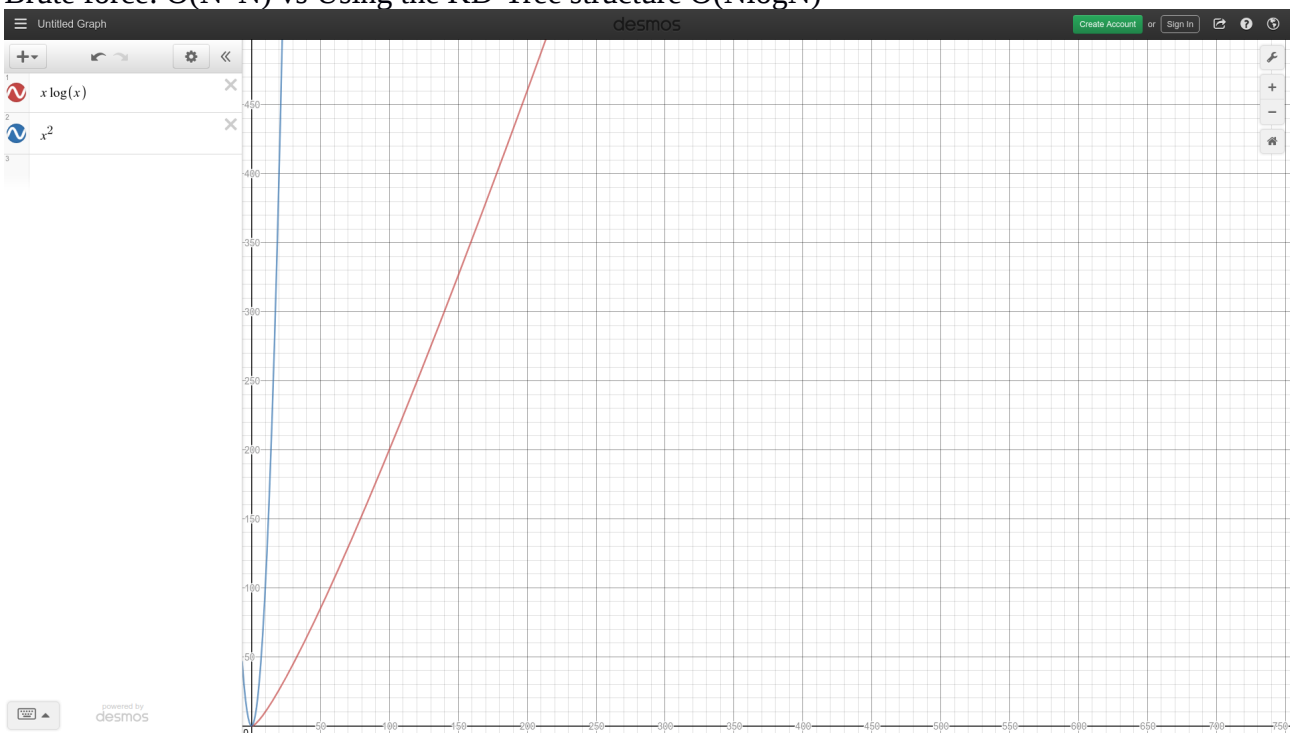
Intro. to Machine Learning Project2
KDTree KNN Classifier and PCA Algorithm
0416324 An-Fong Hwu

Build environment (Note, this report is written in md-like format)
* Ubuntu 16.04 LTS 64bit
* python 3.5.2
* Intel Core i5-7500 3.4GHz 4C4T
* DDR4 -2400 16GB dual channel

What is KD-Tree?
* A BST like structure ,which now implement in the high dimensional structure for fast querying the nearest point in high dimension, i.e. K-Nearest Neighbors algorithm can be finished faster
See more at: https://en.wikipedia.org/wiki/K-d_tree
* Algorithm time complexity analysis.
Brute force: O(N^N) vs Using the KD-Tree structure O(NlogN)



How KD-Tree is built
It is just same as the BST, but now extended into K-Dimensions.
We just change the axis of comparison each time we build one more depth in the tree, recording the current splitting axis and split the data in half (data that < medium and the other, according to the splitting axis)

```python
def create_kd_tree(root,point_data_set,split_attribute,top_N):
    point_data_len = len(point_data_set)
    median_index = int(len(point_data_set)/2)
    point_data_set.sort(key=lambda x:x[split_attribute])
    point = point_data_set[median_index]
    root =  kd_point(point,split_attribute)

    if point_data_len ==1: #build over
        return root

    if split_attribute == top_N+1:
        split_attribute = 2
    else:
        split_attribute += 1

    if median_index > 0:
        root.left_child = create_kd_tree(root.left_child, point_data_set[:median_index],split_attribute,top_N)
    if median_index < len(point_data_set)-1:
        root.right_child = create_kd_tree(root.right_child, point_data_set[median_index+1:],split_attribute,top_N)

    return root
```

How to find K Nearest Neighbors

[Pseudo code]

current_knn how many k I've done

all_knn of k nearest neighbors

Initialize current_point =root min_distance = INF traversed_point = None

while current_point is not leaf

    do binary_search in KD-Tree

    traversed_point.push(current)

    update the min_dustance if the current_point has not been marked as KNN_traversed before

end while

while traversed_point is not empty

    current_point = traversed_point.pop()

    if current_point is leaf

        if distance(query_point,current_point) < min_distance

            NN = current_point

            min_distance = distance(query_point,current_point)

        end if

    end if

    else

        if distance(query_point[current_split_axis],current_point[current_split_axis]) < min_dist or current_knn < all_knn) #distance b/w current point of splitting axis with query point is closer than the min_dist shows that there might be a closer neighbor at other side

        if query_point[current_split_axis]<current_point[current_split_axis]

            go to its right_child(opposite)

        end if

        else

            go to its left_child

        end else

        while current_point is not leaf

            do binary_search from opposite point

            traversed_point.push(current)

            update the min_dustance if the current_point has not been marked as KNN_traversed before

    end else

end while

return NN and mark the NN.KNN_traversed as true for found the kth nearest neighbor

What is PCA algorithm?
* A statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.

* Procedure
1.Find the mean_value_matrix according to each column's average (minus the average for shifting to see how much does the current data differs from the group's average, in order to check the data dependency)
2.Find the covariance matrix according to the mean value matrix to see the data dependency and the degree of distribution
3.Find the eigen value and eigen vectors according to covariance matrix
4.The top L eigen value will be in the dimension of N*L where the dimension of original mean (the eigen value will be listed in column vector) value matrix is M*N --→ thus the oprimized data structed can be reduced to L where L<<N, hence dimensional reduction is completed