CM4107 CW Part 1: Comparative Evaluation

Liam Cottrell

Student Identification Number: 1302859

School of Computing Science and Digital Media

RGU

Aberdeen, UK

*Abstract*— Investigate the relationship between the hyperparameters of ANN and K-NN models and their effect on prediction accuracy. We are interested to see if larger epoch values in ANN always lead to improved accuracy and weather a hybrid model could exist that takes the best elements of both methods of classification. Taking the hidden layer output of a model and using that as our training set in a K-NN model has shown us improvement in model accuracy. The hyperparameters play a key role and seem to be unique to a dataset and its qualities.

# Comparative Study setup

# The purpose of our research is to investigate the advantages and differences of using different computational algorithms on a variety of datasets. We are interested in how adjusting the hyperparameters that these algorithms impact will affect the accuracy of their predictions.

# We proposed two datasets for our research due to their differences in features and characteristics. `MNIST` is a dataset involving data in the form of single handwritten numbers from the range of `0` to `9`. The ‘SONAR’ dataset consists of data received from sonar from samples of Rocks and Metal Cylinders. Having datasets with different styles of makeup should help us to evaluate the differences between our proposed algorithms, K-NN and ANN. They will also allow us to provide a more accurate and consistent analysis when we come to manipulate the variety of hyperparameters that we can change in the training stages.

The MNIST dataset’s main feature is its large quantity of data and abundance of unique features. On inspection of the data, the MNIST sample we are using has `4999` entries and `784` unique features. MNIST'S makeup of features are the intensity values, from a range of `0` to `255`, of each pixel of a 28x28 pixel image. MNIST enables us to allow both of our algorithms the ability to utilise a large training set of data to learn from and propose `10` possible output nodes, which was why it was one of our candidates. It would be interesting to observe how these algorithms dealt with a significantly larger number of predictive factors and whether this affected the accuracy of these predictions.

In comparison, the ‘SONAR’ dataset has `208` entries and `60` unique features. The makeup of SONAR’s features are the normalised energy values reflected and sensed within a particular frequency band, over a period of time. The energy values were captured after a frequency-modulated chirp, rising in frequency was performed. The ‘SONAR’ dataset provides insight into both the reactions of our algorithms to smaller quantities of data and features, and their capability to cope with smaller output node options. Due to the differing in makeup to ‘MNIST’, ‘SONAR’ was an excellent comparison to be able to compliment our findings. Another useful comparison that is present in our evaluation is the time it takes to train each network when faced with datasets of varying sizes.

A consistant ratio of training and testing subsets were set for all investigations. In order to give our models an ample amount of training data, a ratio of 80:20 between training and testing data entries were applied. A larger training sample allowed by this ratio would increases the accuracy of the models (Torralba, Fergus and Freeman, no date). The advantage of a more substantial training set of data is also apparent when looking at smaller datasets, such as Sonar. If the ratios were equal, such as 50:50 between training and testing, we would only provide `104` entries for both sets which may not prove to be enough entries to provide adequate training for our models.

We perform two different methods of predictive analysis with our datasets, ANN (Artifical Neural Network) and K-NN (K-Nearest Neighbour). Artificial Neural Networks are an electronic representation of the human nervous system and the large cluster of interconnected neurons (Zhang, 2018). The purpose for their conception originates from trying to replicate the way humans make decisions, by obtaining inputs, processing them, then producing an output (Zhang, 2018). ANNs have been widely used to help study and predict topics such as climate change trends (Zhang, 2018). K-Nearest Neighbour is a process more centred around providing a prediction due to the probability of similarly presented data that it has learnt (Keller and Gray, 1985). It makes its decision based on the most similar entries of data provided by the training set, which it then uses to provide its classification (Keller and Gray, 1985).

# Neural Network

## Neural Network Hyperparameters

## If our ambition is to create applications that allow us to make a judgement to an equal or better degree than a human, it makes sense to replicate how humans think. Humans are the most intelligent lifeforms on earth and are an excellent basis to build upon (Guilford, 1967). The idea of Neural Networks invoked this desire to replicate human decision. The key to complete artificial human intelligence is still a question that has not been fully achieved, but the principles of decision making in humans does work as a basis of predictive analysis. The Neural Network is effectively a computational graph, which is a hierarchy of nodes that are interconnected. Nodes on the same hierarchy combine their values, and their results become the value of the next node in the hierarchy. The number of hierarchical layers is configurable, where they all feed into a final output layer. The result of this layer becomes our prediction and can be fed back into the input to be reiterated to reduce errors in our predictions. The layers between our input and output, referred to as hidden layers, are present to give us a better representation of our data.

We tune our networks with certain hyperparameters that allow us to manipulate the decision-making process, to make our models more accurate, and less prone to errors. These play a significant part in the final prediction as they influence the weights and biases involved in the combination of nodes and therfore the results. The final result of this process is to create a mathematical function that is the best fit to allow us to put input our data for accurate predictions.

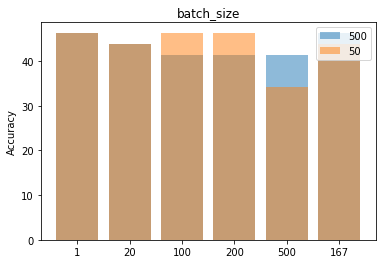
The epoch value is an adjustable period of time that can influence model accuracy. In neural networks, this epoch dictates the number of iterations that we cycle through our network, passing the result of our output nodes back to the input. Large epoch values can lead to a model that learns throughout multiple repetitions, improving model accuracy. However, in some cases this can lead our model to the wrong answers, and the resultant error influences our model to be less accurate over a more substantial number of iterations referred to as overfitting. Overfitting occurs when the model is too specific and loses the ability to understand data outside of its training set effectively. However, there needs to be a balance with the epoch value, as if the model is not subjected to a sufficient amount of reiterations and inaccurate underfitting may occur. The epoch value is particular to the dataset and needs to be tuned to get the best results.

Batch size is the number of entries of our data that we want to send into our network at every epoch. The batch size can be equal to the value of `one` which performs a process in our model called Stochastic Gradient Descent. In Stochastic Gradient Descent, single entries from our dataset are chosen at random and placed into the model’s input with an aim to make every epoch iteration less impactful. Stochastic Gradient Descent allows us to have a slow change in the mathematical function in every epoch to get to our final probability matrix in small granular steps. If the batch size is larger than one and smaller than the number of records in a training set, then this is called Mini Batch Gradient Descent. Very similar to Stochastic Gradient Descent, Mini Batch takes the value as the size of a sample of data placed as values into our input nodes at random. If the batch size is equal to the number of records in the training set, it is referred to as Batch Gradient Descent. The advantage of Batch Gradient Descent is that each value in the training set would have representation as an input node. A setback of Batch Gradient Descent is its computational intensity meaning that models take a long time to train, especially with more massive datasets.

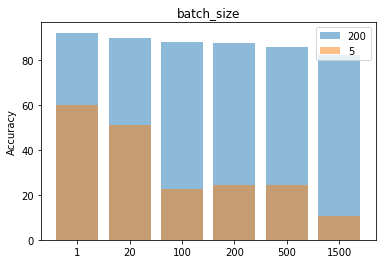
Learning Rate value dictates how our model’s output value is fed back into our input after every epoch. It is typically a value between zero and one as it is multiplied by the result of our output nodes weights to produce a fraction. The fraction of our output is then brought in as the weights for our input nodes for our next iteration.

The hidden layer node value can also be manipulated to help us with our model’s accuracy. To get a better representation of our data we sometimes want to decrease the amount of parameters to allow a more substantial degree of change between factors of our dataset affects the decision process our network. The opposite may be required in datasets with too many factors. Dimensionality reduction is the process of making the hidden layer node quantity less than the number of the input nodes. This reduction can allow the network to group up multiple factors that don't have a significant impact in when changed to our classification, to give them a more meaningful purpose.

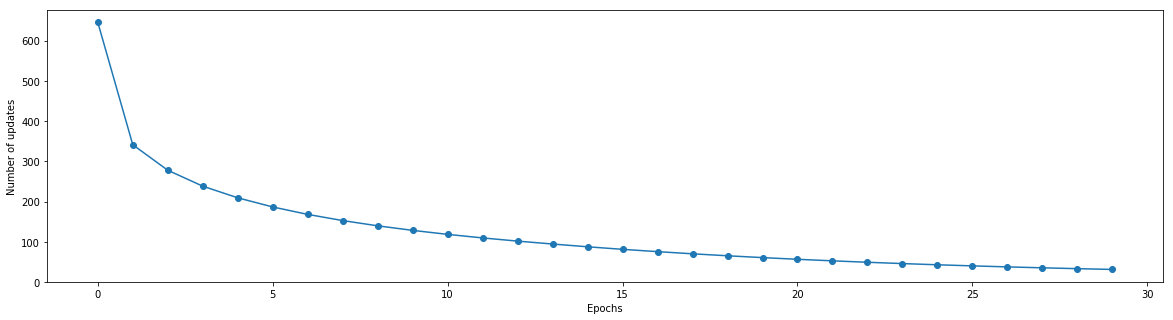
## Visualisation of Results



*Sonar – Batch size & epoch values effect on Accuracy.*

**

*MNIST – Batch size & epoch values effect on Accuracy*

**

*MNIST – Epoch values effect on model error*

## Discussion of Results

### Sonar produced interesting results when epoch values were changed. For the majority of tests, it was found for this dataset that the epoch value did not play a significant role in model accuracy. The epoch values of `5` and `500` were used to guage the differencee between a larger and small epoch number. Batch size appeared to have a larger impact, influencing the accuracy ± 5% from the extremes of `1` to the value `500`. What was interesting to see was the impact of a small epoch on the MNIST dataset which dramatically reduced the accuracy ±60%. It was evaluated that for larger featured datasets, having a larger epoch value to allow the model to build upon, positively impacted accuracy of prediction. It was observed for both models that a higher quantity of epochs reduced the number of errors in the prediction, which did show that the models did learn when given time to reiterate over the test data.

# k-nearest neighbour

## k-NN Hyperparameters

## The K-Nearest Neighbour approach differs from ANN as it looks at the classification problem with more simplistically. In its purest form, when making a prediction, K-NN looks at features between the candidate row of the dataset and compares them in similarity to the features of the training dataset. After performing a similarity metric between that row and the training datasets features, it will decide what the closest rows are between the candidate and the training dataset. To make its prediction, it then looks at the classification of the closest neighbouring entries in the training set and determines that the classification must be equal to that by performing a voting process. The winning classification is determined by the factor with the majority vote.

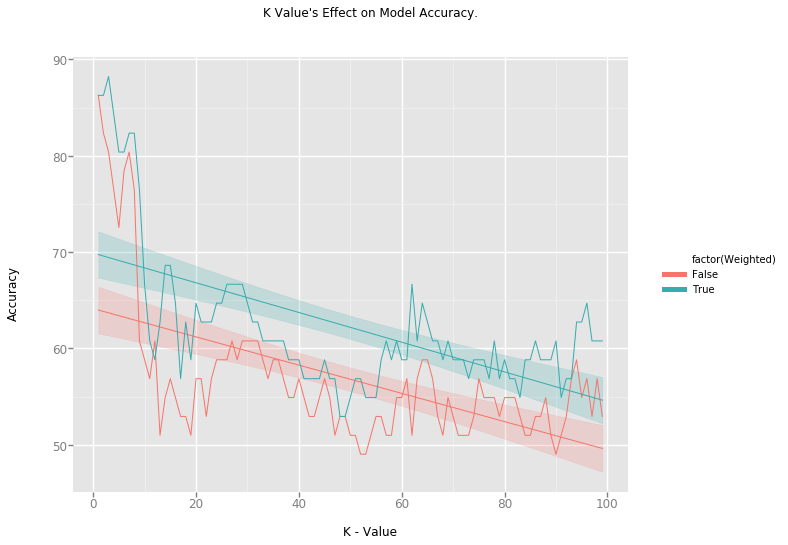
Some important factors can our results such as, how many entries in the training set determine the end classification. The K-Value is the value of closest entries that are considered for prediction. To aid the prediction process, the K-Value should be an odd number as a majority will be deduced when voting. The value that K needs to be in a dataset is a process of parameter tuning and is very specific depending on the dataset that it is using. The consequences of choosing a K value that is too low results in an unwanted bias that may be due to background noise and small clusters of groups in the dataset. If the K value is too large, it has consequences in computational intensity and also can lead to issues where the result may bias a classification where there happen to be more records for in the dataset. A good starting point for K is to take the square root of the sum of entries in the dataset (Hassanat, Abbadi and Alhasanat, 2014) . If this value is even, then either add or subtract one from the value. The answer for what the best method of determining a K value for a particular dataset is trial an error however as the best K value for a given dataset is almost impossible to be determined (Hassanat, Abbadi and Alhasanat, 2014).

How we deduce the nearest neighbour in similarity to our candidate is by using Similarity metrics. The Euclidian distance is the most popular measurement used for K-NN (Kurchaniya and Johari, 2017). The Euclidean's function performs a distance calculation in the difference between x and y positions between two objects. The x and y position will be deduced from two features of our dataset. This distance is what we then use to find the nearest neighbours to our candidate. There are other choices of similarity measurements that can be used in place of Euclidean's. Manhattan Distance is an alternative that uses the sum of the difference between the two features. Manhattan has been documented to provide a higher percentage of accuracy with datasets with a more significant number of factors (Kurchaniya and Johari, 2017).

To help improve our predictions, we can make K-NN look into additional features of our dataset. If our record that we are comparing with our dataset happened to be particularly close in Euclidian distance to a particular classification, then this should be weighted as a bias. This weighted bias may be used to help improve K-NNs' ability to perform classification. Even if the amount of votes is high for a specific classification, the classifications that have the closest distance may be closer to the correct prediction. Weighted predictions do not tend to perform well with datasets with a high quantity of classifications (Hechenbichler, 2004) and are more tailored to datasets with a small number of predictable factors.

## Visualisation of Results

*SONAR*



*MNIST*

**

## Discussion of Results

From our findings in both datasets, increasing the K-Value decreases the models ability to make correct predictions. Both Datasets show to have a sweet spot which is a K-Value between `1` and `5` where accuracy seems to remain consistantly high. What is interesting to see when observing the SONAR datset is the deviation of accuracy from K-Values between `20` to `100`. The accuracy of our model fluctuates around 50% to 60% suggesting that we have reached a point where manipulating this value any higher would not make any diffrence to the accuracy. Allowing the closeness of our nearest neighbours, by allowing weighting showed to have a positive effect on model accuracy of both datasets. When paired with the K-Value, it was observed that weighting had impact with smaller K-Values, but was shown to mostly improve K-Values in higher ranges. Using weighting when trying to evaluate a good value for K would prove to be a bad idea as the weighting would skew you ability to see if increasing K was having a negitive impact on classification. When an appropriate K-Value is found, weighting seems to have a positive impact in improving accuracy.

# Hybrid

## Combining ANN and k-NN

## The purpose of the hidden layer in ANN is to give the model a better representation of our data before being trained and re-iterated. It would, therefore, be beneficial if we performed the first stages of our ANN, and then used the output of our hidden layer. We would take the hidden layer output with no weights applied and use that as our dataset in KNN. The ANNs' input would become our training data variable, and its features would be scaled to the value of hidden layer nodes configured. In large datasets with a large number of features, the ability to perform dimensionality reduction on the amount of features would improve the time it would take a computer to train a K-NN model. The output layer would act as an improved method of normalisation and could lead to a more accurate classification and reduce variables with little impact to classification.

## Visualisation of Results

|  |  |  |
| --- | --- | --- |
| Dataset | Method | Accuracy |
| MNIST | ANN | 90.34 |
| MNIST | K-NN | 85.84 |
| MNIST | `Hybrid` | 92.72 |

## Discussion of Results

For both models, my best-known settings were used to perform a best case scenario result. I found that for our K-Value, the best results were had using a low and odd K-Value, so we decided to use 3 for our example. We set out hidden node value of our KNN at `200` as model accuracy in the previous testing indicated that the reduction helped improve our predictions. As we can see from the results, processing MNIST via a Hybrid model produces a result with higher accuracy than one model on their own. The research suggests that this method of classification works well with a dataset that have multiple features. The reasoning may be due to these features, not all being equally weighted in the final classification. The use of Dimension reduction from `784` features down to 200 features will have given a clearer picture for K-NN to make a classification with. If given more time, I would like to implement this method with the sonar dataset. I feel the relationship of increasing the number of features and the effect this would have in K-NNs ability to be accurate would be interesting.

##### References

Guilford, J. P. (1967) *The nature of human intelligence.*, *The nature of human intelligence.* New York, NY, US: McGraw-Hill.

Hassanat, A. B., Abbadi, M. A. and Alhasanat, A. A. (2014) ‘Solving the Problem of the K Parameter in the KNN Classifier Using an Ensemble Learning Approach’, *International Journal of Computer Science and Information Security*, 12(8), pp. 33–39. doi: 10.1007/s00500-005-0503-y.

Hechenbichler, K. (2004) ‘Hechenbichler, Schliep: Weighted k-Nearest-Neighbor Techniques and Ordinal Classification’, 399.

Keller, J. M. and Gray, M. R. (1985) ‘A Fuzzy K-Nearest Neighbor Algorithm’, *IEEE Transactions on Systems, Man and Cybernetics*, SMC-15(4), pp. 580–585. doi: 10.1109/TSMC.1985.6313426.

Kurchaniya, D. and Johari, P. K. (2017) ‘Analysis of Different Similarity Measures in Image Retrieval Based on Texture and Shape’, *International Research Journal of Engineering and Technology(IRJET)*, 4(4), pp. 2137–2143. Available at: https://www.irjet.net/archives/V4/i4/IRJET-V4I4549.pdf.

Torralba, A., Fergus, R. and Freeman, W. T. (no date) ‘80million tiny Images.pdf’, pp. 1–12. doi: 10.1109/TPAMI.2008.128.

Zhang, Z. (2018) *Multivariate Time Series Analysis in Climate and Environmental Research*. doi: 10.1007/978-3-319-67340-0.