

Abstract

Greyscale data of 60,000 handwritten digits from the MNIST database were used to train classifiers to predict the numerical value of a handwritten digit from test data of 10,000 labeled digits. Classifiers were based on regression, support vector machines, and neural networks. Accuray of the classifiers ranged roughly between 80 to 98 percent.

1 The Data

The present project focused on the MNIST database of handwritten digits, which may be found at `http://yann.lecun.com/exdb/mnist/`. The data consists of a training set of 60,000 labeled, fixed-sized, greyscale images of handwritten digits and a test set of 10,000 handwritten digits with the same properties. The images are saved in a special file format called IDX, which can be converted to a NumPy array of the greyscale values of the pixels with the library idx2numpy found at `https://pypi.org/project/idx2numpy/`.

Further, the website hosting the data describes that the data from the trainnig set is comprised of 30,000 handwritten digits from employees of the Census Bureau and 30,000 digits recorded from high school students. The test data is also a fair mix of handwritten digits from these two categories. There are at least 250 different writers represented in the training data. Also, note some greyscale pixels appear due to anti-aliasing used to normalize the data.

2 The Algorithms Used

The datas are labelled according to the value of a digit the handwriter was instructed to write, making the dataset prime for analysis by supervised learning techniques. The question that was proposed for study was: Can the computer be trained to learn the (label) value of a handwritten digit from its IDX representation? The three algorithms that were used to approach this question were: 1. linear/ridge regression. 2. support vector machines. 3. neural networks. A brief overview of each mehtod is given in the following subsections. Note that the implemented classifiers are saved in the project page at `https://github.com/bwright8/csproject`.

2.1 Linear Regression

The idea of linear regression for binary classification is as follows: Given a set of training data X with labels y, interpret the features as inputs and the labels as output. Then, compute the hyperplane represented by w in the feature-label space that minimizes the sum of squares of the difference between the actual labels y and the value of the hyperplane evaluated at each point of training data. Namely, $||Xw - y||_2^2$ is minimized. Now, each side of the hyperplane in the feature space corresponds

Preprint. Under review.

to a predicted label for any input data. The hope is that because the squared-error is minimized that there are few misclassifications.

However, the model is biased because it assumes that labels can be predicted by a linear function. One way to try to reduce the bias is to introduce a regularization parameter λ that tells the regresssion model to prefer w with smaller coefficients. Namely, instead of minimizing $||Xw - y||_2^2$, we seek to minimize $||Xw - y||_2^2 + \lambda||w||_2^2$. Both regularized (ridge) regression and standard linear regression have multiple algorithmic solutions, say, such as gradient descent, but they also have closed-form solutions, which were the solutions used in this project. Namely, for linear regression, the solution is $w = (X^T X)^{-1} X^T y$, and for ridge regression, we have $w = (X^T X + \lambda I)^{-1} X^T y$.

The advantage to the regression model is that it is computationally fast, but the model has high bias because it assumes the function to be learned is linear or affine, meaning, the decision boundary of the classifier is going to be linear. So, if the data is not linearly seperable, this model will make many misclassifications. Also, the model is extremely vulnarable to outliers in the data because such outliers account for a large portion of the mean-squared error, which the model tries to minimize. This might disrupt our present analysis because maybe, for example, some writers cross their 7s with a slash through the middle, but maybe most do not. Or maybe, some of the high school students surveyed simply have awful handwriting.

2.2 Support vectors

A support vector machine is similar to a regression model in the sense that we are looking for a hyperplane to separate data in the feature-label space, only now, we consider a different measure of inaccuracy of the model, that is, a different "loss" function. In what follows, the following loss function was used: $\max(0, 1 - y_i(w^T X_i))^2$ where y_i is 1 or -1 if the datapoint X_i respectively is or is not a certain digit, and so the expression the SVM algoritm tries to minimize is

$$\frac{1}{60000} \sum_{i=1}^{60000} \max(0, 1 - y_i(w^T X_i))^2 + \lambda||w||_2^2$$

where $\lambda > 0$ is some regularization parameter.

When the data is linearly seperable, w can be understood at the hyperplane that separates the data that maximizes the margin between the hyperplane and the data. When the data is not seperable, a hyperplane is penalized by the above loss function for making a misclassification.

This algorithm that was implemented in this project to build the SVM classifier was the sklearn.svm.LinearSVC method. Basically, the algorithm is iterative; it makes an initial guess for the classifier, and then updates the guess based on a (subgradient) descent-style computation on the loss function. The algorithm terminates when losses of sequential guesses are within a certain distance threshold of each other. Details can be found in `https://www.csie.ntu.edu.tw/~cjlin/papers/liblinear.pdf`.

Support vectors are, as noted, again classifiers with linear decision boundaries and therefore introduce some bias in the learning process. Although, one advantage support vector machines have over linear regression in this setting is that support vector machines are relatively immune to outliers in the data. This is because, when the data is linearly seperable, for example, the decision boundary depends only on the datapoints closest to a separating hyperplane.

2.3 Neural Networks

A neural network is an organized collection of nodes arranged layers, the first layer being the input layer that takes the data to be classified. Data from one layer is then weighted and passed into each node in the next layer, where a certain nonlinearity is applied. Eventually, the final (output) layer outputs the predicted class of the data.

The advantage of neural networks in a learning problem such as this one is that neural networks, given enough nodes on a bounded number of layers, can learn any function, meaning, they can have highly nonlinear decision boundaries. In other words, the model will introduce very little bias inot the learning process.

Table 1: Total Misclassifications on the Test Set of the Digit Classifiers

Digit	0	1	2	3	4	5	6	7	8	9
Misclassifications	980	1135	1032	1010	982	892	958	1028	974	1009

For this project, the weights of the networks were trained by doing a stochastic-gradient-descent-type algorithm called "Adam." The essential difference between Adam and standard stochastic gradient descent through backpropogation - which updates weights in the network by utilizing the chain rule of calculus to compute the gradient of the loss function and update the weights in the network - is that Adam uses different learning rates for each weight in the network and updates the weights at each iteration of the descent based on how quickly the network is learning the classifier. See, for example, `https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/` for a further explaination and see also the sklearn documentation for multilayered neural networks at `https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html`.

One serious disadvantage to neural networks, though, in the context of this problem, is that they have low-interpretability. For example, with both support vectors and linear classifiers, one could look at the magnitude of a weight in the classifier and conclude wether or not the corresponding feature played an important role in the classification. For neural networks, the model is not as transparent. So the price one pays for the low-bias of the model is low-interpretability.

Another clear disadvantage to this model, of course, is it takes lots of computation, because instead of learning one weight per feature, the number of weights that must be learned (in a fully connected network) is roughly the product of the sizes of the layers. One way around this difficulty is to train the model first on subsets of the training data to maximize generalizability before training on the full training set.

3 Results

3.1 Linear regression

A binary, linear classifier was trained on all training data for each digit between 0 and 9. A digit selector then takes an input data and checks which of the ten classifiers assigns the data the highest output norm and predicts the corresponding digit. This process is 82.38 percent accurate on the test data of 10,000 handwritten digits. One might wonder which digits are the hardest to distinguish. Well, it turns out the digit 5 was the easiest to classify with over 91 percent accuracy, and 1 was the hardest to classify with just over 88 percent accuracy. (These figures mean that, for example, given any test data, the classifier could correctly identify whether or not the digit was 5 over 91 percent of the time.) More details can be found in Table 1.

As for whether this model can be improved by introducing a regularization parameter, the answer strangely appears to be not really. In fact, regularization parameters $1, 10, 100, 10000$ and 10000 appeared to have no affect on the generalizability of the model (despite the non-regularized classifiers having L2 norms each at least on the order of 10).

3.2 Support vector machines

The support vector classifiers performed better than the linear classifiers. The support-vector-digit-selector, formed the same was as the linear-regression-digit selector described above, predicted the correct digit on the test data 89.46 percent of the time (with regularization parameter $\lambda = 1$). The individual digit classifiers ranged between over 79 percent effective (for the digit 8) to over 99 percent effectiveness (for the digit 1). More details can be found in Table 2.

Again, this model appeared to be relatively immune to changes in the regularization parameter. For example, comparing the above digit classifiers to their counterparts with regularization parameters .1 and 10, misclassifications on the test set changed only by two-tenths of a percent (despite the classifiers having large enough L2 norms to be affected significantly by these regularizations).

Table 2: Total Misclassifications on the Test Set of the Digit Classifiers

Digit	0	1	2	3	4	5	6	7	8	9
Misclassifications	143	67	227	482	186	234	231	172	2070	736
False Negatives	10	22	88	52	60	164	27	84	45	73
False Positives	133	45	139	430	126	70	204	88	2025	663

Table 3: Confusion Matrix for the Neural Network Classifier

$$\begin{bmatrix}
969 & 0 & 3 & 1 & 0 & 3 & 2 & 0 & 3 & 1 \\
0 & 1123 & 4 & 0 & 1 & 0 & 3 & 2 & 0 & 2 \\
0 & 1 & 1006 & 5 & 4 & 0 & 1 & 6 & 3 & 0 \\
0 & 2 & 5 & 988 & 0 & 10 & 1 & 3 & 2 & 2 \\
0 & 0 & 3 & 0 & 964 & 1 & 5 & 0 & 1 & 9 \\
0 & 1 & 0 & 2 & 1 & 869 & 5 & 0 & 6 & 2 \\
3 & 2 & 1 & 0 & 3 & 3 & 940 & 0 & 2 & 2 \\
1 & 1 & 4 & 3 & 2 & 1 & 0 & 1008 & 2 & 3 \\
5 & 5 & 6 & 4 & 0 & 3 & 1 & 3 & 949 & 5 \\
2 & 0 & 0 & 7 & 7 & 2 & 0 & 6 & 6 & 983
\end{bmatrix}$$

3.3 Neural network

The most accurate neural network implemented for this project is a fully-connected network with ReLU activation functions and three hidden layers of sizes 546, 210, and 126. The number of layers and layer sizes were tweaked and selected by training the network on subsets of the training data and using cross-validation. The subsets of the data used to train the model ranged in size between 1000 to 3000 datapoints and averaged between 80 to 90 perecent effectiveness on hold-out data from the training set.

After the model was trained on the full training set of 60000 entries, accuracy (on the testing set) jumped up to 97.99 perecent. The very simple explaination for this jump in effectiveness is that the network learned the patterns of the handwritten digits better with more samples to look at.

How effective was the classifier when breaking the data down by digit? The confusion matrix appears in Table 3.

4 Discussion

Based on the nonlinear natures of the data (most digits do not have linear boundaries), it is not surprising that the neural network performed best on classifying the test data. On the other hand, the linear shape of the digit 1 may account for the success the support vector classifier had at identifying this digit, but then why did linear regression fail to perform as well on the digit 1? Perhaps this has to do with the fact that there are multiple ways of writing the digit 1 - for example, as a simple straight segment or with a hat and shoes as it appears here in type. The rigitdity of SVMs in the face of these outliers could explain the difference in performance.

Although, is it fair to compare these three methods simply by the metric of accuracy on the testing set? After all, the predictions for linear regression and support vector machines are based on collections of ten binary classifiers that are trained to look at only one digit at a time; meanwhile, the neural network is trained from the start to sort data into ten different classes. The alternative would have been to use the labels as their numerical values before performing linear regression, but this seemed like a bad idea because there do not seem to be linear relationships between how the digits are written.

Another question raised by this project is: Why is the linear predictor basically immune to a standard regularization? The answer might have to do with the fact that prediction is based on the norm of the prediction vectors formed by applying the matricies for each digit classifer to a datapoint.

Table 4: Percent Accuracy of Prediction Models on Test Set

Linear Regression	SVM	Neural Network
82.32	89.46	97.99

Regularizing changes the norms of the prediction vectors - but often not drastically enough to affect the ordering of the prediction vectors by their norms. Maybe instead, the predictor should select the digit whose corresponding prediction vector closest to 1 in norm instead of the vector in the correct hyperplane with the largest norm.

5 Conclusion

Look one more time at the effectiveness of each algorithm on the test set, shown in Table 4.

Overall, all three of the above methods were effective at classifying the numerical value of a handwritten digit. On smaller training sets, performances of the three methods were comparable, but on the full training set of 60000 data points, the neural network gives predictions with the highest accuracy. Though, improvement of the other two prediction algoritms, making better use of the binary digit classifiers, may be possible.

References

[1] Brownlee, J., 2020. Gentle Introduction To The Adam Optimization Algorithm For Deep Learning. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/> [Accessed 12 December 2020].

[2] Fan, R., Chang, K., Hsieh, C., Wang, X. and Lin, C., 2020. [online] Csie.ntu.edu.tw. Available at: <https://www.csie.ntu.edu.tw/ cjlin/papers/liblinear.pdf> [Accessed 12 December 2020].

[3] Scikit-learn.org. 2020. Sklearn.Neural_Network.Mlpclassifier — Scikit-Learn 0.23.2 Documentation. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html> [Accessed 12 December 2020].

[4] Yann.lecun.com. 2020. MNIST Handwritten Digit Database, Yann Lecun, Corinna Cortes And Chris Burges. [online] Available at: <http://yann.lecun.com/exdb/mnist/> [Accessed 12 December 2020].