# Approaches

1. Classification of numbers from 0 to 9 in a sequence based on Sparse Distributed Representations (SDRs)

At the start, after investigating the KNN Classifier and how it runs on C#, we develop a KNN Algorithm evaluation using random unclassified features and known features with label classes Then we explore the Neocortex repository which we fork, understand how it generates SDR’s based on the sequence, and how can we used to classify the value in the sequence based on these SDR’s Value. At the start we gave a sequence of 0 to 9 to sequence the learning experiment and extract the SDRs of sequence 0 to 9 then we labeled the SDRs with the class 0 to 9 respectively as the last value of the array. The classified SDR data is given below.

data[0] = { 7, 18, 24, 29,……, 1012, 0 };

data[1] = { 25, 31, 44, 48,…… ,188, 1 };

data[2] = { 118, 123, 127, 156,………, 340, 2 };

data[3] = { 240, 242, 257, 266,………., 444, 3 };

data[4] = { 302, 314, 324, 327,……., 518, 4 };

data[5] = { 393, 405, 428, 429,…….., 624, 5 };

data[6] = { 483, 487, 500, 509,………, 726, 6 };

data[7] = { 579, 587, 595, 607,…………, 814, 7 };

data[8] = { 676, 691, 700, 707,……… , 916, 8 };

data[9] = { 772, 779, 780, 800,…………., 1007, 9 };

We train the model based on these classified SDR’s, then evaluate the model using the unclassified data as given below.

Unclassified = {461, 495, 515, 501,…, 712 }

The model successfully classified the unclassified data using KNN classifier and labeled the unclassified data with class 6 as the data point of unclassified data resembles closely with the SDR’s having class label as class 6 in classified training data. The value of K is 1 in this case, The reason for using is that the training dataset is so limited, every data in the training dataset has a separate class so the algorithm should only consider the first nearest neighbor, as classify the unclassified dataset.

With this approach, we can classify the value of the sequence as instructed, but we want to solve some real-life problems out of this, so after some research, we came to an idea that we want to be executed with this approach but in a much better way. The idea was to predict or identify the whole sequence that belongs to sets of numbers. This approach is being discussed in the next section.

1. Classification of different Sets of Numbers sequences based on Sparse Distributed Representations (SDRs)

To implement this approach, we generate SDR from different sequences S1, S2, and S3, S1 consider is of even numbers, S2 as odd number, and S3 as decimal number which are Neither odd nor even. The SDR’s are extracted with multi sequence learning experiment. At first, we just take one SDR’s for each sequence and use it to unclassified sequence which contain most of the elements from one of sets, with this we can classify the unclassified sequence. To make the model more efficient we must increase the dataset. To increase the dataset, we generate SDR 11 times of the same sequences. 11 time is not fixed but we just increase the dataset that many generations. Then we collect this data first in a text file. We validate the model by splitting the dataset in 70- 30 splitting ratio. 70% of the dataset in the text file is used to train the model while 30% remains to evaluate the model performance and prediction. The algorithm randomly splits the data to evaluate the model performance unbiasedly, each testing data is being classified based on the train training dataset. The model predicts the class label of every test data almost every time accurately.

To improve the structure of data we came to this idea that its better to have dataset in JSON file rather than text file. In text file we label the SDR’s with 0, 1, and 2, 0 consider be S1 or even, 1 consider to be S2 or odd and 2 consider to be as Neither odd or even which we evaluate when we are generating the SDR’s but for us it was not consider to be a great approach so we change file type to JSON which help us to read data in more structure and reliable way. The sample dataset in JSON file we created is shown down below.

.

[

{

"SequenceName": "S1",

"SequenceData": [8039, 8738, 9334, 9558, 9604, 9697, 9772, 9841, 9851, 9922, 9963, 10023, 10121, 10197, 10373, 10459, 10594, 10629, 10664, 11124]

},

{

"SequenceName": "S2",

"SequenceData": [9051, 9075, 9133, 9178, 9365, 9448, 9481, 9599, 9635, 9740, 10032, 10224, 10281, 10762, 10778, 10934, 11143, 11306, 11494, 11763]

},

{

"SequenceName": "S3",

"SequenceData": [10808, 10834, 11053, 11085, 11434, 11471, 11479, 11553, 11597, 11634, 11720, 11743, 11766, 11812, 11872, 11897, 11909, 12094, 12332, 12504]

}, ...

]

Using this we can map SDR’s to sequences exactly. To consider optimal value of K, we cross validate the value of K for different test dataset which is define and a cross validation table is described in result section.

# Results:

1. Classification of numbers from 0 to 9 in a sequence based on Sparse Distributed Representations (SDRs)

A screenshot of a computer

Description automatically generated

Figure 7. Illustration the output window of Class label of unclassified data on different value of K

As we discussed the whole scenario in the above portion. Figure 7 is the result of the classifier as it is predicting the class label for unclassified data for different value of K. As we can see for the value of k = 1 the classifier is predicting the class accurately because obviously the data point of the unclassified data is closed to the SDR’s label as 6 but as we are increasing the value of K, the algorithm considering more nearest neighbor make the model not predicting the class label correctly.

1. Classifying Sets of Numbers based on SDR’s Value

In our scenario, we utilize a data splitting method that randomly shuffles the dataset’s rows, allocating a specified ratio for training and the remainder for testing. Specifically, 70% of the data is allocated for training, while 30% is reserved for testing. The method returns the training and testing datasets as separate Lists. The reported accuracy values represent the model’s prediction accuracy at various values of k when tested with randomly generated test data. These accuracy percentages provide insights into the performance of our model under different configurations of the k-Nearest Neighbors algorithm.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| K | Accuracy | Random Generated Test Data Accuracy in Percentage | | | | | | |
|  | (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) |
| 1 | 100 | 90.9 | 90.9 | 90.9 | 90.9 | 90.9 | 90.9 | 100 |
| 2 | 100 | 100 | 90.9 | 90.9 | 90.9 | 90.9 | 90.9 | 100 |
| 3 | 100 | 100 | 100 | 100 | 100 | 90.9 | 90.9 | 100 |
| 4 | 90.9 | 100 | 100 | 90.9 | 90.9 | 90.9 | 90.9 | 100 |
| 5 | 100 | 100 | 100 | 90.9 | 100 | 90.9 | 90.9 | 100 |
|  |  | | | | | | | |

Table 1: Accuracy of the KNN Classifier for Different Values of K for Different Testing Data

The KNN cross validation is presented in Table 1 on different random generated testing data, indicate that the most desirable value of k for the k-Nearest Neighbors (k-NN) classifier, when applied to a dataset with three classes with the dataset we have, is k=3. The value of K can vary if dataset is being increase. The value of K and respective accuray is explained down below:

1) K=1, 2, and 4: When the model relies solely on the single nearest neighbor (k=1) or a small number of nearest neighbors (k=2, 4), the accuracy tends to hover around 90.9%. This indicates that the classifier may be prone to misclassifications or inconsistencies when considering a small number of neighbors, leading to accuracy fluctuations.

2) K=3: At k=3, the accuracy consistently remains around 100% for most randomly generated testing data splits from the dataset. This suggests that considering the three nearest neighbors leads to more stable and reliable predictions, resulting in higher accuracy across various testing scenarios.

3) K=5: While the accuracy at k=5 is also reported as 100%, it is noted that there are instances where the accuracy drops to 90.9%. This indicates some variability in performance compared to k=3, where the accuracy remains consistently high.

Based on these observations, it’s reasonable to conclude that k=3 appears to be the optimal choice for the k-NN classifier with this dataset.

A screenshot of a computer

Description automatically generatedA screenshot of a computer program

Description automatically generated

Figure 8 and 9 Output Window presenting the predicted result for test data with accuracy around 90 %.

Figures 8 & 9 showcase the output window presenting the predicted results with great accuracy around 90% during the random splitting test data. In the figures, the nearest neighbors for the test data point is being displayed with a value of k set to 3, along with the calculated distances and the classification of the class for that specific test data.

Subsequently, the predicted class for the test data is indicated, determining whether it falls in class label S1, S2, or S3 where the S1 is even, S2 is odd and S3 is neither odd or even based on the voting method used for predicting the sequence. This process repeats for each test data point. Finally, the model’s predicted accuracy 90.9% for certain test data is displayed at the end.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

Figure 10 and 11 Output Window presenting the predicted result for test data with accuracy around 100 %.

Figures 10 & 11 display the output window presenting the predicted results with impressive accuracy around 100% during random splitting test data. Mean all the sequence have been match perfectly. The figure 10 & 11 are mirror as in in Figure 8 and 9 but having different accuracy. The exceptional accuracy is because when the algorithm split data, the data point of the test data in the dataset is align nearest to the data point in the training data, achieving the accuracy to be near to perfect, resulting matching all the sequences.

A screenshot of a computer

Description automatically generated

Figure 10. KNN Unit Test

Figure 10, depicts the unit test conducted on the KNN classifier. A random SDR is selected from the dataset, serving as the test data list to evaluate the classifier’s performance. The test ensures that the predicted value by the classifier aligns with the actual class value of the test data. The KNN classifier successfully passes the unit test, as evident from the figure. Additionally, we’ve incorporated an exception in the unit test to accommodate varying values of K. If the value of K surpasses the length of the SDR data, the test gracefully handles this scenario. Second test case has been handle when unknown feature or testing features are null , last case handle when training features are null.