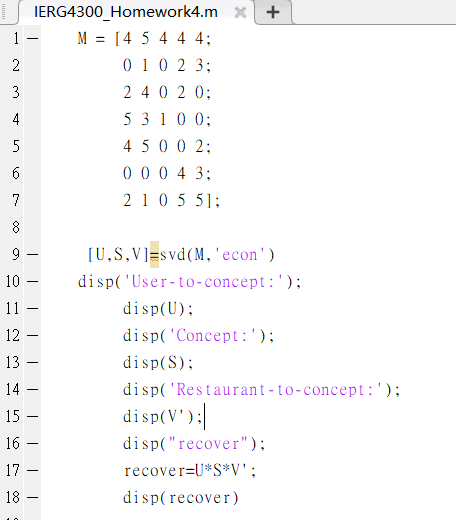
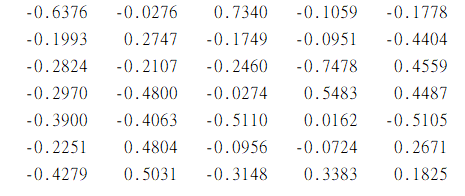
IERG4300 Homework4

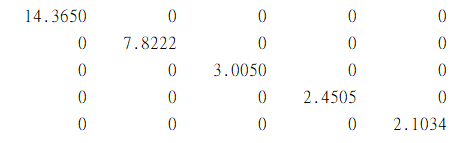
Chim Ka Long 1155094482

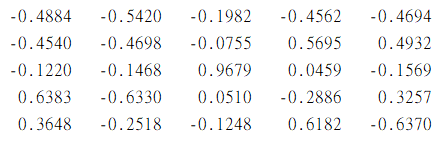
1a). We use the following code in MatLab to find SVD.



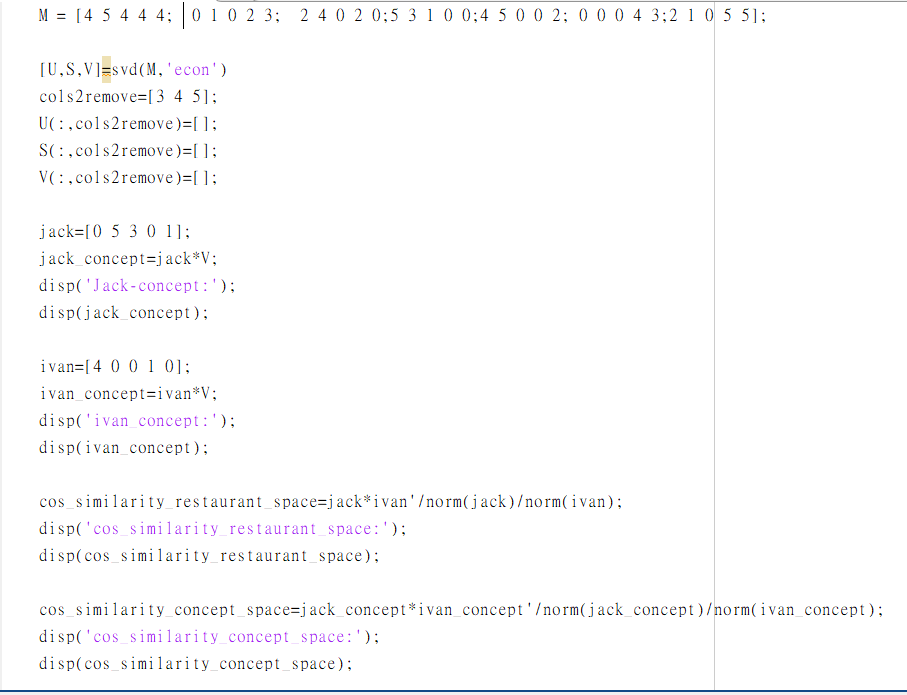
Result:

U= 

Σ=

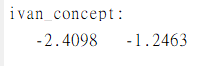
V^T=

1b). Matlab code:



Result:

1bi). 

1bii). 

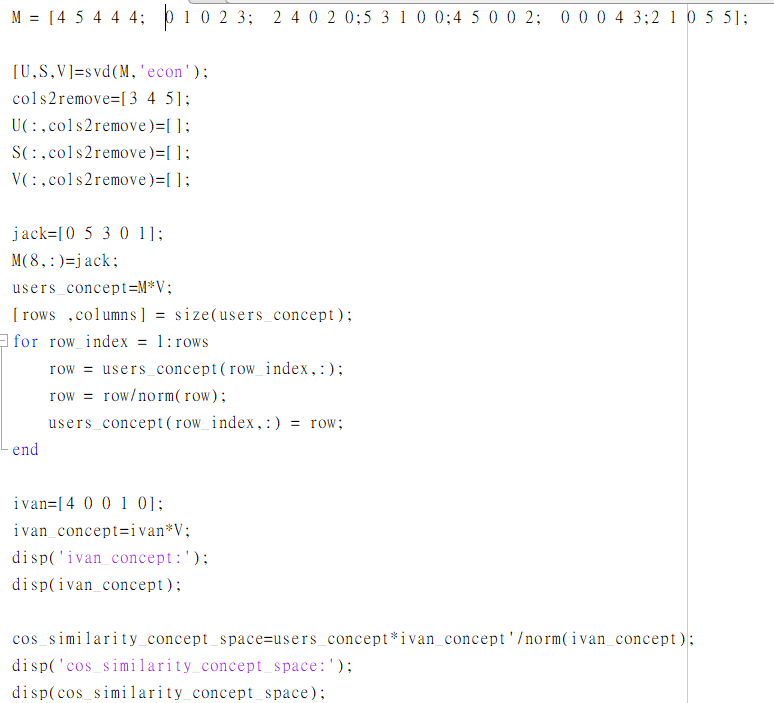
1biii). 



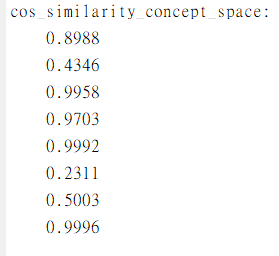
1biv). Similiarity of concept space reflect more accuratly. In restaurant space, Ivan and jack do not try same restaurant, so that their similiarity is 0. Although they have not tried same restaurant, we can still guess their preference by the rating on different restaurant and the correlationship between restaurants (represent by V). For example, Ivan only try restaurant A and rate 5 and Jack only try restaurant B and also rate 5. If all other users who have tried these 2 restaurants give same rating, we can guess these 2 restaurant are very similar, thus guess Ivan and Jack have similar preference. Through this extreme example, we can obseve that concept space is more accurate.

1bV). I calculate the cosine similarity between Ivan and all other user include Jack.

Code:



Result:



Then I select the top 2 user with highest similarity and all of them have rated Franklin and UC (Jack, Eve). Then prediction is the average of rating of these 2 users. Predicted rating of Franklin and UC are 5 and 1.5

2). In Homework3, I unzip the file and preprocess the raw file to a line of integers for each image. I reuse the processed file. The initial random seed is generated by random choosing 10 points which have different label from 0 to 9, by code gen\_seed\_new.py. I rerun HW3 with this new generate seed method to compare the result of HW4.

We need to convert the original data with 784 dimensions to new 25 dimensions with PCA. Firstly, we need to center all data points, we calculate the offset of training set and apply it to train set, test set and initial random seed by code in center\_train.py. Then we need to calculate the eigenvector of train set in eigenvector\_train.py and output the top 25 eigenvectors with highest eigenvalues at data/train\_images\_eigenvector. Finally, we convert the train set, test set and initial random seed to vectors with new 25 dimension.

In the rest, all are same with Homework3 question 2. All algorithm explanation please refer to Homework 3 lab report. The following is the result with 25 dimensions data after implement PCA.

Comparing to implement K-mean with 784 dimensions, the accuracy of this one with 25 dimensions is sometimes higher and sometimes. In HW3, the accuracy of random seed 0, 1 and 2 on train set are 57.708%, 58.942% and 59.672% respectively. In HW4, the accuracy of random seed 0, 1 and 2 on train set are 59.077%, 57.617% and 59.231%. The advantage is that memory drops and the computing speed rise largely.

Train set Output:

Table. 1. The Accuracy of Clustering Performance with Random Seed 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cluster Number | # train images belong to the cluster | Major label of certain images | # correctly clustered images | Classification Accuracy (%) |
| 0 | 5139 | 6 | 4420 | 86.01 |
| 1 | 6655 | 7 | 2772 | 41.65 |
| 2 | 5413 | 4 | 2257 | 41.70 |
| 3 | 7453 | 3 | 3983 | 53.44 |
| 4 | 3682 | 0 | 3421 | 92.91 |
| 5 | 4551 | 2 | 4112 | 90.35 |
| 6 | 7284 | 9 | 2459 | 33.76 |
| 7 | 5540 | 8 | 3491 | 63.01 |
| 8 | 10349 | 1 | 6585 | 63.63 |
| 9 | 3934 | 0 | 1946 | 49.47 |
| Total | 60000 | NA | 35446 | 59.08 |

Table. 2. The Accuracy of Clustering Performance with Random Seed 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cluster Number | # train images belong to the cluster | Major label of certain images | # correctly clustered images | Classification Accuracy (%) |
| 0 | 5699 | 1 | 3744 | 65.70 |
| 1 | 6917 | 8 | 3162 | 45.71 |
| 2 | 6829 | 7 | 2801 | 41.02 |
| 3 | 4996 | 0 | 4693 | 93.94 |
| 4 | 5493 | 4 | 2212 | 40.27 |
| 5 | 7643 | 3 | 3853 | 50.41 |
| 6 | 4613 | 2 | 4131 | 89.55 |
| 7 | 7178 | 9 | 2469 | 34.40 |
| 8 | 5429 | 6 | 4561 | 84.01 |
| 9 | 5203 | 1 | 2944 | 56.58 |
| Total | 60000 | NA | 34570 | 57.62 |

Table. 3. The Accuracy of Clustering Performance with Random Seed 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cluster Number | # train images belong to the cluster | Major label of certain images | # correctly clustered images | Classification Accuracy (%) |
| 0 | 3965 | 0 | 1961 | 49.46 |
| 1 | 5287 | 6 | 4566 | 86.36 |
| 2 | 3688 | 0 | 3425 | 92.87 |
| 3 | 7084 | 7 | 2424 | 34.22 |
| 4 | 10394 | 1 | 6588 | 63.38 |
| 5 | 7436 | 3 | 3970 | 53.39 |
| 6 | 5389 | 4 | 2279 | 42.29 |
| 7 | 4571 | 2 | 4123 | 90.20 |
| 8 | 6591 | 7 | 2712 | 41.15 |
| 9 | 5595 | 8 | 3491 | 62.39 |
| Total | 60000 | NA | 35539 | 59.23 |

Test set Output with Random Seed 0:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cluster Number | Label of cluster | # test image belongs to the cluster | # correctly clustered images | Classification Accuracy (%) |
| 0 | 0 | 656 | 306 | 46.65 |
| 1 | 6 | 864 | 745 | 86.23 |
| 2 | 0 | 618 | 573 | 92.72 |
| 3 | 9 | 1318 | 477 | 36.19 |
| 4 | 1 | 1653 | 1102 | 66.67 |
| 5 | 3 | 1301 | 700 | 53.80 |
| 6 | 4 | 858 | 375 | 43.71 |
| 7 | 2 | 808 | 709 | 87.75 |
| 8 | 7 | 1000 | 409 | 40.9 |
| 9 | 8 | 924 | 586 | 63.42 |
| Total | NA | 10000 | 5982 | 59.82 |

3a). Algorithm:

To implement minhash, we need to convert the byte array of 784 dimensions to bit array of 784 dimensions firstly. The purpose of this part is to convert a byte array of 784 dimensions to 64 signatures.

We generate 25 universal hash functions f(x) = (a\*x + b) % 787 %784, where a and b are random integer below 787 and 787 is a prime number larger than 784, to do a trick with same result of do permutation on array. For each hash function, we input the dimension index from 0 to 783 as x into f(x) and the property of universal hash function will output value uniformly, such that we will get something like array of elements from 0 to 783 after permutation. Furthermore, we will finally store the least hash value where the corresponding bit in that dimension is 1.

The advantage of this trick is to avoid computing the permutation and save the memory storing the array. Random permutation on an array is a hard work of computation.

3b). Algorithm:

First, we need to find 10 representative points and other points on their cluster. I write a code representative.py to find the nearest point of 10 centroids and other points in their cluster in term of index with initial random seed 2 in the Homework 3 question 2. The code output points’ index and label in first line and the points index on this cluster in second line. We find the index instead of the byte array of 784 dimensions, because we can find the corresponding 64 signatures generated before by index.

Then we can implement the LSH with code LSH.py. In first part, we find the corresponding 64 signatures for each representative and store it with label into class representative. In second part, for each set of bands, rows and buckets, we generate the universal hash function f(s) = ((a1 \* x1 + a2 \* x2 + a3 \* x3 + …… + b) %p) %k. Then we calculate the hash value (or called bucket number) of each band for each representative. For each cluster, calculate the hash value of data points and compare it to the hash value of representatives in same cluster. If there are same hash values in at least one band, the data point is recognized as candidate point with that representative.

Output:

Table 1: LSH Results with k=10, r=8, B=8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cluster id of the representative | Label of the representative | # candidate images similar to the representative | # candidate images having the same label as the representative | Accuracy (%) |
| 0 | 9 | 4737 | 1549 | 32.70 |
| 1 | 0 | 2115 | 1198 | 56.64 |
| 2 | 6 | 3252 | 2846 | 87.52 |
| 3 | 8 | 2640 | 2212 | 60.77 |
| 4 | 7 | 4275 | 1835 | 41.94 |
| 5 | 4 | 3397 | 1460 | 42.98 |
| 6 | 1 | 6412 | 4016 | 62.63 |
| 7 | 3 | 4763 | 2531 | 53.14 |
| 8 | 0 | 2100 | 1955 | 93.10 |
| 9 | 2 | 2713 | 2452 | 90.38 |
| Total Set | NA | 37504 | 22054 | 58.80 |

Table 1: LSH Results with k=10, r=4, B=16

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cluster id of the representative | Label of the representative | # candidate images similar to the representative | # candidate images having the same label as the representative | Accuracy (%) |
| 0 | 9 | 6625 | 2297 | 34.67 |
| 1 | 0 | 3213 | 1796 | 55.90 |
| 2 | 6 | 4880 | 4300 | 88.11 |
| 3 | 8 | 5471 | 3364 | 61.49 |
| 4 | 7 | 5692 | 2375 | 41.73 |
| 5 | 4 | 4588 | 2025 | 44.14 |
| 6 | 1 | 9596 | 6204 | 64.65 |
| 7 | 3 | 6878 | 3748 | 54.49 |
| 8 | 0 | 3338 | 3105 | 93.02 |
| 9 | 2 | 4368 | 3951 | 90.45 |
| Total Set | NA | 54649 | 33165 | 60.69 |

Table 1: LSH Results with k=10, r=16, B=4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cluster id of the representative | Label of the representative | # candidate images similar to the representative | # candidate images having the same label as the representative | Accuracy (%) |
| 0 | 9 | 2256 | 725 | 32.14 |
| 1 | 0 | 1660 | 935 | 56.33 |
| 2 | 6 | 2031 | 1777 | 87.49 |
| 3 | 8 | 2168 | 1328 | 61.25 |
| 4 | 7 | 2818 | 1135 | 40.28 |
| 5 | 4 | 1635 | 685 | 41.90 |
| 6 | 1 | 3967 | 2491 | 62.79 |
| 7 | 3 | 3456 | 1825 | 52.81 |
| 8 | 0 | 1150 | 1058 | 92 |
| 9 | 2 | 2302 | 2088 | 90.70 |
| Total Set | NA | 23443 | 14047 | 59.92 |

Table 1: LSH Results with k=100, r=8, B=8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cluster id of the representative | Label of the representative | # candidate images similar to the representative | # candidate images having the same label as the representative | Accuracy (%) |
| 0 | 9 | 786 | 262 | 33.33 |
| 1 | 0 | 367 | 268 | 73.02 |
| 2 | 6 | 689 | 630 | 91.44 |
| 3 | 8 | 695 | 506 | 72.81 |
| 4 | 7 | 766 | 423 | 55.22 |
| 5 | 4 | 461 | 220 | 47.72 |
| 6 | 1 | 1106 | 825 | 74.59 |
| 7 | 3 | 684 | 411 | 60.09 |
| 8 | 0 | 614 | 586 | 95.44 |
| 9 | 2 | 536 | 504 | 94.03 |
| Total Set | NA | 6704 | 4635 | 69.14 |

Table 1: LSH Results with k=100, r=4, B=16

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cluster id of the representative | Label of the representative | # candidate images similar to the representative | # candidate images having the same label as the representative | Accuracy (%) |
| 0 | 9 | 3891 | 1621 | 41.66 |
| 1 | 0 | 1742 | 1161 | 66.65 |
| 2 | 6 | 3354 | 3033 | 90.43 |
| 3 | 8 | 4028 | 2689 | 66.76 |
| 4 | 7 | 3777 | 1816 | 48.08 |
| 5 | 4 | 2123 | 1154 | 54.36 |
| 6 | 1 | 4122 | 3144 | 76.27 |
| 7 | 3 | 4041 | 2641 | 65.36 |
| 8 | 0 | 2632 | 2488 | 94.53 |
| 9 | 2 | 3107 | 2898 | 93.27 |
| Total Set | NA | 32817 | 22645 | 69.00 |

Table 1: LSH Results with k=100, r=16, B=4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cluster id of the representative | Label of the representative | # candidate images similar to the representative | # candidate images having the same label as the representative | Accuracy (%) |
| 0 | 9 | 265 | 92 | 34.72 |
| 1 | 0 | 155 | 89 | 57.42 |
| 2 | 6 | 193 | 165 | 85.49 |
| 3 | 8 | 221 | 143 | 64.71 |
| 4 | 7 | 267 | 130 | 48.69 |
| 5 | 4 | 225 | 103 | 45.78 |
| 6 | 1 | 406 | 257 | 63.30 |
| 7 | 3 | 355 | 191 | 53.80 |
| 8 | 0 | 133 | 125 | 93.98 |
| 9 | 2 | 188 | 165 | 87.77 |
| Total Set | NA | 2408 | 1460 | 60.63 |

Table 1: LSH Results with k=1000, r=8, B=8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cluster id of the representative | Label of the representative | # candidate images similar to the representative | # candidate images having the same label as the representative | Accuracy (%) |
| 0 | 9 | 253 | 119 | 47.04 |
| 1 | 0 | 98 | 87 | 88.78 |
| 2 | 6 | 400 | 394 | 98.5 |
| 3 | 8 | 301 | 267 | 88.70 |
| 4 | 7 | 338 | 207 | 61.24 |
| 5 | 4 | 161 | 94 | 58.39 |
| 6 | 1 | 320 | 279 | 87.19 |
| 7 | 3 | 256 | 195 | 76.17 |
| 8 | 0 | 422 | 412 | 97.63 |
| 9 | 2 | 237 | 235 | 99.16 |
| Total Set | NA | 2786 | 2289 | 82.16 |

Table 1: LSH Results with k=1000, r=4, B=16

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cluster id of the representative | Label of the representative | # candidate images similar to the representative | # candidate images having the same label as the representative | Accuracy (%) |
| 0 | 9 | 3553 | 1545 | 43.48 |
| 1 | 0 | 1565 | 1090 | 69.65 |
| 2 | 6 | 3066 | 2798 | 91.26 |
| 3 | 8 | 3915 | 2630 | 67.18 |
| 4 | 7 | 3586 | 1755 | 48.94 |
| 5 | 4 | 1723 | 983 | 57.05 |
| 6 | 1 | 3003 | 2589 | 86.21 |
| 7 | 3 | 3656 | 2467 | 67.47 |
| 8 | 0 | 2557 | 2426 | 94.88 |
| 9 | 2 | 2881 | 2694 | 93.51 |
| Total Set | NA | 29505 | 20977 | 71.10 |

Table 1: LSH Results with k=1000, r=16, B=4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cluster id of the representative | Label of the representative | # candidate images similar to the representative | # candidate images having the same label as the representative | Accuracy (%) |
| 0 | 9 | 33 | 14 | 42.42 |
| 1 | 0 | 19 | 10 | 52.63 |
| 2 | 6 | 19 | 16 | 84.21 |
| 3 | 8 | 30 | 22 | 73.33 |
| 4 | 7 | 25 | 11 | 44 |
| 5 | 4 | 21 | 5 | 23.81 |
| 6 | 1 | 45 | 26 | 57.78 |
| 7 | 3 | 21 | 13 | 61.90 |
| 8 | 0 | 27 | 25 | 92.59 |
| 9 | 2 | 23 | 23 | 100 |
| Total Set | NA | 263 | 165 | 62,74 |

3c). In above 9 setup, the total accuracy of them are above or equal to about 58%. If the total accuracy is less than 59.67%, which is the accuracy in HW3, that means LSH with this setup is useless or even harmful to accuracy. In HW3, we calculate accuracy of a cluster is selecting all points in this cluster and count how many of them have same label. In this part, we know that some points in this cluster may have different label with representative points, so that we are selecting some points similar to representative by LSH method instead of selecting all points. As a result, with suitable setup, the accuracy should be higher.

With more buckets (k), the average accuracy is higher. We find candidate points by hashing them into buckets for each band. The best situation is that the different signature vector in a band must hash into different buckets. When the number of buckets is not high enough, the probability of different signature vectors hashing into same band is higher.

With more row (r) and less band (b), it is more difficult to find candidate points. Obviously, the total candidate points are less in part b result. However, the requirement of being candidate point is higher does not means that the accuracy is higher, because there is a situation that the signature vector is different to the one of representative wrongly hashing into same buckets. With less row (r) and more band (b), it is easy to classify a point is candidate point, which means that a lot of points with different labels are also classified as candidate points.

As a result, to aim higher accuracy, the number of buckets should be high enough. The number of row and bands should not be extreme, but the suitable setup may be different to different kinds of input. At least, in our case, the setup of 1000 buckets, 8 rows and 8 bands have the highest accuracy.

3d). The best result is the setup of 1000 buckets, 8 rows and 8 bands which has 82.16% total accuracy. The accuracy of question 2 with same initial random seed is 59.23%. Nevertheless, they are difficult to compare directly. Both are technique add to K-mean cluster. PCA technique does the dimension reduction on data, which mainly reduce the memory size and boost computation speed of K-mean clustering. PCA does not significantly improve the accuracy comparing to K-mean clustering on 784-dmmensions. If K-mean is called a classification of 60000 data points into 10 cluster, LSH is a filter of points in cluster to rise accuracy. Since LSH filter out other points in same cluster and find candidate point to representative point, the accuracy surely become higher. However it does not give a classification to points which are not candidate points.