CS2305S Take-home exam
Lai Xiao Ni
U077151L

1. The problem can be summarized as follows: how to select the minimal subset of vectors in the dataset as the dictionary entries, under the condition that each vector in the dataset has a Euclidean distance of no more than d from its closest approximation in the dictionary.

   It can be abstracted as the "minimum dominating set" graph problem.

   In the graph, each node represents a vector in the large dataset; each edge between two nodes represents the existence of Euclidean distance of no more than d between two vectors, i.e. vectors with Euclidean distance larger than d would not be linked. The minimum dominating set of this graph is found and nodes contained in this selected subset would be the vectors that are qualified to be dictionary entries.

2. I have designed an optimal algorithm for finding the minimum dominating set of a graph. There are multiple loops in this algorithm and it can be considered as a brute-force algorithm in the worst case. But despite its inefficiency, it can always returns the best solution.

   For a graph with n nodes, we have an integer k such that $k = \{x : 1 \le x \le n, x \in Z\}$.

   Void findMDS(input graph) {

   k is initialized to 1.
   Boolean value needToContinue is set to true;

   Loop (n times; needToContinue must be true) {
   Choose ${}^nC_k$ number of combinations with k nodes.
   S = {set of nodes in one of the many combinations of k nodes}
   Loop (${}^nC_k$ times; needToContinue must be true) {
   Initialize all nodes outside S to be unconnected.
   Select a node A_i in the graph which is outside S.
   Loop ((n-${}^nC_k$) times) {
   Check if A_i is adjacent to at least one node in S.
   If (no nodes in S is adjacent to A_i)
   {stop this loop;} //no longer need to consider this combination
   If (some nodes in S are adjacent to A_i)
   { Mark A_i to be connected;
   A_i is set to another unconnected node;}
   }
   If (all nodes outside S are connected)
   {Store the combination S;

needToContinue is set to false;
stop the loop;} //successful in finding MDS
S is set to another unsearched combination of k nodes;
}
k is incremented by 1;
}

}

One heuristic used here is that the algorithm stops once it finds all nodes outside S are adjacent to at least one node in S, the program will stops and recognizes S as the minimum dominating set.

This can be well justified because I start the size of dominating set in increasing order (from 1 to n) over the time. That means once I found a suitable S, that is the minimum dominating set at this moment. Those dominating sets that are found in later iterations would have a size larger than the current one. So this is why it is sensible to stop the program and put S as the minimum dominating set.

3. I used induction to prove the statement.

Let P_n be the statement "the greedy algorithm will generate a dictionary with size $Y_n \le ceiling(s_n \log |D|)$, given Y_n is the size of dictionary generated by greedy algorithm, |D| is the size of dataset and $|D| = \{x; x \ge 2; x \in Z\}$, s_n is the size of smallest dictionary.

Initial step: when |D| = 2, assume $s_2 = 1$ or two vectors present are within Euclidean distance of d;
LHS = Y_2=1;
RHS = ceiling(s_2 log 2) = ceiling (0.693 * 1) = 1;
LHS $\le$ RHS;
Assume s_2 =2 or two vectors are not within Euclidean distance of d;
LHS = Y_2=2;
RHS = ceiling(s_2 log 2) = ceiling (0.693*2) = 2;
LHS $\le$RHS;
So P_2 is true.

Inductive step: Assume |D| = k, s_k , P_k is true, i.e. $Y_k \le ceiling(s_k \log k)$;
We want to prove P_{k+1} is also true, i.e. $Y_{k+1} \le ceiling(s_{k+1} \log(k+1))$.
There are a few different cases that are under consideration:

Case 1: the newly-added vector is not within Euclidean distance d with any vectors in the smallest dictionary.
$s_{k+1} = s_k + 1$;
$Y_{k+1} = Y_k + 1$; the greedy algorithm would not return a dictionary such that not all vectors can find one vector in dictionary that are within Euclidean distance of d

because unlinked nodes/vectors remained in set S would always be scanned and put in dictionary.

$Y_{k+1} = Y_k + 1 \le ceiling(s_k \log_e k) + 1$

$\le ceiling(s_k \log_e (k+1)) + \log_e (k+1)$

$\le ceiling(s_k \log_e (k+1) + \log_e (k+1))$

$= ceiling((s_k + 1)\log_e (k+1))$

So $Y_{k+1} \le ceiling((s_k + 1)\log_e (k+1))$;

In case 1, P_{k+1} is true when P_k

Case 2: the newly-added vector is within Euclidean distance d with any vectors in the smallest dictionary.

$s_{k+1} = s_k$;

Sub-case a: $Y_{k+1} = Y_k$; the greedy algorithm recognizes newly-added vector is within Euclidean distance d of any of the vectors in constructed dictionary.

$Y_{k+1} = Y_k \le ceiling(s_k \log(k))$

$\le ceiling(s_{k+1} \log(k+1))$

So in sub-case a, P_{k+1} is true when $P_{k.}$

Sub-case b: $Y_{k+1} = Y_k + 1$; the newly-added vector is not within Euclidean distance d of any vectors in the original dictionary generated by greedy algorithm.

But if we think if carefully, if the newly-added vector is within Euclidean distance d with any vectors in the smallest dictionary, the greedy algorithm will either select it or select vectors adjacent to it. If the algorithm selects it, one of its adjacent vector in the generated dictionary would become redundant. If the algorithm select vectors adjacent to it, the adding of this new vector will not increase the size of the generated dictionary.

Therefore, sub-case b does not exist.

In all, P_{k+1} is true when $P_{k.}$

Since the base case and the inductive step is also true, the statement P_n is true.

4. See in the Appendix the attached code and printed details on the tests provided by the significance testing software, which is R in this case.

5. Drawbacks of Professor KnowItAll's work:

5.1. <u>Formulation of the problem 1:</u> The abstraction of image compression to the Euclidean distances problem is limited to simple gray-scale images. This is because the calculation of Euclidean distances over and over again during the dictionary construction, image storing and retrieval is computationally space-consuming and time-consuming, to the point that it may cause great inaccuracy in the conversion of image data. The question chooses images that are consisted of 8-bit grey-scale image to prove the effectiveness of the proposed method. But in reality, 36-bit RGB color model is more frequently used. In this case, the

summation of product of two numbers with large number of decimal places is involved frequently. The computers may not have enough processing power to do these computations accurately and over-flow in numbers is quite likely to happen.

5.2. Formulation of the problem 2: The size of dictionary constructed may be excessively large if the n-dimentional vectors are sparsely scattered around in Euclidean space. In the worst case, all formulated vectors could not find one another which is within Euclidean distance of d and thus the size of dictionary is the same as the size of the dataset, which makes the whole process of image compression purposeless. To avoid this, d may be set to be a larger number. But this again creates another problem: the accuracy of the compressed images is in doubt because many formulated vectors would be compressed as vectors that are not very similar to their original shapes.

5.3. Formulation of the problem 3: Some parts of the image may be stored with higher quality than other parts, because some formulated vectors can find vectors in dictionary that are within closer Euclidean distances whilst some others can only find vectors in dictionary that are within further Euclidean distances with them. The distorts the scalability in the precisions of different parts of the image.

5.4. Experiment Setup: The experiment proposed is to construct a dictionary based on 1000 vectors extracted from 10 images and then use this particular dictionary to compress the 10 images, and thus prove the effectiveness of the theory. The sample size in this experiment may be considered as too small to be able to prove how the theory is effective. This is because 10 images may not represent all possible combinations of images and thus many possible problems of this method may not be detected.