ECE521A1 Report

Wenjia Liu - 1000522589
Jiashen Wang - 1001259568
Seunghyun Cho - 1001681325

February 1, 2018

1 Euclidean distance function

Include the snippets of the Python code.

```
def euclidianDist(a, b):
    x_z = (tf.expand_dims(a,0) - tf.expand_dims(b,1))
    return tf.transpose(tf.reduce_sum(tf.multiply(x_z, x_z), 2))
```

2 Making Predictions for Regression

2.1 Choosing the nearest neighbours

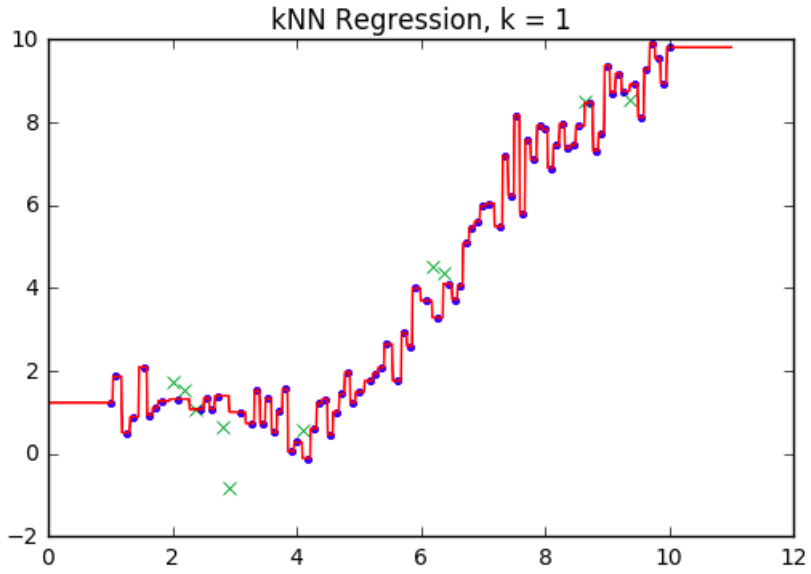
Include the relevant snippets of your Python code.

```
def pickKNearestNeighboursUnscaled(DistMatrix, k):
    length = tf.shape(DistMatrix)[1]

    values, indices = tf.nn.top_k(-DistMatrix, k)
    range = tf.range(length)

    rangeblock = tf.expand_dims(tf.expand_dims(range, 0),0)
    indexblock = tf.expand_dims(indices, 2)

    truth_matrix = tf.reduce_sum(tf.to_float(tf.equal(rangeblock, indexblock)),1)

    return truth_matrix

def pickKNearestNeighbours(DistMatrix, k):
    length = tf.shape(DistMatrix)[1]

    values, indices = tf.nn.top_k(-DistMatrix, k)
```

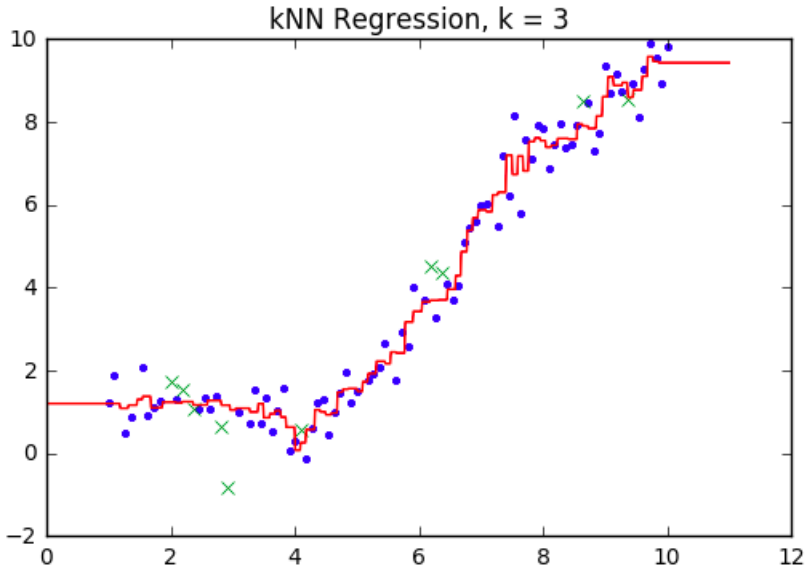
```
    range = tf.range(length)

    rangeblock = tf.expand_dims(tf.expand_dims(range, 0),0)
    indexblock = tf.expand_dims(indices, 2)

    truth_matrix = tf.reduce_sum(tf.to_float(tf.equal(rangeblock, indexblock)),1)

    return truth_matrix / tf.to_float(k)
```

2.2 Prediction

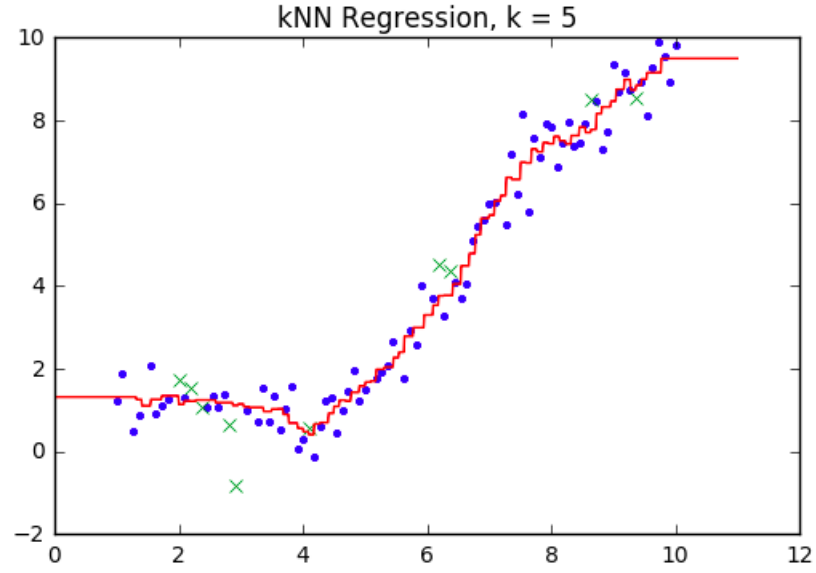
The best k using the validation error is k=1, which has the minimum validation MSE loss.



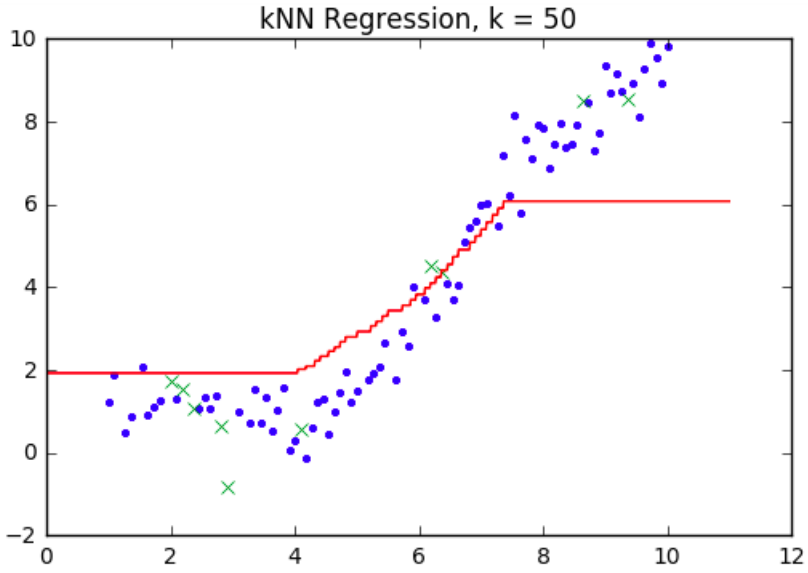

```
(K = 1) MSE values: Valid = 0.5430993, Test = 0.622008, Train = 0.0
```




```
(K = 3) MSE values: Valid = 0.6525566, Test = 0.29018393, Train = 0.21048422
```




```
(K = 5) MSE values: Valid = 0.6208772, Test = 0.35665315, Train = 0.23708253
```




```
(K = 50)        MSE values: Valid = 2.4574032, Test = 1.4138691, Train = 2.4960175
Value of K for lowest MSE in valid data set is 1
```

The plot shows that for k=1, there is an overfitting problem, so the curve has many wiggles, including much noise information. However, if k is too large, such as k=50, the difference between predicted y value and true y value is obvious. So in summary, k = 5 is the best.

3 Making Predictions for Classification

3.1 Predicting class label

Include the relevant snippet of code for this task.

```
def predictFinalValue(inputX, trainX, trainY, K):
    Knearest = pickKNearestNeighboursUnscaled(euclidianDist(inputX, trainX), K)
    predictionMatrix = tf.multiply(Knearest, trainY + 1)

    outputY = tf.zeros(0)
    for i in range(0, runCount):
        y, idx, count = tf.unique_with_counts(predictionMatrix[i])
        count2 = tf.slice(count, [1], tf.shape(count)-1)
        item_class = tf.expand_dims(y[tf.argmax(count2)+1],0)
        outputY = tf.concat([outputY, item_class],0)

    outputY = outputY - 1
    return tf.to_int32(outputY)
```

3.1.1 Face recognition using k-NN

For k = 1, validation accuracy is 61/92;
For k = 5, validation accuracy is 52/92;
For k = 10, validation accuracy is 51/92;
For k = 25, validation accuracy is 52/92;
For k = 50, validation accuracy is 49/92;
For k = 100, validation accuracy is 43/92;
For k = 200, validation accuracy is 28/92;

The value of k that achieves the best validation accuracy is k = 1. If k = 1, the test accuracy is

For k = 1, test accuracy is 66/93;
For k = 5, test accuracy is 63/93;
For k = 10, test accuracy is 63/93;
For k = 25, test accuracy is 62/93;
For k = 50, test accuracy is 52/93;
For k = 100, test accuracy is 44/93;
For k = 200, test accuracy is 34/93
;

3.1.2 Face recognition using k-NN