Comparing Multilayer Perceptrons and Support Vector Machines on Handwritten Digit Classification

Author: Jerome Nainan Titus
Email: jerome.titus@city.ac.uk

1. Problem Description and Motivation

The employment of machine learning methods on handwritten recognition proliferated due to a confluence of factors including increased computing power, novel algorithms and the availability of large datasets containing handwritten characters [1]. An example of the utility of these methods can be observed in the US Postal Service – where they are used to read zip codes on mail [2].

The objective of this paper is to critically evaluate and compare multilayer perceptrons and support vector machines on the task of handwritten digit classification - this task serves as a good benchmark of comparison for algorithms related to automated shape recognition [3]. While model performance on identifying machine generated text is quite good, handwritten digits pose a more challenging problem given a multitude of factors, including the subjective nature of handwriting. For example, a 2 written by one individual may significantly differ to that written by another.

2. Description of Dataset

The dataset used for this task was obtained from Kaggle [4] - where it was collected from the MNIST handwritten digit database [5] and converted to a csv format. The handwritten digits come from approximately 250 writers. The training and test sets consist of 60000 and 10000 images, respectively, with the writers in each dataset being disjoint. To improve training time for grid search, a validation set was created by randomly sampling 10% of the training set.

Each row in the training, validation and test sets correspond to a single image. The first columns contain discrete labels for each image (0-9), with the remaining representing the pixel values for each of the 784 pixels within the 28x28 images. These values are integers, ranging from 0-255, where 0 and 255 correspond to black and white pixels, respectively. To improve convergence properties in model training, these pixels were re-scaled to fit within the interval [0,1].

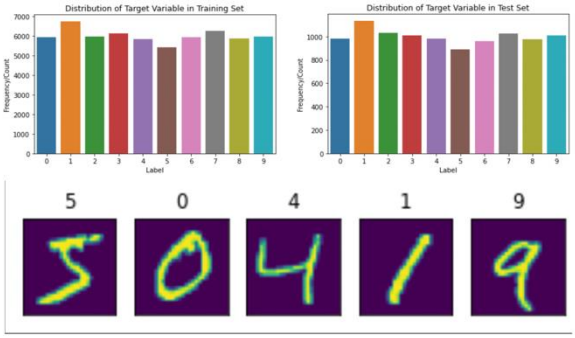


Fig 1 showing distribution of target class + Sample images

Basic EDA showed no presence of outliers or missing values in the dataset. All entries corresponded to integers – pixel values were converted to floats to ensure compatibility with model building libraries.

3. Brief Summary of Models Used

3.1. Multilayer Perceptrons

3.1.1. Summary

Multilayer perceptrons are neural networks – which serve as function approximators. The mechanisms behind neural nets are motivated from neurobiological processes, where the firing of a neuron sets off (or "activates") neurons in its neighbouring vicinity, initiating a chain reaction of fired neurons throughout the brain (or "network"). The interconnections between neurons are represented as "weights", and the thresholds for activation by "biases". The goal of the network then becomes to employ optimization techniques borrowed from numerical analysis (e.g., backpropagation), which alters these weights and biases to minimise a specified loss function.

3.1.2. Pros

a) Serve as universal function approximators with the utilization of just one hidden layer and sigmoid activation function [6]
b) Parallel distribution and interconnection of neurons allow for modelling of complex relationships

3.1.3. Cons

a) Often referred to as a “black-box” algorithm due to its complexity, limiting explanations of model results by the human analyst
b) Can get stuck in local optima given large parameter spaces
c) Results may vary depending on initialisations of parameter space – leading to non-unique solutions

3.2 Support Vector Machines

3.2.1. Summary

Support vector machines (SVMs) were introduced as a statistical learning technique in classification problems. Its central tenet lies in the separation of classes by a decision boundary - the goal being for this boundary to separate classes by as wide a margin as possible, specifying a convex optimisation problem.

Motivation for SVMs stemmed from maximal margin classifiers. These classifiers performed sub-optimally on non-linearly separable datasets, leading to the introduction of support vector classifiers and the idea of a "soft margin". Problems arose with support vector classifiers for data containing lots of overlap. SVMs solved this problem by introducing the idea of a kernel - which transforms the data to higher dimensional space to find an optimal decision boundary.

3.2.2. Pros

a) Convex maximisation yields a unique solution – an advantage over neural nets
b) Kernels allow transformation of data to find optimal decision boundaries

3.2.3. Cons

a) Utilisation of kernel is expensive – given that it operates on all pairs of training examples

4. Hypothesis Statement

Given **3.2.2.**, it is hypothesized that the support vector machine will achieve a greater testing accuracy than the multilayer perceptron.

5. Choice of Training and Evaluation Methodology

5.1. Choice of Training

Per algorithm of study, base models were constructed to serve as a benchmark for comparison. Successive iterations of training involved adaptations to this base model, with results informing decision making in hyperparameter selection, and in the case of the neural network, modification of its architecture. Finally, grid search was implemented to analyse the best combinations of hyperparameters per model.

5.2. Methodology

1) Load data and split into training, validation, and test sets
2) EDA (Missing value/outlier detection, investigation of target variable)
3) Data preparation (scaling, conversion of data types): Scaling was utilised to improve model convergence properties.
4) Construction of base model + Evaluation of results
5) Adaptation of hyperparameters of base model (for MLP, this included adaptation of network architecture) + Evaluation of results
6) Hyperparameter tuning using grid search + Evaluation of results: Hyperparameter tuning was conducted on the validation set to mitigate time constraints
7) Optimal model construction using previous results + Evaluation of results

Note: Evaluation of results involved running the built models on the test set, plotting confusion matrices and learning curves, as well as identifying misclassified digits.

6. Choice of parameters and Experimental Results

6.1. Choice of parameters

6.1.1. MLP

The hyperparameters of interest are:

- **Max_epochs** - An epoch passes when the model has run through every example in the training set. Increasing the number of epochs increases the number of alterations made to the weights and biases of the network. Values tested = {20, 50}
- **Learning rate** - The learning rate controls the size of the updates made to the network parameters. Low learning rates may require a longer number of epochs as learning is slow. High learning rates may correspond to faster learning, but the network may converge to sub-optimal results. Values tested = {0.025, 0.1, 0.25}
- **Weight Decay** - Penalizes the loss function to reduce changes made during updates. It is used to prevent overfitting as well as prevent the gradient explosion problem. Values tested = {0.1, 0.01}
- **Momentum** - A downside of neural networks is their convergence to local optima. Momentum helps the network "jump" out of these extremums in the error space. Values tested = {0.8, 0.9}
- **Number of neurons in hidden layers –** Large values may lead to overfitting

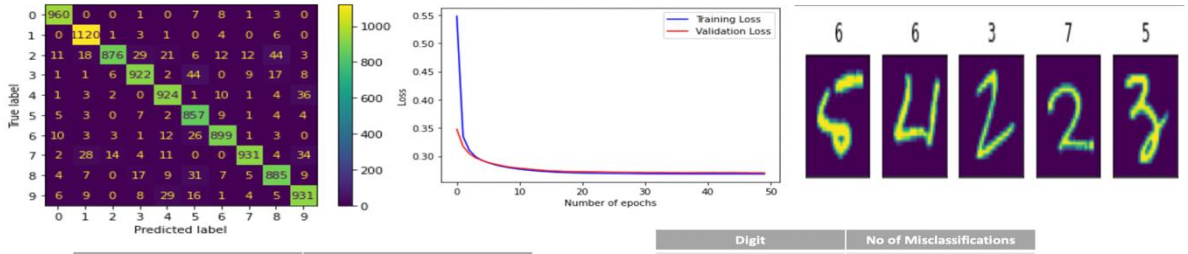
6.1.2. Support Vector Machine

The hyperparameters of interest are:

- **Kernel** - transforms the input data to obtain a better decision boundary. This transformation can be linear. In the case where classes cannot be linearly separated, other functions can be utilised (e.g., polynomial kernel, when this is used, an additional hyperparameter, "degree" can be specified). Values tested = {“poly”, “rbf”}
- **C** (regularisation parameter) - indicates the level of importance given to misclassified observations. Higher values of C correspond to a narrower margin, which can result in overfitting. Lower values of C imply a wider margin, which in turn can result in underfitting. Values tested = {0.5, 10}
- **Gamma** - regulates the amount of curvature in the decision boundary. Higher values correspond to more curvature and may lead to overfitting. Lower values imply less curvature, and may result in underfitting, as the decision boundary may fail to capture the complexities within the dataset. Values tested = {0.01, 10}

6.2. Experimental Results

6.2.1. Multilayer Perceptron



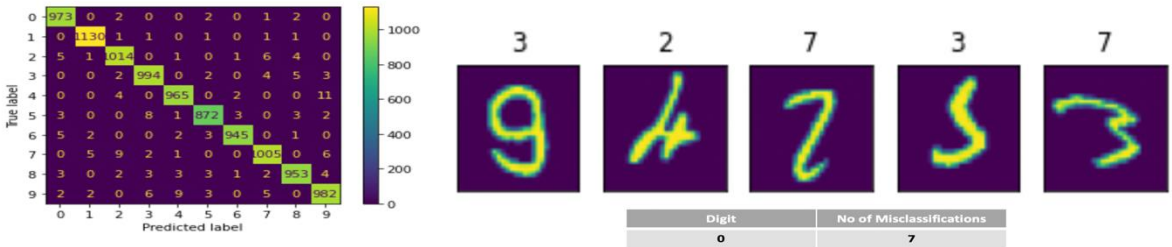
Hyperparameter	Optimal Value
Learning Rate	0.025
Max Epochs	50
No of neurons in hidden layer	200
Momentum	0.9
Weight Decay	0.01

Accuracy obtained: 0.9305

Digit	No of Misclassifications
0	20
1	15
2	156
3	88
4	58
5	35
6	59
7	97
8	89
9	78
Total	695

Figure 2 showing experimental results for the Multilayer Perceptron

6.2.2. Support Vector Machines



Hyperparameter	Optimal Value
C	10
Gamma	0.01
Kernel	rbf

Accuracy Obtained: 0.9833

Digit	No of Misclassifications
0	7
1	5
2	18
3	16
4	17
5	20
6	13
7	23
8	21
9	27
Total	167

Figure 3 showing experimental results for the Support Vector Machine

7. Analysis and Critical Evaluation of Results

Analysis of the results above show that the SVM outperforms the MLP for the given task, with model accuracy on the test set corresponding to 0.9833 and 0.9305, respectively. This validates our hypothesis from section 4.

7.1. MLP

Figure 2 shows a minimal difference in the training and validation losses for this model, suggesting that it does not suffer from overfitting. This can be explained by the inclusion of the weight decay parameter. Additionally, both losses seem to level out after 10 epochs. This suggests the need for early stopping which can optimise time constraints during training.

The optimal number of neurons from grid search were 200, compared to 50. More neurons imply that the network can accurately model complex intricacies within the dataset. While a higher number of neurons may lead to overfitting, the point above shows that this does not occur.

The optimal learning rate corresponded to 0.025, which was the lowest of the values tested during tuning. Lower rates correspond to smaller updates to parameters, implying that the model is cautious about making updates on newly learned information. This is an important point given the similarity of certain digits. New information may push the model to misinterpret these digits, so it is important that it makes updates cautiously.

To analyse the point of digit similarity made above, we refer to the confusion matrix in Figure 2. Two similar digits are 2 and 8. Observing the matrix, 2 is misclassified as 8, 44 times. This is the largest misclassification error between any pair of digits. One possible explanation for this is that the model has not sufficiently captured the complexities of the dataset to separate the two. One method of improvement may be to increase the number of hidden layers, or number of neurons in these layers. Both improvements will correspond to greater complexity – but care will need to be taken to ensure overfitting does not occur.

Regarding momentum, a higher value was chosen as a result of grid search. This seems like a reasonable result. Given the high dimensionality of the parameter space, a higher momentum will increase the likelihood of the model “escaping” from local optima.

7.2.2. SVM

The optimal kernel chosen was the radius basis function kernel. This kernel provides the advantage of being able to work optimally in high-dimensional feature spaces – like ours. In fact, this kernel can operate in infinite dimensions. Its power lies in its similarity to the Gaussian distribution, which has proven to be a very useful tool in mathematics. Hyperparameters related to this kernel include the regularisation parameter C, and gamma – the effects of each are discussed below.

The optimal value of gamma corresponded to 0.01 – in comparison to 10. This implies less curvature in the decision boundary, prohibiting the model from overfitting.

The value of the regularisation parameter, C, chosen from hyperparameter tuning was 10, in comparison to 0.5. This implies that high importance was given by the model to misclassifications of observations, resulting in a narrower margin.

Both results above show that special attention was made by the model to prevent overfitting.

8. Conclusion, Lessons Learned and Future Work

8.1. Conclusion

The results of model comparison between support vector machines and multilayer perceptrons show that SVMs outperform MLPs for the task of handwritten digit recognition.

8.2. Lessons Learned

The primary lesson learned during this research was the importance of paying attention to hyperparameters, and what they explicitly mean. For example, for the MLP, by employing critical thinking on the effects of hyperparameters, adaptations made improved model performance by a greater margin than grid search (indicating that a little bit of critical reasoning can outperform brute force methods, which can be computationally expensive). Please see implementation details below for more information.

8.3. Future Work

8.3.1. Investigation of additional models such as convolutional neural networks to expand analysis and facilitate greater comparison between algorithms.

8.3.2. Introduce noise to datasets to test model robustness

9. References

[1] LeCun, Y., Jackel, L.D., Bottou, L., Cortes, C., Denker, J.S., Drucker, H., Guyon, I., Muller, U.A., Sackinger, E., Simard, P. and Vapnik, V., 1995. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective*, *261*(276), p.2.

[2] Matan, O., Baird, H.S., Bromley, J., Burges, C.J.C., Denker, J.S., Jackel, L.D., Le Cun, Y., Pednault, E.P.D., Satterfield, W.D., Stenard, C.E. and Thompson, T.J., 1992. Reading handwritten digits: A zip code recognition system. *Computer*, *25*(7), pp.59-63.

[3] LeCun, Y., Jackel, L., Bottou, L., Brunot, A., Cortes, C., Denker, J., Drucker, H., Guyon, I., Muller, U.A., Sackinger, E. and Simard, P., 1995, November. Comparison of learning algorithms for handwritten digit recognition. In *International conference on artificial neural networks* (Vol. 60, pp. 53-60).

[4] Dariel Dato-On (Year unknown) MNIST in CSV *Retrieved from:* https://www.kaggle.com/datasets/oddrationale/mnist-in-csv?resource=download

[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE*, 86(11):2278-2324, November 1998.

[6] Cybenko, G., 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, *2*(4), pp.303-314.

10. Implementation Details

10.1. MLP

The steps of implementation for MLP are shown below.

1. Construction of base model: Models were built using the PyTorch and Skorch libraries. For this task, the NeuralNetClassifier from Skorch was utilised. The base model was ran using the default parameters specified in Skorch. For example, the default value for the number of epochs was 10. The base architecture of the network consisted of 784 input units, 1 hidden layer consisting of 50 neurons and 10 output units.
2. Adaptation of basic network architecture: An additional hidden layer was added, with the number of neurons increased to 100, for each hidden layer. This iteration showed very slight improvement in results (model accuracy improved from 0.9083 to 0.9088). It was decided to revert to the initial iteration.
3. Adaptation of hyperparameters: Hyperparameter adapted included the learning rate, which was increased from 0.01 to 0.025 and max_epochs, which was increased from 10 to 20.
4. Grid search was finally run (using sklearn's GridSearchCV) to investigate a greater range of values for each hyperparameter to obtain optimal results – these are shown in Figure 2.

10.2. SVM

The steps of implementation for SVM are shown below:

1. Construction of base model: Models were built using sklearn's SVC function. The base model was run using the default hyperparameters provided by the library. The only specification was setting the kernel to "linear".
2. Adaptation of hyperparameters: Hyperparameters adapted included the kernel used, which was changed to "poly", which has a default "degree" of 3. The C and gamma values were changed to 0.5 and 10 respectively. A significant improvement in model accuracy was observed – an increase from 0.9404 to 0.9787.
3. Grid search was finally implemented to determine the optimal range of hyperparameter values. Results are shown in Figure 3.

11. Glossary

C (regularisation parameter) - indicates the level of importance given to misclassified observations in SVMs
Epoch – number of passes through entire dataset during training
Gamma - regulates the amount of curvature in the decision boundary for SVMs
Kernel - transforms the input data to higher dimensions
Learning rate - Controls the size of the updates made to neural network parameters
Momentum – Adds inertia to gradient descent algorithm to "escape" from local optima in error domain
Weight Decay - Penalizes the loss function in neural networks to reduce changes made during updates