Final Project Report

Kacie L. Salmon
ECE532: Matrix Methods in Machine Learning
University of Wisconsin-Madison
Madison, WI 53703
`klsalmon@wisc.edu`

Abstract

This work focused on the application and understanding of three machine learning algorithms applied to a large dataset. The selected dataset was the Fashion MNIST, and the three algorithms applied were a least squares classifier, a support vector machine, and a neural network. These algorithms were implemented, validated for parameter choices, and analyzed for performance in classification and run time. The results of this analysis showed that SVM was the best classifier for this dataset if run-time was not an issue, but that least squares is a very good alternative for quick results. The neural network was found to be inappropriate for this dataset or unable to form good weights without very large run-times that were unable to be tested within the scope of this project.

1 Introduction

Machine learning is a growing and relevant field in many sectors. In handling large data-sets and making decisions or identifying classifications, it is unmatched by human abilities to compile and understand large scale generalized trends and correlations [1]. Over the course of the semester, the concepts and algorithms we have learned can be directly applied to these kinds of problems. This final project aimed to apply and analyze the utilization of three machine learning classifiers applied to a single common dataset. In doing this, we can better understand how to apply these algorithms and what effects a choice in algorithm can have on performance of your machine learning classification solution.

2 Fashion MNIST Dataset

The dataset that chosen for this final project was the Fashion MNIST benchmark dataset. The Fashion MNIST dataset consists of images of ten types of different clothing items with associated clothing type labels, as seen in Figure 1 [2]. 70,000 gray scale images of these clothing items make up this dataset, each a 28 x 28-pixel square made up of 784 pixels. These pixels are stored as a pixel-value, an integer from 0 to 255, that indicates lightness or darkness with 255 being the darkest, which is located in a csv file through a number assigned based on the row and column [2].

Each image also has an assigned label, an integer from 0 to 9, indicating what kind of clothing item it is. These images and the associated data are also broken into training and test data that can be used to train and test classifier algorithms on learning and applying correct labels to these images from the data given. The dataset is allocated into two groups, a training and a test set. The sets are made up of 60,000 images and 10,000 images respectively. The dataset was publicly source from Zalando Research [2]. The assignments of class labels and clothing types can be seen below in Table 1.

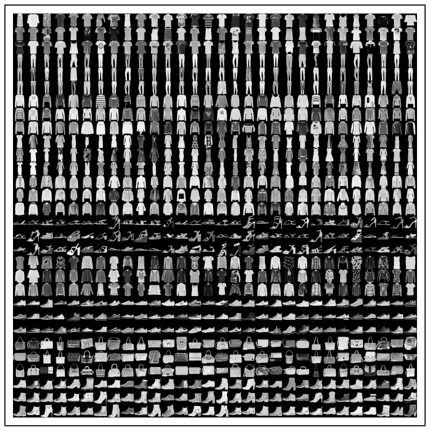


Figure 1: Images of Fashion MNIST Data.

Table 1: Fashion MNIST Labels

Label	Description
0	T-shirt\Top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle Boot

3 Algorithms

Three algorithms were selected and implemented for the chosen dataset. These were a least squares model, support vector machine, and a neural network. Each algorithm-its design, preprocessing and parameters are described below. Each of the algorithms and implementations can be found on the project GitHub page at `https://github.com/klsalmon/ECE532_FinalProject` [3].

3.1 Least Squares

A least squares linear regression classifier was the first algorithm chosen to be applied to the dataset. Regularization through ridge regression was utilized to ensure the least squares could solve and to tune the error, and a one-versus-the-rest scheme was designed to solve the multi-class labeling problem. This one-vs-all scheme was implemented by creating separate weight vectors for each label that classifies if the label is applied or not. These are found by creating new feature vectors of the data that identify the one label being tested for each weight vector. These label vectors assign a 1 or -1 label to each item identifying if it is or is not the chosen label. Each of these label vectors, y^j, are used to train an individual weight vector with notation f_k^j where j is a number 0-9 that represents each label respectively. The weight vectors are trained with least squares and then the final label classification is found with the equation [4]:

$$\hat{y} = argmax_{j=0..9}(X^T f_k^j)$$

Preprocessing In order to prepare the data for processing, singular value decomposition was performed. This decomposed the least squares solution into the form:

$$f_k^j = VDU^T y^j$$

where the D matrix is the diagonal matrix of the singular values with the regression applied, which are given in a list of d, given by:

$$d = s/(s^2 + \lambda)$$

3.2 Support Vector Machine (SVM)

The support vector machine was created with an sklearn.svm module with prepared algorithms [5]. The chosen algorithm from sklearn was the LinearSVC module, and parameters were tested to ensure they allowed for best results, as discussed further in results.

Table 2: LinearSVC Parameter Choices

Parameter	Value	Rational
multi class	ovr	Set the SVM for one-vs-all multiclass.
max iter	40000	Convergence issues led to larger iteration maximum.
tol	1e-4	Default tolerance chosen to allow convergence.
C (regularization term)	0.0001	Parameter variance tested, best used.

The main parameter choices made are available in Table 2. These were chosen based on the correct setup for multi-class, a one-vs-all scheme, allowing for the SVM to converge, and regularization.

3.3 Neural Network

The last classification algorithm applied to the dataset was a neural network. The neural network consisted of three layers, an input layer, one hidden layer, and the output layer. The output layer consisted of the ten one-vs-all classifiers necessary for the multi-class scheme as implemented in the other algorithms as well. The neural network was implemented with a backpropagation algorithm and a Gaussian activation function. Variables tested included the backpropagation iterations (epoch) and step size (alpha). For the number of hidden nodes, 100 was chosen to ensure the network was sturdy and able to capture complicated patterns.

4 Results and Discussion

4.1 Results

Each of the implemented algorithms were compared through their performance on the test dataset after being trained. In addition, the parameters of each algorithm were analyzed.

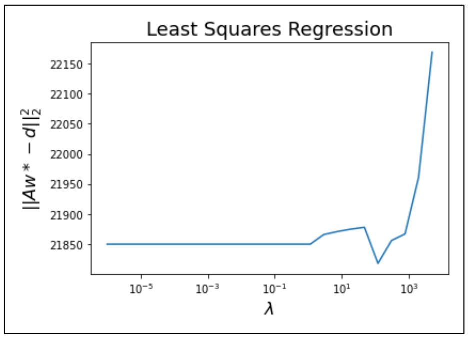
Table 3: Overall Results for Given Training and Test Sets

Algorithm	Misclassification Error	Run-time
Least Squares	1782	20.122 s
SVM	1554	5399.872 s
Neural Network	9235	2786.818 s

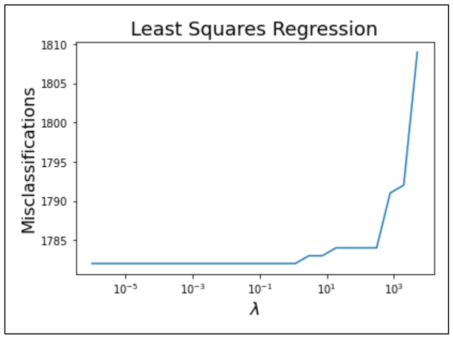
Overall, when the training dataset was used to train each algorithm and then tested on the designated test dataset with the chosen parameters, the error rates and run-times of the algorithms were as shown in Table 3. As can be seen, the least squares algorithm ran significantly faster than the other two algorithms, but misclassified data at a higher rate. In contrast, the SVM algorithm increased the run-time quite drastically, taking nearly an hour - 180 times longer than the least squares, and reducing misclassification by roughly 12.8%. Perhaps there was an issue in implementation, but it seems that the neural network was a poor fit for this dataset.

4.1.1 Least Squares Parameters

The least squares parameter for regression was varied to find the effect on misclassification and fit. As seen below in Figure 2a and Figure 2b, the regularization term becomes an issue as it becomes larger than 0.2, but below that effectively has the same effect. For this reason a regularization parameter of 0.1 was chosen.



(a) Squared L2-Norm

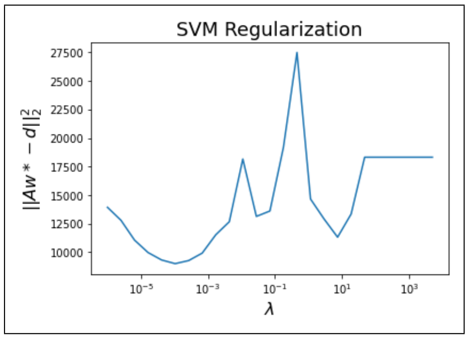


(b) Misclassifications

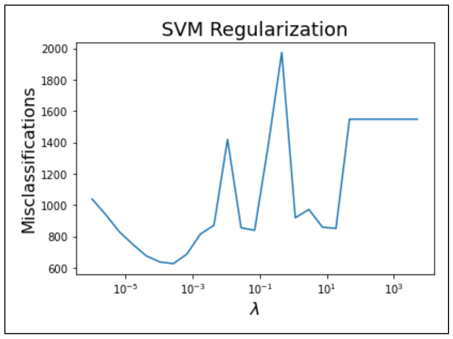
Figure 2: Ridge regression λ choice for least squares.

4.1.2 Support Vector Machine Regularization Term

Shown below in Figure 3a and Figure 3b, the support vector machine regularization term was also varied to view effect on performance. Above values of 0.1, the program has issues converging leading to those large inconsistencies in the results with a value above 0.1. Below this though, the code converged and saw the best results at a λ in the range of 0.0001. For this reason the regularization term for the algorithm was set to 0.0001.



(a) Squared L2-Norm



(b) Misclassifications

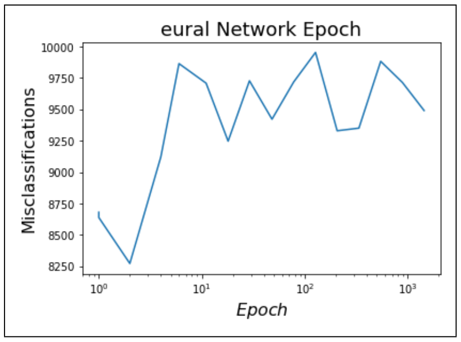
Figure 3: Ridge regression λ choice for SVM.

4.1.3 Neural Network Backpropagation

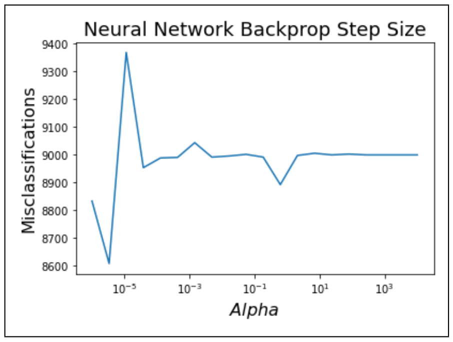
The neural network utilizes backpropagation, which requires an epoch term that dictates the number of backpropagation iterations and an α term that dictates the step size of the backpropagation. Each of these variables was tested for performance, as shown below in Figure 4a and Figure 4b. Discernible patterns were not truly found, and neither variable had a strong correlation with improved results. It seems that this was a result of using too small of an epoch for this large dataset. The patterns did not seem well captured, even with a robust hidden layer and many variable implementations of epochs and alpha values.

4.2 Discussion

As can be seen from the results, performance of the classifiers varied greatly and varied dependent on parameter choice as well. Each algorithm had unique challenges in implementation and difficulties associated with understanding how parameters and iterations effected the outcomes.



(a) Backpropagation Iterations



(b) Backpropagation Step-size

Figure 4: Parameter variance for neural network.

Strengths and Limitations For least squares, the main strength of the algorithm is its simplicity and solution time. The limitation of least squares though is its performance isn't as good as other options and it can't be improved greatly. The support vector machine has the strength of better performance but it has limitations in convergence and very long run-times. Neural networks have the strength of easy implementation of many class problems and the ability to capture complex decision boundaries, but it is difficult to implement and possibly requires much longer run-times than was capable of being run in the scope of this project. The decision boundaries were not well captured in this application.

5 Conclusion

In conclusion, the implementation of these three algorithms was an interesting way to strengthen the understanding of concepts from the Matrix Methods for Machine Learning course. It was a unique challenge to work with such a large dataset and need to apply techniques such as preprocessing to handle it. Compared to the classwork, implementing the algorithms for a new dataset was a unique challenge and taught new issues that can arise and how to address them. From the results, one can see that even though in theory the more complex and recent methods would be best, the least squares was a quick and relatively easy algorithm to implement with comparable results in terms of misclassification. It also showed that neural networks and backpropagation and be difficult to implement.

References

[1] Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, 55(10), 78-87. doi:10.1145/2347736.2347755

[2] Zalando Research. (2017, December 07). Fashion MNIST. Retrieved November 14, 2020, from `https://www.kaggle.com/zalando-research/fashionmnist`

[3] Salmon, K. (2020). ECE532 Final Project. Retrieved from `https://github.com/klsalmon/ECE532_FinalProject`

[4] Band, A. (2020). Multi-class Classification - Onve-vs-All and One-vs-One. Retrieved from `https://towardsdatascience.com/multi-class-classification-one-vs-all-one-vs-one-94aed32a87b`

[5] Sci-kit Learn. (2007). sklearn.svm.LinearSVC. Retrieved December 1, 2020, from `https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html`