Assignment 2 - Report

Sophia J Quinton

Grand Canyon University

DSC-540: Machine Learning

Dr. Aiman Darwiche

10 November 2021

Assignment 1 - Report

Part 1: Theory

Confusion Matrix

In general, a confusion matrix is used to measure how accurate a model is to predict the correct results. This is important because there are times when misclassification error is not suited well for problems with a skewed distribution of data (Gopal, 2019). The confusion matrix will classify data from a model as a false positive, a true positive, a false negative, and a true negative. From there, costs can be assigned to the overall matrix and applied to the model. The confusion matrix "gives you insight not only into the errors being made by your classifier but more importantly the types of errors that are being made" (Brownlee, 2020).

In terms of a pattern recognition task, the confusion matrix will help classify the correct images even if the data is skewed. "Any performance metric that uses values from both columns will be inherently sensitive to class skews" (Fawcett, 2006). From here, the data can be used to create a binary classifier based on the true positives and the true negatives. Overall, the confusion matrix provides a way to help classify images if portions of the image are difficult to detect.

An example of an example would be a journal article by Sun et. al. (2018). In the article they are trying to recognize the pattern of hand gestures using machine learning. They apply the confusion matrix to understand the issues with the pattern recognition model. They then apply it to a probability model to correct this error.

ROC Curve

The ROC curve is complementary to the confusion matrix. An ROC graph shows "shows relative trade-offs between advantages (true positives) and costs (false positives)" (Gopal, 2019).

To graph an ROC curve, one can use the confusion matrix and by "thresholding the classifier with respect to its complexity" (Gopal, 2019). The graph it self will plot the points of fp,tp on the graph. An ideal point is (0,1) as that shows no false positives but all true positives. To compare multiple classifiers, one can plot them all in one ROC graph. To determine the best classifier from plotting them all on the same graph, one can look at the line that is closest to the (0,1) point.

An example of a successful plot would be in a study by Xu et. al. (2021) where they used an ROC to analyze their multi-omics model predicting tumor mutation burden. They used the ROC to compare the use of different variables in their model like age, gender, and risk scores. Overall, ROC curves can be applied to multiple fields.

Part 2: Application

Introduction

The k-nearest neighbor algorithm is a supervised learning classification algorithm. It "does not make assumptions about the model between the class membership (y) and the features" (Gopal, 2019). It requires the user to define the number of groups that the algorithm will short, but this can be done by iterating through multiple k values or by using an elbow plot to find the inflection point. The k-nearest neighbor clustering model is non-parametric as it "finds similar past patterns or instances from the training set" (Gopal, 2019). This is useful when pattern recognition is the choice for the model. The model follows a denisity based classification method that is useful when trying to determine the groups. The model for the k-nearest neighbor is: $\hat{p}(x) = \frac{k}{2Nd_k(x)}$ The larger the sample the better the performance (Gopal, 2019). The distances between the points are "arranged in ascending order from x to the points in the sample" (Gopal, 2019). In the case of multivariate data (like the hand writing classification set) the density

probability is given by: $\hat{p}(x) = \frac{k}{NV_k(x)}$ (Gopal, 2019). The V represents the vector of the different classes in the dataset.

Because this dataset (LeCun et. al., 1998) is based on pattern recognition, the k-nearest neighbor classification would work well with this dataset.

Methods

First, to understand this dataset that is based on Euclidean distance, an algorithm was generated to test the distances of a few points. The points that were excepted was two points in the training set, two points in the testing set, and two points between the training and testing datasets. This can be seen in Figure 1.

```
In [3]: ify the variables in the dataset and define Euclidean distance
        en an element in test and training
         labels represent the Y data and the images are the X variables
        lley, 2020) (Brownlee, 2020)
        ath import sqrt
        clidean_dist(imA, imB):
        ased on the pythagorean theorem
        stance = 0.0
        r i in range(len(imA)-1):
          distance += (imA[i] - imB[i]) **2
        turn sqrt(distance)
        ulate the difference between two images example
        _array = np.array(images)
        _test_array = np.array(images_test)
        "Example of calculating the euclidean Distance of two elements training set: ", euclidean_dist(images_array[0], images_array[2]))
        "Example of calculating the euclidean Distance of two elements testing set: ", euclidean_dist(images_test_array[0], images_test_ar
        "Example of calculating the euclidean Distance of two elements from each: ", euclidean_dist(images_array[5], images_test_array[5])

Example of calculating the euclidean Distance of two elements training set:  2773.149112471235
Example of calculating the euclidean Distance of two elements testing set:  2176.560819274297
Example of calculating the euclidean Distance of two elements from each:  2509.1213601577742
```

Figure 1: Euclidean Distance of train and test point

The dataset downloaded as separate .gz files was decompressed and loaded appropriately based on a special mnist package. The test and train datasets were already pre-determined before downloading the data, as they were already set. The training set has 60,000 images and the testing set has 10,000 images. The variables in the dataset are the ‘labels’ which represent the true number value or the response variable. The ‘images’ are vectors that contain classification data. These are the predictor variables. The sklearn kNeighborsClassifer (sklearn, nd) was used to create the classifier based on the dataset. The k was set to 10 as there is a total of 10 numbers.

An elbow plot or iteration was not used to determine this because the numbers are already set (LeCun et. al., 1998) (Babu et. al., 2014).

To find the distances between the above test elements that were chose and each of the k-nearest neighbors, a function in the sklearn package was used. K-neighbors will show the distances to all the different groups. This is shown in Figure 2. The smaller the number pair, the nearest location to that group. Groups increase from 0 to 9. For example, the first neighbor is in the group of 0's because it is right on that centroid (0,0).

```
In [5]: #calculate the distance between the test elements and each of neighbors
        ##test1
        print("Train to neighbors: ", kModel.kneighbors([images_array[0]]))
        ##test2
        print("Test to neighbors: ", kModel.kneighbors([images_test_array[2]]))
        ##test3
        print("Test to neighbors: ", kModel.kneighbors([images_test_array[2]]))

Train to neighbors:  (array([[   0.        , 1561.47238208, 1591.60139482, 1594.71909752,
        1596.70942879, 1604.44694521, 1604.887535  , 1605.70918911,
        1609.69438093, 1613.70474375]]), array([[    0, 32248,  8728, 18932, 30483, 24149, 42338, 52295, 26251,
        50173]], dtype=int64))
Test to neighbors:  (array([[321.66286699, 332.46353183, 341.04838366, 367.71456321,
        377.33009421, 416.66533333, 429.1340117 , 431.8471952 ,
        439.3529333 , 442.75726984]]), array([[58741, 46512, 15224, 47333, 44038, 42531, 39364, 53361, 12578,
        27684]], dtype=int64))
Test to neighbors:  (array([[321.66286699, 332.46353183, 341.04838366, 367.71456321,
        377.33009421, 416.66533333, 429.1340117 , 431.8471952 ,
        439.3529333 , 442.75726984]]), array([[58741, 46512, 15224, 47333, 44038, 42531, 39364, 53361, 12578,
        27684]], dtype=int64))
```

Figure 2: Distance of test element and neighbors

```
In [8]: for i in range(len(train_counter)):
            print("The number of ", i, "'s", train_counter[i])

The number of  0 's 5923
The number of  1 's 6742
The number of  2 's 5958
The number of  3 's 6131
The number of  4 's 5842
The number of  5 's 5421
The number of  6 's 5918
The number of  7 's 6265
The number of  8 's 5851
The number of  9 's 5949

In [9]: ##find max
        def find_max(list):
            max_num = max(list)
            for i in range(len(list)):
                if max_num == list[i]:
                    print("The most popular number is ", i)
        find_max(train_counter)

The most popular number is  1
```

Figure 3: Popular number in train set

The last preprocessing task was to determine the distribution of the numbers in the training set. That was determined by self-made algorithm that iterated through the training results and calculated the different groups. For the training set, the most popular number was 1. This can be seen in figure 3. Then another analysis was done to confirm what type of number each k group represents. The

numbers show that group 1 contains the zero numbers, group 2 contains the one numbers, and so on.

After all the pre-processing and exploration of the data, the testing dataset can be applied.

```
In [13]: #identify the test element as the digit as popular
         predictions = kModel.predict(images_test)

In [14]: predictions
Out[14]: array([7, 2, 1, ..., 4, 5, 6], dtype=uint8)

In [15]: find_max(count_occurence(predictions))
         The most popular number is  1
```

Figure 5: classification of test data

This was done by sklearn's predict function (n.d.). I also applied the self-generated algorithms to the predictions, and the most popular number was one like the training set.

Lastly a confusion matrix, classification report, and ROC curve was generated (Figures). The confusion matrix is used to determine how well a model preformed. The results of the confusion matrix were pulled to determine the true positives, the false positives, the true negatives, and the false negatives (Loukas, 2020). The confusion

```
In [17]: table_result = pd.crosstab(np.array(labels_test), predictions, rownames = ['Actual'], colnames = ['Predicted'])
         table_result['Total'] = table_result.sum(axis=1); table_result.loc['Total'] = table_result.sum()
         table_result
```

Out[17]:

Predicted	0	1	2	3	4	5	6	7	8	9	Total
Actual											
0	972	1	1	0	0	2	3	1	0	0	980
1	0	1132	2	0	0	0	1	0	0	0	1135
2	13	12	982	2	1	0	2	17	3	0	1032
3	0	3	3	976	1	10	1	7	6	3	1010
4	2	11	0	0	940	0	4	1	1	23	982
5	4	0	0	12	1	863	6	1	1	4	892
6	6	4	0	0	3	2	943	0	0	0	958
7	0	27	4	0	2	0	0	983	0	12	1028
8	6	4	5	11	7	9	4	7	914	7	974
9	7	6	3	7	10	3	1	10	2	960	1009
Total	1010	1200	1000	1008	965	889	965	1027	927	1009	10000

Figure 6: confusion matrix

matrix here reveals a strong and successful classification. The results generated in a confusion matrix can then be applied to the results in the classification report (sklearn-confusion_matrix, nd) (sklearn-classification_report, nd). The classification report will determine the accuracy, precision, recall and f1-score for each of the classes. From the results the model had an overall

score of 97%. For most of the classes, the precision, recall, and f1-scores are above 95%. This indicates a strong model that successfully was able to predict the results.

```
In [20]: ##(Loukas, 2020)
         FP = conf_matrix.sum(axis=0) - np.diag(conf_matrix)
         FN = conf_matrix.sum(axis=1) - np.diag(conf_matrix)
         TP = np.diag(conf_matrix)
         TN = conf_matrix.sum() - (FP + FN + TP)
         print("FP: ", FP)
         print("FN: ", FN)
         print("TP: ", TP)
         print("TN: ", TN)

         FP:  [38 68 18 32 25 26 22 44 13 49]
         FN:  [ 8  3 50 34 42 29 15 45 60 49]
         TP:  [ 972 1132  982  976  940  863  943  983  914  960]
         TN:  [8982 8797 8950 8958 8993 9082 9020 8928 9013 8942]
```

Figure 7: Model Summary results

```
In [18]: print(classification_report(np.array(labels_test), predictions))
```

	precision	recall	f1-score	support
0	0.96	0.99	0.98	980
1	0.94	1.00	0.97	1135
2	0.98	0.95	0.97	1032
3	0.97	0.97	0.97	1010
4	0.97	0.96	0.97	982
5	0.97	0.97	0.97	892
6	0.98	0.98	0.98	958
7	0.96	0.96	0.96	1028
8	0.99	0.94	0.96	974
9	0.95	0.95	0.95	1009
accuracy			0.97	10000
macro avg	0.97	0.97	0.97	10000
weighted avg	0.97	0.97	0.97	10000

```
In [19]: scores_y = kModel.score(np.array(images_test), np.array(labels_test))
         scores_y

Out[19]: 0.9665
```

Figure 8: Classification report

The last model summary report is given in the ROC curve. The ROC curve for a multiclass model requires a line to be determined for each class (sklearn-ROC, nd). In Figure 9, this is hard to tell because the lines fall on top of them. A successful model will have the cure near point (0,1) (Gopal, 2019). This is shown in the figure as it quickly jumps up to that value. There is

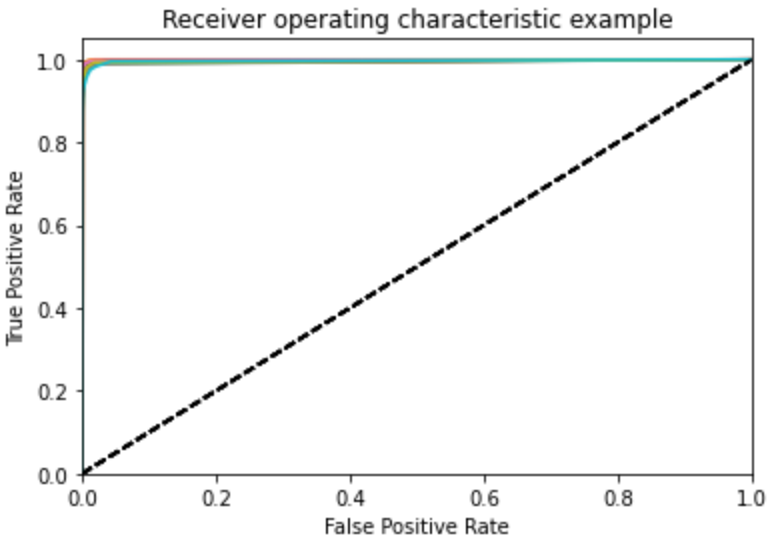

Figure 9: ROC curve

little to no false positives and many true positives. Again, this is another report for the amount of success of the value. The error of this algorithm can be determined by accuracy -1. The error is 3%.

Conclusions and Discussion

Due to the structure of the k-nearest neighbor algorithm, it is successful with pattern recognition tasks. Using this algorithm on the mnist dataset is successful at k=10 as there are ten different numbers. The most popular element was the number one in each of the datasets. The classification of the dataset was successfully determine when the model summary and errors were determined. The confusion matrix and the classification report together determined the values of each of the different classes. It then was applied to determine the accuracy, recall, and f-1 scores. All were above 95%. The ROC curve is a great illustration for the success of the model. It uses the TP and FP from the confusion matrix for each of the classes. It shows very little false positive rates.

Another application of this model might be useful in facial recognition because it is another type of pattern recognition model.

References

Babu, U. R., Venkateswarlu, Y., & Chintha, A. K. (2014). Handwritten digit recognition using K-nearest neighbour classifier. *2014 World Congress on Computing and Communication Technologies*. https://doi.org/10.1109/wccct.2014.7

Bhalley, R. (2020, April 3). *Digit recognition*. Medium. Retrieved November 10, 2021, from https://towardsdatascience.com/mnist-with-k-nearest-neighbors-8f6e7003fab7.

Brownlee, J. (2020, February 23). *Develop K-nearest neighbors in python from scratch*. Machine Learning Mastery. Retrieved November 10, 2021, from https://machinelearningmastery.com/tutorial-to-implement-k-nearest-neighbors-in-python-from-scratch/.

Brownlee, J. (2020, August 14). *What is a confusion matrix in machine learning*. Machine Learning Mastery. Retrieved November 9, 2021, from https://machinelearningmastery.com/confusion-matrix-machine-learning/.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, *27*(8), 861–874. https://doi.org/10.1016/j.patrec.2005.10.010

Laporte, L. (2016, November 4). *Extract images from .IDX3-ubyte file or gzip via python*. Stack Overflow. Retrieved November 10, 2021, from https://stackoverflow.com/questions/40427435/extract-images-from-idx3-ubyte-file-or-gzip-via-python.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998) <LBB+98>`. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86, 2278--2324.

Loukas, S. (2020, June 19). *Multi-class classification: Extracting Performance Metrics from the confusion matrix*. Medium. Retrieved November 10, 2021, from

https://towardsdatascience.com/multi-class-classification-extracting-performance-metrics-from-the-confusion-matrix-b379b427a872.

Gopal, M. (2019). *Applied machine learning*. McGraw-Hill Education.

sklearn. (n.d.). *Sklearn.neighbors.kneighborsclassifier*. scikit. Retrieved November 10, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html.

sklearn-classification_report. (n.d.). *Sklearn.metrics.classification_report*. scikit. Retrieved November 10, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html#sklearn.metrics.classification_report.

sklearn.confusion_matrix. (n.d.). *Sklearn.metrics.confusion_matrix*. scikit. Retrieved November 10, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html.

sklearn-ROC. (n.d.). *Receiver operating characteristic (ROC)*. scikit. Retrieved November 10, 2021, from https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html.

Sun, K., Feng, Z., Changsheng, A., Li, Y., Wei, Jun., Yang, X., Guo, X., Liu, H., Han, Y., Zhao, Y. (2018). An intelligent discovery and error correction algorithm for misunderstanding gesture based on probabilistic statistics model. *International Journal of Performability Engineering*. https://doi.org/10.23940/ijpe.18.01.p10.89100

Xu, Q., Xu, H., Deng, R., Wang, Z., Li, N., Qi, Z., Zhao, J., & Huang, W. (2021). Multi-omics analysis reveals prognostic value of tumor mutation burden in hepatocellular carcinoma. *Cancer Cell International, 21*(1). https://doi.org/10.1186/s12935-021-02049-w

Assignment2 Github: https://github.com/squinton-gcu/Data-Science/tree/main/DSC-540/Assignment2