ECE 521 Assignment 1

# Work Breakdown

|  |  |
| --- | --- |
| Group Member Name | Contribution Percentage |
| Fan Guo | 0% |
| Jeffrey Kirman | 50% |
| Connor Smith | 50% |

# Part 1: Euclidean distance

As stated in the assignment for input tensor and input tensor the Euclidean distance is

The **euclidean\_distance** (Appendix A) function evaluates this using vectorization. It first converts the input matrices into 3D tensors of shape and for input tensors and , respectively. These new tensors are subtracted from each other which broadcasts both vectors into the shape of a tensor before evaluation. This resultant tensor is then piecewise squared and all the elements on the length axis are summed together to result in the Euclidian distance matrix.

# Part 2: Regression

## Question 1: Choosing nearest neighbours

For a given input features vector and targets vector , let denote thek nearest neighbors as measured using the above **euclidean\_distance** function above and selected using the **tf.nn.top\_k** function. Then, the prediction function is defined as

Where is the responsibility vector defined as

The full code for calculating this responsibility vector is available in Appendix A – get\_responsibility\_matrix.py

## Question 2: Prediction

For the *data1D* dataset generated as instructed (Appendix B), the following mean squared error loss was calculated from the prediction function as follows:

The calculated MSE values for the Training, Validation and Test datasets for values of are recorded in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Training MSE Loss | Validation MSE Loss | Test MSE Loss |
| 1 | 0.0000 | 0.2716 | 0.1394 |
| 3 | 0.1065 | 0.3244 | 0.1588 |
| 5 | 0.1211 | 0.3167 | 0.1851 |
| 50 | 1.2460 | 1.2287 | 0.7026 |

TODO: plots

# Part 3: Making Predictions for Classification

## Question 1: Predicting Class Label

Using the code from Part 2 above, input features were sorted by Euclidean distance from all available training features and had the smallest selected. From these results, the most common associated label is returned as the predicted label for the new point. Code to accomplish this task is available in Appendix A.

## Question 2: Face recognition using k-NN

Using the given face dataset with the label (name) prediction code above and varying , the following accuracy results were obtained

|  |  |  |  |
| --- | --- | --- | --- |
|  | Training Accuracy | Validation Accuracy | Test Accuracy |
| 1 | 100% | 66.3% | 71.0% |
| 5 | 80.2% | 60.9% | 68.8% |
| 10 | 72.4% | 57.6% | 66.7% |
| 25 | 66.1% | 59.8% | 65.6% |
| 50 | 58.8% | 57.6% | 58.1% |
| 100 | 52.1% | 47.8% | 49.5% |
| 200 | 42.7% | 31.5% | 39.8% |

Using the value of which maximizes validation accuracy, the calculated test accuracy was 71.0%.

For the case, the images of an incorrect prediction with its 10 nearest neighbors is available in Appendix B.

## Question 3: Gender Recognition using k-NN

Repeating the above process for gender classification, the following accuracy results were obtained:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Training Accuracy | Validation Accuracy | Test Accuracy |
| 1 | 100% | 91.3% | 92.5% |
| 5 | 90.6% | 91.3% | 90.3% |
| 10 | 90.2% | 89.1% | 89.2% |
| 25 | 83.3% | 90.2% | 88.2% |
| 50 | 82.3% | 89.1% | 86.0% |
| 100 | 78.2% | 85.9% | 86.0% |
| 200 | 72.6% | 78.3% | 77.4% |

Using the value of which maximizes validation accuracy, the calculated test accuracy was 92.5%

For the k=10 case, the images of an incorrect prediction with its 10 nearest neighbors is available in Appendix C.

# Appendix A – Code

## euclidian\_distance.py

|  |
| --- |
| **import** tensorflow **as** tf  **def** **euclidean\_distance(**X**,** Z**):**  D **=** X**.**shape**[-**1**]**  X\_int **=** tf**.**reshape**(**X**,** **[-**1**,** 1**,** D**])**  Z\_int **=** tf**.**reshape**(**Z**,** **[**1**,** **-**1**,** D**])**  distance\_pairs **=** X\_int **-** Z\_int  eucl\_dist **=** tf**.**reduce\_sum**(**tf**.**square**(**distance\_pairs**),** **-**1**,** name**=**"euclidean\_distances"**)**  **return** eucl\_dist  **if** \_\_name\_\_ **==** '\_\_main\_\_'**:**  session **=** tf**.**InteractiveSession**()**  X **=** tf**.**constant**([[**1**,**2**,**3**],** **[**4**,**5**,**6**]])**  Z **=** tf**.**constant**([[**7**,**8**,**9**],** **[**1**,**2**,**3**]])**  expected\_result **=** tf**.**constant**([[**108**,** 0**],** **[**27**,** 27**]])**  tf**.**assert\_equal**(**euclidean\_distance**(**X**,**Z**),** expected\_result**)**  **print(**session**.**run**(**euclidean\_distance**(**X**,**Z**)))** |