Logistic Regression for MNIST Dataset Classification

Brandon Montez

June 9, 2024

1 Introduction

This report focuses on the application of Logistic Regression to classify the MNIST dataset.

The dataset $\text{mnist}_7 84$ is already processed as a training set within the first 60,000 samples, and a test set of the next 10,000 samples.

2 Data Preprocessing

The first step for logistic regression is normalizing onto the range [0, 1]. Standardization is not necessary for logistic regression, but it was applied for better interpretability of the coefficients. It also made evaluating other models easier. The training data had each row standardized, and those parameters were used to treat the test set to avoid data leakage from the holdout.

The data was first split into a train/test, and a validation set was made from the training data during nested K-Fold Cross Validation. The number of splits was chosen to be 6, so that the validation sets would be size 10,000, mimicking the size of the holdout set.

3 Model Training and Evaluation

3.1 Hyperparameter Tuning

The logistic regression model with scikitlearn utilizes the L2 norm as the default, which was not adjusted for this evaluation. The hyperparameter which was tuned is the 'C' parameter, which controls the weight of the Regularization term via an inverse relationship.

Nested CV was utilized, involving an outer loop to split the data into training and validation sets, and an inner loop that performs hyperparameter tuning on the training set using randomized search cross-validation across the hyperparameter space, ranging from [0.01, 10], in an attempt to randomly produce both weak and strong regularization terms. Notably, this perhaps produced a biased selection process, as it doesn't evenly disperse the values between [0.01, 1] and [1, 10].

3.2 Performance Metrics

Although other models such as SVM and KNN were similarly evaluated and showed stronger validation performances than Logistic Regression (accuracy scores for 4 models: SVM = 0.98; KNN = 0.95; Logistic Regression = 0.92; Decision Tree = 0.86) logistic regression was explored to gain insight into its conditional probabilistic approach. Proper tuning for the model-wise comparison was not fully completed in the restriction of time, so the accuracy scores reported above are held tentatively.

Scikitlearn uses accuracy as the scoring metric for Logistic Regression by default. This is acceptable because the MNIST dataset is well balanced already.

The logistic model was evaluated using several metrics, including accuracy, precision, recall, F1 score, and ROC AUC. The results are summarized in Table 1.

Model	Accuracy	Precision	Recall	F1 Score	ROC AUC	Log-Loss
Logistic Regression	0.922	0.922	0.922	0.922	0.99	0.33

Table 1: Performance metrics for Logistic Regression.

3.3 Confusion Matrix

The confusion matrix provides a more localized perspective into the model's performance. Notice the column for 9, for example, we can infer that our model does well to understand that 9 is different enough from 0, 1, and 6. But it sometimes had problems identifying a 9 from a 4, or a 9 from a 7. We can see variation among the confidence that our model has when making predictions. The diagonal values are all high comparatively which is a good sign for model performance, assuming that we have not overfit.

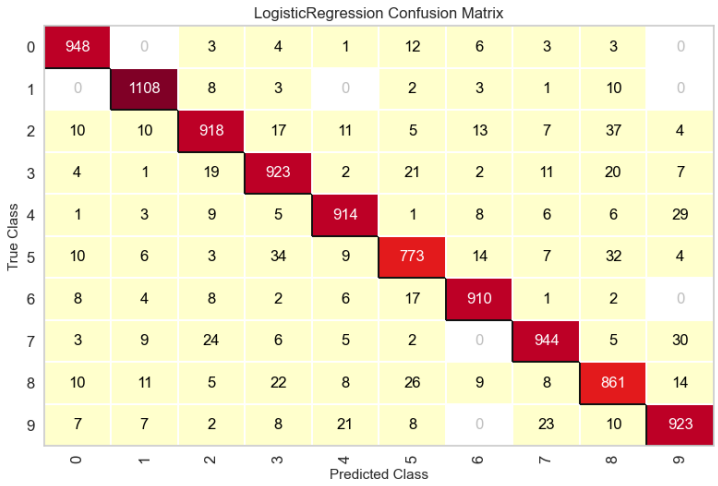

Figure 1: Confusion matrix for Logistic Regression.

Classification Report

	precision	recall	f1-score	support
0	0.95	0.97	0.96	980
1	0.96	0.98	0.97	1135
2	0.92	0.89	0.90	1032
3	0.90	0.91	0.91	1010
4	0.94	0.93	0.93	982
5	0.89	0.87	0.88	892
6	0.94	0.95	0.95	958
7	0.93	0.92	0.93	1028
8	0.87	0.88	0.88	974
9	0.91	0.91	0.91	1009
accuracy			0.92	10000
macro avg	0.92	0.92	0.92	10000
weighted avg	0.92	0.92	0.92	10000

The best model, agreed unanimously by all 6 folds, was C = 0.59. That is, logistic regression with a regularization to penalize large weights.

I believe my hyperparameter sampling method was flawed to skew towards comparatively larger numbers. Yet even with an imbalance we see that the smallest tested hyperparameter value won out. This suggests to me that I should focus on hyperparameter values between [0, 1] for this parameter.

4 Mathematical Analysis of Logistic Regression

4.1 Model Formulation

Logistic regression is a linear model utilizing the logistic sigmoid activation function, $\sigma : R \to (0, 1)$ to form binary classifications.

We can use the logistic sigmoid function to to model the posterior:

$$P(C = 1|\mathbf{x}) = \frac{1}{1 + e^{(-\mathbf{w}^T\mathbf{x} - w_0)}}$$

4.2 Maximum Likelihood Estimation (MLE)

Assume the data are i.i.d and $t \in \{0, 1\}$, then the probability distribution of each of our training points $p(t_1, \cdots, t_N|\phi(\mathbf{x}_1), \cdots, \phi(\mathbf{x}_N))$:

$$p(t_1, \cdots, t_N|\phi(\mathbf{x}_1), \cdots, \phi(\mathbf{x}_N)) = \prod_{n=1}^{N} p(t_n|\phi(\mathbf{x}_n))$$

Which can be written as:

$$p(t_n|\phi(\mathbf{x}_n)) = p(C = 1|\phi(\mathbf{x}_n))^{t_n} p(C = 0|\phi(\mathbf{x}_n))^{1-t_n}$$

To find the optimal vector w such that t_n is most likely conditioned on seeing $\phi(x)$, we maximize the conditional likelihood, which is equivalent to minimizing the log-likelihood:

$$-\ell_{\log}(\mathbf{w}) = -\sum_{n=1}^{N} t_n \log(p(C = 1|\phi(\mathbf{x}_n))) - \sum_{n=1}^{N}(1 - t_n) \log(1 - p(C = 1|\phi(\mathbf{x}_n)))$$

The cross-entropy loss, a binary classifier. Notably, the MNIST dataset is a multi-class problem. Multi-class logistic regression involves a softmax function to convert logits into probabilities for each class. This method was unfortunately outside of the scope of this paper. Instead, a one-vs-all approach was used, evaluating an ROC-AUC curve to see a global picture.

4.3 Gradient of the Log-Likelihood

To maximize the log-likelihood function, we need to compute its gradient with respect to the parameters $\mathbf{w}$. Let's do that now,

We denote $\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n)$ as z_n. Then, $\sigma(z_n)$ is the predicted probability for the n-th data point.

The gradient of $E(\mathbf{w})$ with respect to $\mathbf{w}$ is:

$$\nabla E(\mathbf{w}) = -\sum_{n=1}^{N} \left[t_n \frac{\partial}{\partial \mathbf{w}} \log \sigma(z_n) + (1 - t_n) \frac{\partial}{\partial \mathbf{w}} \log(1 - \sigma(z_n)) \right]$$

Computing the partial derivatives of $\log \sigma(z_n)$ and $\log(1 - \sigma(z_n))$ with respect to $\mathbf{w}$:

- For $\log \sigma(z_n)$:

$$\frac{\partial}{\partial \mathbf{w}} \log \sigma(z_n) = \frac{1}{\sigma(z_n)} \cdot \sigma(z_n)(1 - \sigma(z_n))\boldsymbol{\phi}(\mathbf{x}_n) = (1 - \sigma(z_n))\boldsymbol{\phi}(\mathbf{x}_n)$$

- For $\log(1 - \sigma(z_n))$:

$$\frac{\partial}{\partial \mathbf{w}} \log(1 - \sigma(z_n)) = \frac{1}{1 - \sigma(z_n)} \cdot (-\sigma(z_n))(1 - \sigma(z_n))\boldsymbol{\phi}(\mathbf{x}_n) = -\sigma(z_n)\boldsymbol{\phi}(\mathbf{x}_n)$$

Substituting these partial derivatives back into the gradient expression:

$$\nabla E(\mathbf{w}) = -\sum_{n=1}^{N} [t_n(1 - \sigma(z_n))\boldsymbol{\phi}(\mathbf{x}_n) + (1 - t_n)(-\sigma(z_n))\boldsymbol{\phi}(\mathbf{x}_n)]$$

$$\nabla E(\mathbf{w}) = -\sum_{n=1}^{N} [t_n\boldsymbol{\phi}(\mathbf{x}_n) - t_n\sigma(z_n)\boldsymbol{\phi}(\mathbf{x}_n) - \sigma(z_n)\boldsymbol{\phi}(\mathbf{x}_n) + t_n\sigma(z_n)\boldsymbol{\phi}(\mathbf{x}_n)]$$

$$\nabla E(\mathbf{w}) = -\sum_{n=1}^{N} [t_n\boldsymbol{\phi}(\mathbf{x}_n) - \sigma(z_n)\boldsymbol{\phi}(\mathbf{x}_n)]$$

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} \left[\sigma(z_n) - t_n\right] \boldsymbol{\phi}(\mathbf{x}_n)$$

Replacing $z_n = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n)$:

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} \left[\sigma(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n)) - t_n\right] \boldsymbol{\phi}(\mathbf{x}_n)$$

The logistic sigmoid function is smooth and hence has nice properties for gradient descent methods.

ROC AUC Curve

Relates the True Positive and False Positive rates for each class. We can see some variation in the classes but overall there is practically no interaction. We could restrict our attention to the area of highest variation, but overall, the high ROC AUC score of 0.99 suggests that the MNIST data can be linearly separated quite well. Which I find quite surprising, as I suspected a lot of overlap between numbers would suggest less linear separability. This exemplifies the power of machine learning to still be able to extract knowledge from somewhat fuzzy information. No doubt humans are better at one-shot learning, but machines can see and capture different levels of complexity in the data that we take for granted.

- **Y AXIS - True Positive Rate (TPR)**:

$$\text{TPR} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

- **X AXIS - False Positive Rate (FPR)**:

$$\text{FPR} = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}}$$

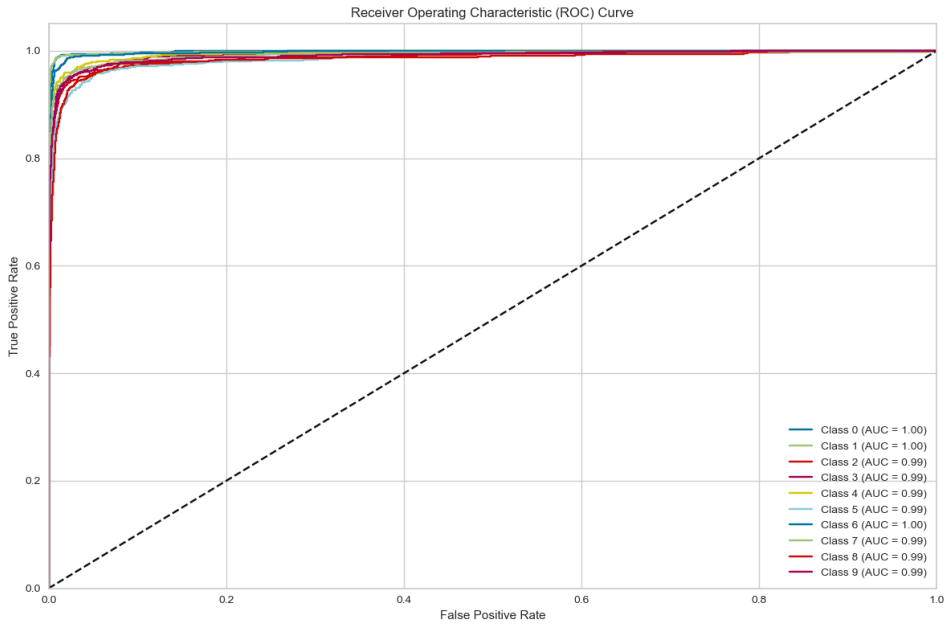

Figure 2: ROC AUC Curve

Gthub repo: https://github.com/brandonmontez/M156-Repo