CS7641 Machine Learning Assignment III
Unsupervised Learning and Dimensionality Reduction

Wang Lu, GTid: 903355610
29th October ,2018

Dataset Choice

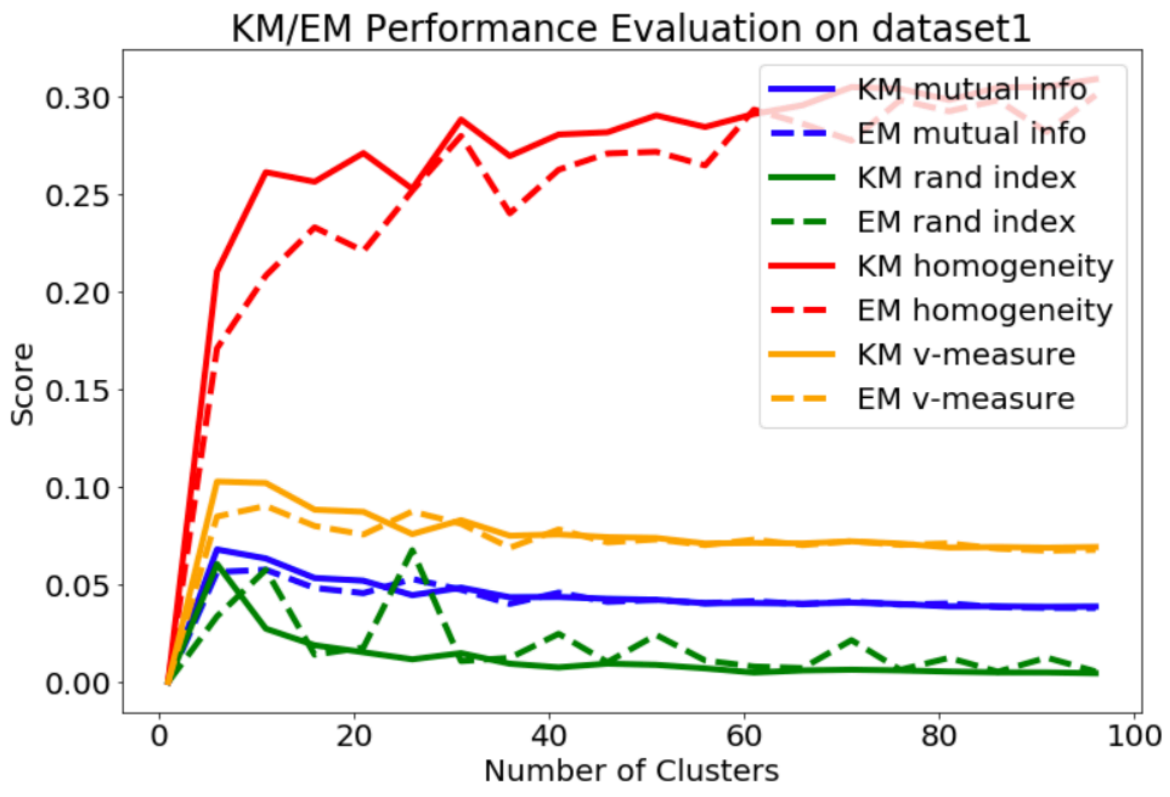
Here I use the two datasets from assignment1 as my subjects. As I mentioned in assignment1, I found the two datasets interesting not only in the application respects but also in their fundamental statistics characteristics: The first dataset is regarding the problem "finding high income individuals" inspired by Udacity's Finding Donors for CharityML project [1], which is an imbalanced binary classification task with originally 8 categorical features and 5 numerical features on 45K entries. The second dataset is collected and processed from commercial website TMDB[2]'s archive, which has a mixture of data types: time series, categories, texts and numbers allowing flexible feature extracting. The regarding problem is a balanced binary classification task on a relatively small dataset (6K entries) with hybrid features. The choice of datasets should allow an interesting comparison of how different feature reduction and clustering methods behave differently in response to the dataset balance, scale, and feature heterogenicity.

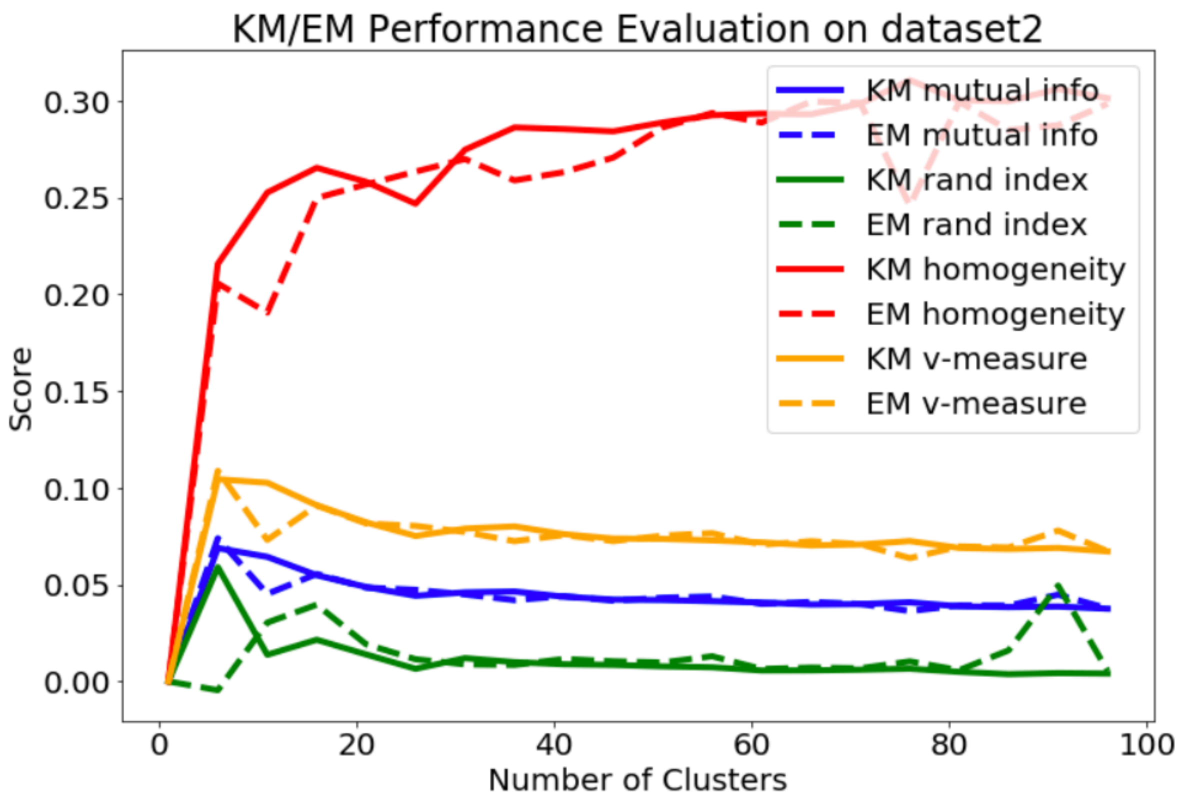
Note: Both datasets are reproduced by same procedure in assignment1, namely feature cleaning, extracting, scaling and standardizing, for the purpose of comparability with previous work. So strictly speaking, the feature preprocessing is finished here already. However, as clustering and feature reduction technique are special focus in this assignment, they are viewed empirically as 'un-preprocessed data' and our observations still hold valid.

1. Clustering

Partial clustering and hierarchical clustering:
Sklearn's KMeans and BayesianGaussianMixture are applied. Both of them by default use k-means++ (producing next center proportional to its distances to all current centers)[3,4] for initialization, and KMeans uses LIoyd's method for iterating and improving[4]. It would be interesting to include a hierarchical clustering such as single link clustering, however this idea was aborted due to computational limit. KM and EM works more efficiently on bigger datasets.

Metrics:
Unlike supervised learning, evaluation of unsupervised learning methods isn't that straightforward. 1) The **silhouette coefficient** is a good indicator at absence of labels: it calculates the mean intra-cluster distance (a) and the mean nearest-cluster distance (b) for each sample, and return coefficient (b - a) / max(a, b)[5], for which values near 0 indicates overlapped clusters and 1 means perfect separation. But notice this metric is highly computationally demanding. 2) Since the data is actually labeled, we have the freedom to use many other metrics such as **mutual information** (to what extend knowing the cluster can tell us about the class)[6], **completeness** (to what extent members of same class actually assigned to same cluster)[7], **homogeneity** (to what extent a class dominates a cluster)[8]. Notice the actual implementation of these metrics include standardizing, with 1 indicates perfect score in that standard.

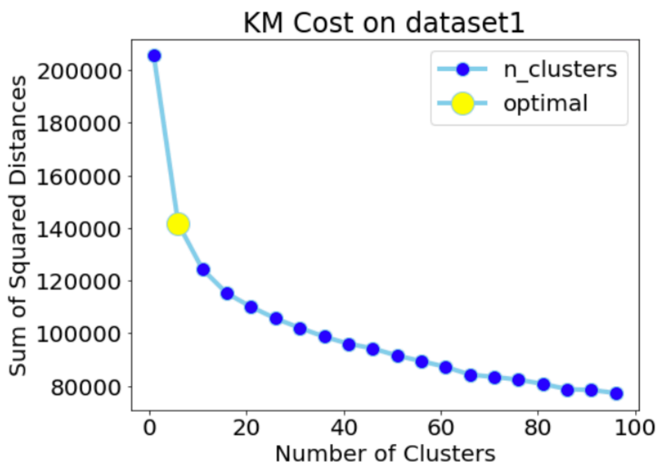


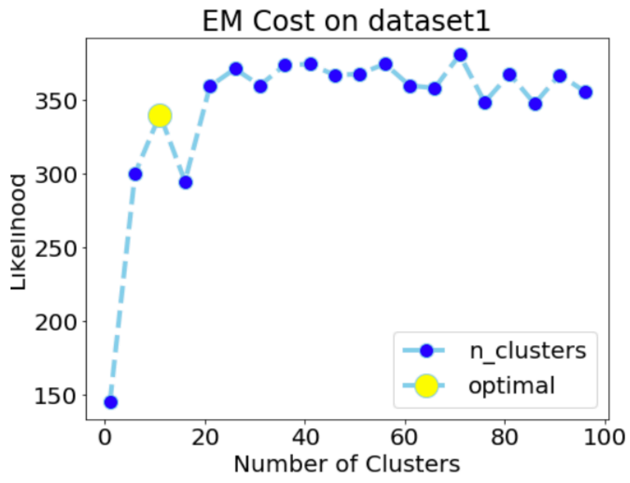


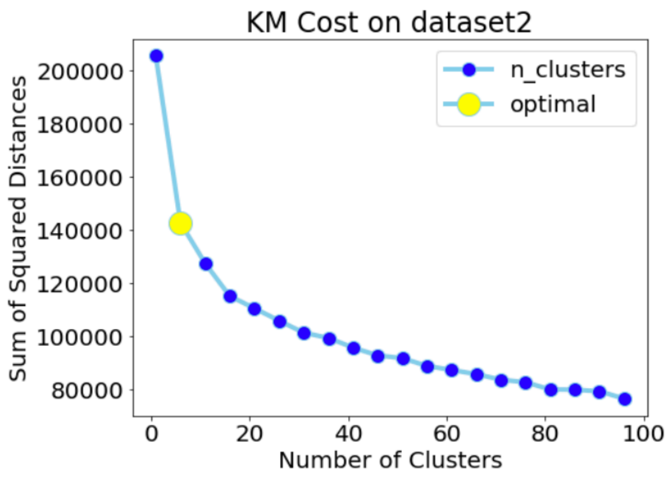
3 patterns are interesting above:
1) KM perform smoother against number of clusters than EM on both datasets. In fact, KM almost always scores slightly better than EM for the two clustering tasks. This is possibly due to the assumption of Gaussian mixture distribution made by EM isn't a good approximation when most of features (98/103 for dataset1, 71/79 for dataset1) are not continuous value that subject to central limit theorem.

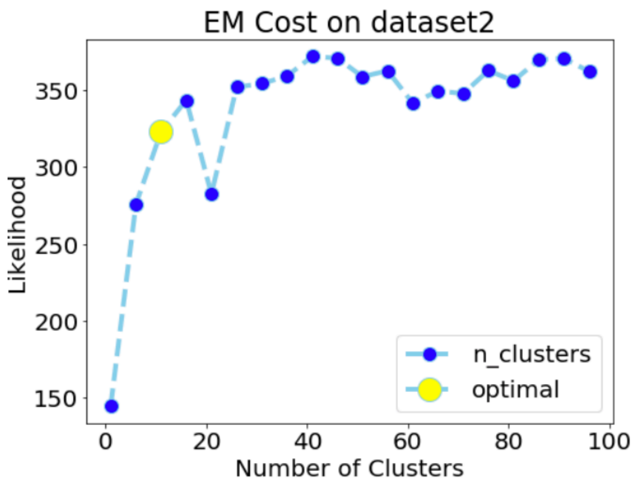
2) homogeneity score grows with increased clusters. Notice this is not a good supporting point to keep increasing clustering numbers, as the naïve solution which every data point forms a cluster of its own would score in homogeneity perfectly.
3) a spike of mutual information/rand index/ v-measure appears around 5-10 clusters and goes down afterwards. This is interesting in that it indicates an ideal cluster number tells us most about the class. As the metrics are adjusted symmetrically to eliminate chance, it's most likely where the optimal cluster number lies, which confirmed by sum of Euclidean distances and log-likelihood curve.

Elbow method:

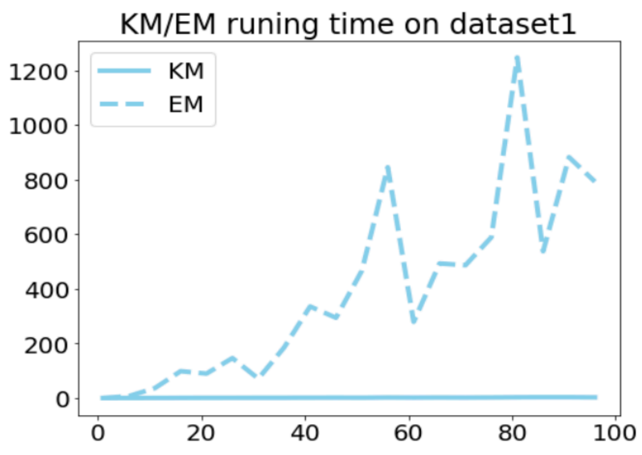


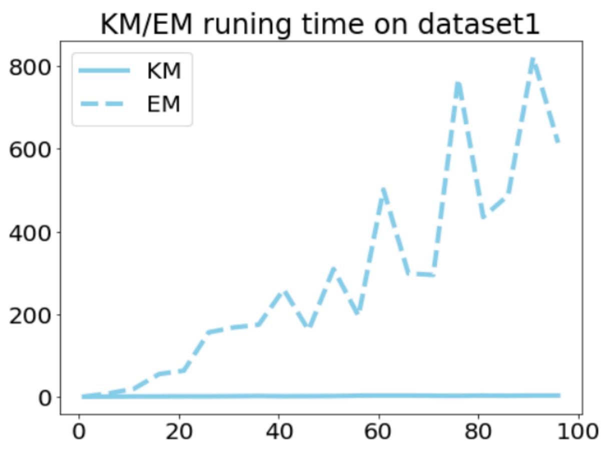






Elbow method[9] on KM score (sum of distances) and EM score (mean log-likelihood) confirm the finding above: about 6 **cluster for KM and** 11 **for EM are ideal for them on both datasets.**



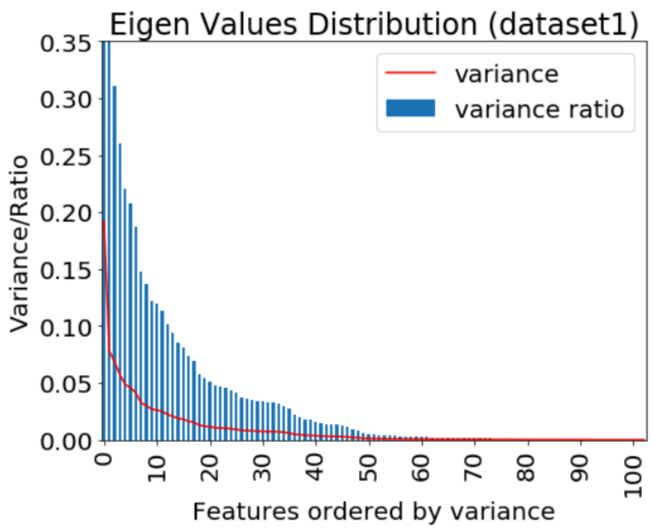


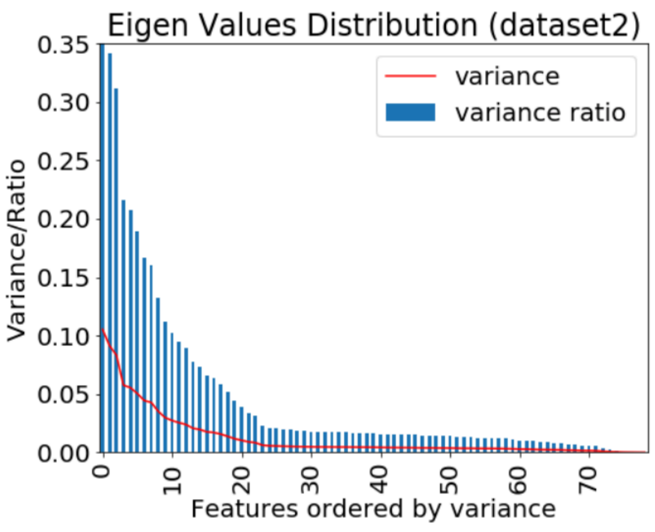
Time cost:
With both max iterations set at 100 and number of initiations set at 1, EM are about 2 to 3 magnitude order slower than KM at the expenses of modeling Dirichlet distribution.

2. Dimensionality Reduction

Beside PCA, ICA and RP, the three dimensionality reduction methods stipulated, I include truncated singular value decomposition (SVD)[10], which does not center data and works well in practice with sparse data structure. This part is meant to explore the optimal dimensions (trade-off between loss of information and gain of time and space efficiency) for each method.

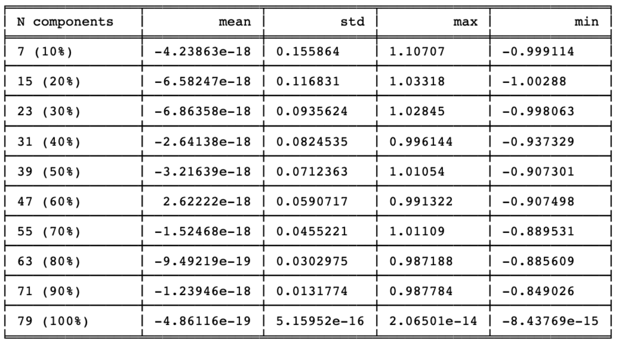
1) Principal Component Analysis





Variance on dataset2 is more averagely distributed, as indicated by the elongated tail of small eigen value features. It's reasonable to think 5% variance bearable information loss. Thus **36** features are chosen for dataset1 (eigen value 0.005 as cutoff, 94% variance captured) and **60** for dataset2 (eigen value 0.003 as cutoff, 97% variance captured).

Reconstruction:
It's intuitive that the more information compression, the more difficult the reconstruction will be, consistent with the increasing variation (range, std) as number of principals decrease. Besides, reconstruction of PCA is high fidelity. When no dimension is thrown, the transformation and inverse transformation itself will only not loss you more information than storage precision limit.



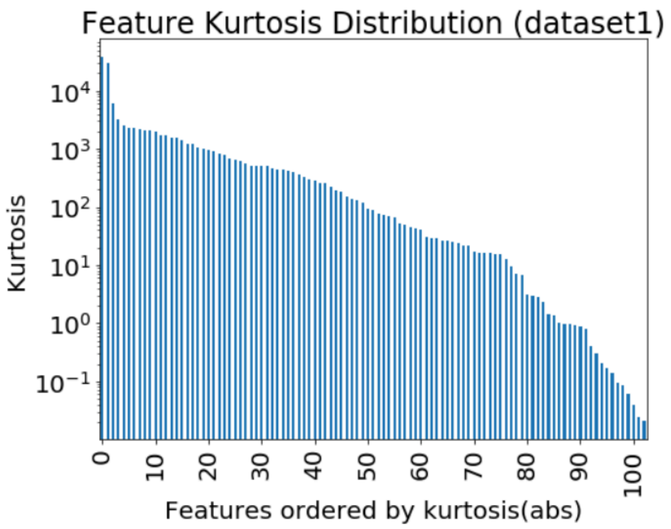
N components	mean	std	max	min
7 (10%)	-4.23863e-18	0.155864	1.10707	-0.999114
15 (20%)	-6.58247e-18	0.116831	1.03318	-1.00288
23 (30%)	-6.86358e-18	0.0935624	1.02845	-0.998063
31 (40%)	-2.64138e-18	0.0824535	0.996144	-0.937329
39 (50%)	-3.21639e-18	0.0712363	1.01054	-0.907301
47 (60%)	2.62222e-18	0.0590717	0.991322	-0.907498
55 (70%)	-1.52468e-18	0.0455221	1.01109	-0.889531
63 (80%)	-9.49219e-19	0.0302975	0.987188	-0.885609
71 (90%)	-1.23946e-18	0.0131774	0.987784	-0.849026
79 (100%)	-4.86116e-19	5.15952e-16	2.06501e-14	-8.43769e-15

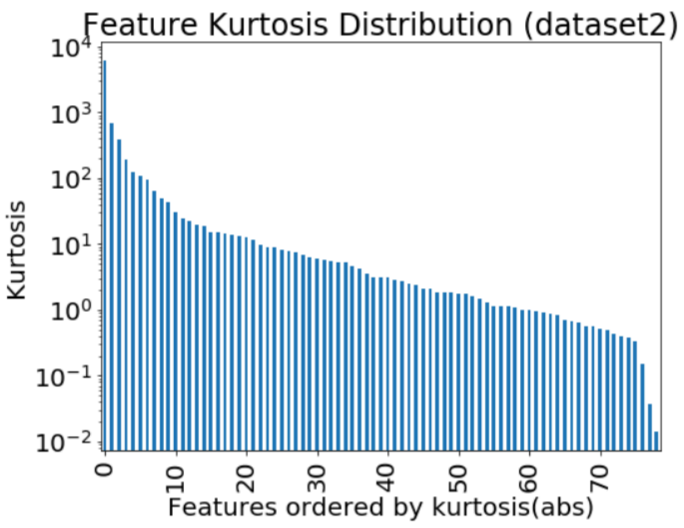
2) Independent Component Analysis

ICA[11] assumes that the observed features are actually blended from independent sources and it wants to sort out the "original" hidden variables guided by central limit theorem that: the more determinants in a process to generate a random variable, the closer it approaches Gaussian distribution. Thus, Kurtosis can be used to rank features. The higher absolute value

Kurtosis(fisher = True) returns, the simpler the feature is and more likely to be the independent variables ICA seeks.

Kurtosis:





Kurtosis[12] score varies across 4 magnitude orders (-1 ~ 4) for dataset1 and 5 orders (-2 ~ 4) for dataset2. When choosing abs(Kurtosis) = 1 as an arbitrary cutoff, **83 and 63** components are suitable ICA on two datasets respectively.

Reconstruction:
ICA aren't as high fidelity as PCA, as indicated by STD and range (5e-16 vs 9e-4, 2e-14 vs 0.02) of original and reconstructed data difference even when no dimension is compressed. This might be ICA get a rather good approximate solution rather than exact solution.

N components	mean	std	max	min
7 (10%)	-2.52638e-18	0.155864	1.10706	-0.999113
15 (20%)	-4.12728e-18	0.116829	1.0351	-1.00295
23 (30%)	-1.2066e-18	0.0935518	1.02592	-0.997637
31 (40%)	-2.56472e-18	0.0821237	0.997609	-0.915939
39 (50%)	-3.70971e-18	0.0707766	1.00184	-0.899981
47 (60%)	-3.98854e-18	0.0587326	1.00853	-0.900561
55 (70%)	-3.02061e-18	0.0454444	1.00021	-0.886862
63 (80%)	-3.03613e-18	0.0302946	0.987367	-0.885344
71 (90%)	-3.65632e-18	0.0131774	0.987784	-0.849026
79 (100%)	9.35987e-06	0.000890325	0.0176876	-0.0364204

3) Gaussian Random Projection and Truncated Singular Value Decomposition

For Gaussian Random Projections, n_components can be automatically adjusted according to the number of samples in the dataset and the bound given by the Johnson-Lindenstrauss lemma which is controlled by eps parameter[13]. However, target space goes larger than original even at eps = 0.99999, **the number of components for PCA (36,60) are thus applied on RP** considering PCA as the gold standard of feature reduction.

SVD is also a feature reduction method based on eigen values however it doesn't center the data and uses efficient eigen solver[14]. Similar standard (5% variance loss) is applied to get the ideal n_components: **43 for dataset1, 59 for dataset2**.

RP Repeats:

```
      dataset    Nrepeats    Npairs    pairwise mean(diff)    pairwise std(diff)
---  ---------  ----------  --------  ---------------------  --------------------
PCA          1           5        10               3.25e-17                 0.001
RP           1           5        10              -0.0136                   0.698
PCA          2           5        10              -3.44e-18                 0.035
RP           2           5        10               0.0004                   0.69
```

Holding n_components the same, transformed results of RP vary much more than PCA between runs. In fact, there is a “random_state” to generate a different random matrix for feature projecting every time.

Reconstruction (RP: 100% to 1000% data space, SVD: 10% to 100% data space):

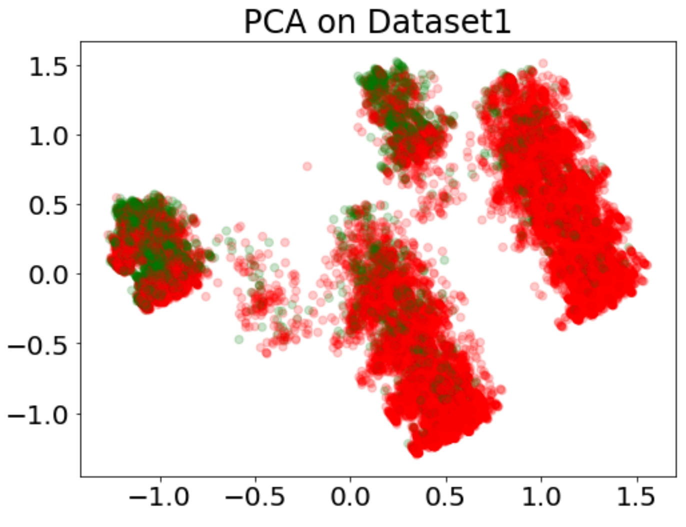
N components	mean	std	max	min
79 (100%)	-0.0891546	0.439493	1.68188	-1.934
158 (200%)	0.00789225	0.315671	1.1064	-1.48574
237 (300%)	-0.0519207	0.24773	0.957342	-1.09446
316 (400%)	0.00641421	0.189863	0.776939	-0.922162
395 (500%)	-0.0423475	0.221773	0.778822	-0.826054
474 (600%)	-0.0117068	0.178676	0.655387	-0.813206
553 (700%)	-0.0015629	0.159118	0.605529	-0.658196
632 (800%)	-0.0181371	0.163575	0.615346	-0.632543
711 (900%)	-0.0418866	0.153219	0.618111	-0.587454
790 (1000%)	-0.000119413	0.138084	0.550905	-0.515014

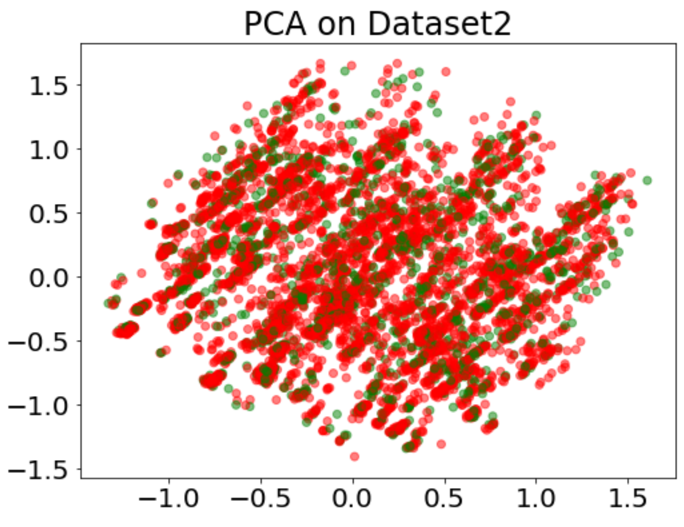
N components	mean	std	max	min
6 (10%)	-4.28654e-06	0.168686	1.17171	-0.966814
14 (20%)	-3.1573e-05	0.124002	1.04167	-0.932826
22 (30%)	-5.53104e-06	0.0969699	1.01104	-0.923991
30 (40%)	1.16372e-06	0.084049	0.997874	-0.839463
38 (50%)	2.54813e-06	0.0728362	0.998413	-0.832607
46 (60%)	4.3558e-06	0.0608254	1.00519	-0.81295
54 (70%)	5.6674e-06	0.047559	1.00457	-0.835442
62 (80%)	2.18284e-06	0.0325622	0.9932	-0.838595
70 (90%)	7.26663e-06	0.0159253	0.987991	-0.610165
78 (100%)	2.46905e-06	0.00139441	0.446156	-0.177343

Considering the low fidelity of RP, up to 10 folds of original data space was tested on its feature reconstruction. Even though the reconstruction actually goes better with more dimensions, it’s still quite bad comparing with other 3 methods (STD, max - min). SVD are better than RP but worse than ICA on dataset2 reconstruction. The loss of information might be pseudo-inverse matrix it uses for calculating singular values. Thus, the reconstruction fidelity rank is: **PCA > ICA ~ SVD >> RP**

3. Re-clustering on Dimension Reduced Data

Visualization of feature reduced data:





Only PCA visualization is discussed due to the limit of paragraph. Points are colored based on their actual label. It’s obvious that the 2D feature on dataset1 is more informative about their label, in contrast differently colored points on dataset2 are most overlapped, which can be confirmed by near zero silhouette score below. Notice the 2 principals chosen by PCA aren’t necessarily to be relevant or useful for later supervised or unsupervised learners. PCA favors direction with most variances, which could come from noise or irrelevant information.

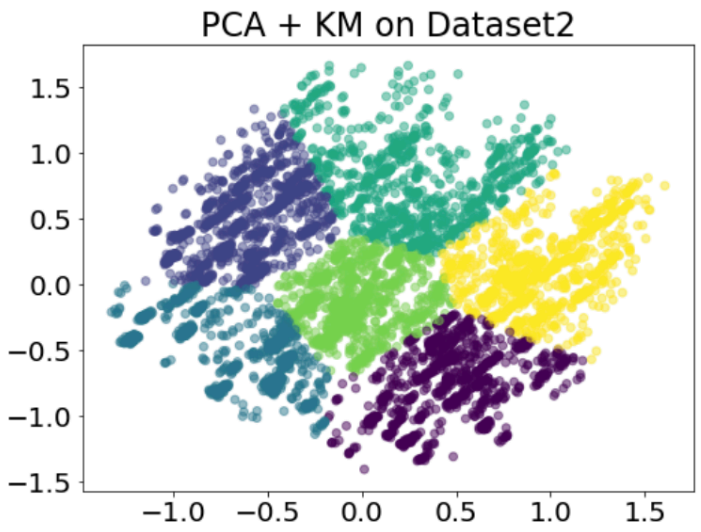
Time cost:

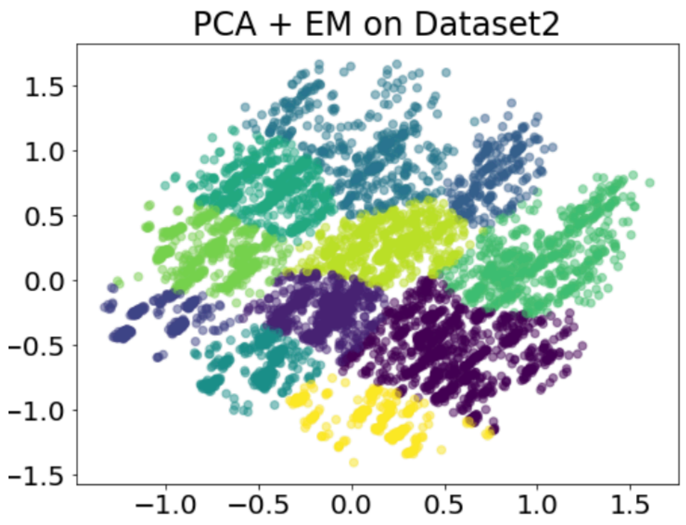
```
       dataset    time(sec)    silhoutte score
---  ---------  -----------  -----------------
PCA          2        0.03               0.008
ICA          2        0.093              0.0082
RP           2        0.002              0.0082
SVD          2        0.024              0.0032
```

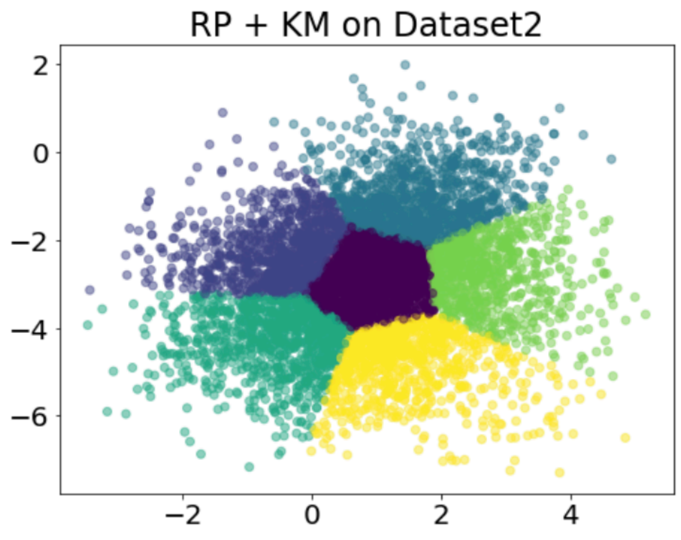
```
       dataset    time(sec)  silhoutte score
---  ---------  -----------  -----------------
PCA          1        0.36   0.1125
ICA          1        1.485  NA
RP           1        0.067  NA
SVD          1        0.23   NA
```

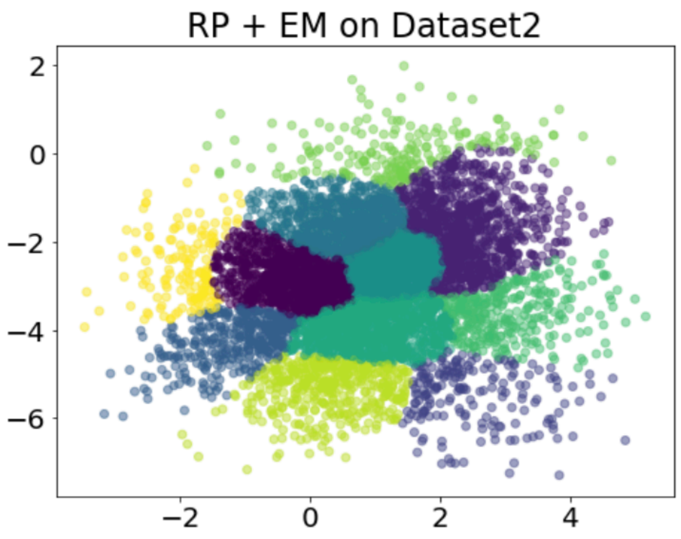
Time costs of running 2-components reducing are recorded. RP is the most efficient methods among all. SVD is slightly better than PCA, while ICA is the worst. The time efficiency is somehow inversely correlated to reconstruction fidelity.

Visualization of re-clustering on feature reduced data:
6 and 11 clusters are used for KM and EM respectively as these are optimal numbers discussed in part 1. Notice that dataset2 (TMDB movie data) is used for re-clustering and training neuron network in part 4 and 5. This is out of storage space and speed consideration as dataset1 has 45K entries and is too expensive for learning purpose. Again, visulizations are done on all 4 methods but only PCA and RP are included in writting due to limited space. Please refer to the 'lwang628-code.ipynb' for full experiment records.









3 patterns can help the understanding of the behavior of algorithms better:

1)PCA and SVD tends scatter data points across the space (most variance); RP’s data points cluster around mean in new feature space (closet to Gaussian distribution); in contrast to PCA and SVD, data points in ICA space densely cluster to each other (most kurtosis), which can be supported by the smallest variation (a side effect of favoring simple form ‘source’ features)
2) PCA and ICA re-center the data, while SVD and RP don’t, inferred by the the symmetry of sample distribution of PCA and ICA around origin. Noticeably, new features generated by ICA are very small in absolute value compared with others.
3) Decision boundary of KM divide the space into a Voronoi diagram; EM shows a mixed pattern resembling both Dirichlet tessellation and overwritten Gaussian clusters.

Time and performance of on KM/EM on optimal sized new features (dataset1)
KM:

```
            sum(dist)                sum     std    MutualInfo    homogeneity    time
--------  -----------  -----------------  ------  ------------  -------------  ------
RP(36)    144117       -15189.5           0.4942          0.06           0.18    0.47
PCA(36)   129000            3.11395e-11   0.3445          0.06           0.18    0.19
SVD(43)   136681       102922             0.4448          0.05           0.16    0.19
ICA(84)       79.1837      -7.14429e-14   0.0047          0.03           0.04    0.39
Ori(103)  141638       425872             0.2774          0.06           0.17    0.54
```

EM:

```
           mean(logP)                sum     std    MutualInfo    homogeneity    time
--------  ------------  -----------------  ------  ------------  -------------  ------
RP(36)         39.8629  -39992.6           0.4941          0.04           0.17   11.92
PCA(36)        31.9051       6.13838e-11   0.3445          0.05           0.2     8.86
SVD(43)        46.1773  102922             0.4448          0.06           0.23   13.16
ICA(84)       428.196       -2.33036e-13   0.0047          0.03           0.1    52.35
Ori(103)      348.077   425872             0.2774          0.07           0.24   62.27
```

Original 103 features are used as a control. Choice of optimal numbers of components have been discussed in part 2. Experiments are ordered here according to feature size (36<=36<43<84<103, etc.).

ICA features have significant smaller sum of squared distances. But it’s not a sign of improved clustering. As we noticed in visualization part, new features somehow get dramatically descaled after ICA projecting, which is supported by small std value here. In fact, clustering on ICA features are worst with regards of predicting power (mutual information) and homogeneity. Not surprisingly, original data are still the best. And more compressed the dataset is, the less computation time required to train the same clustering method.

Time and performance of on KM/EM on optimal sized new features (dataset2)
KM:

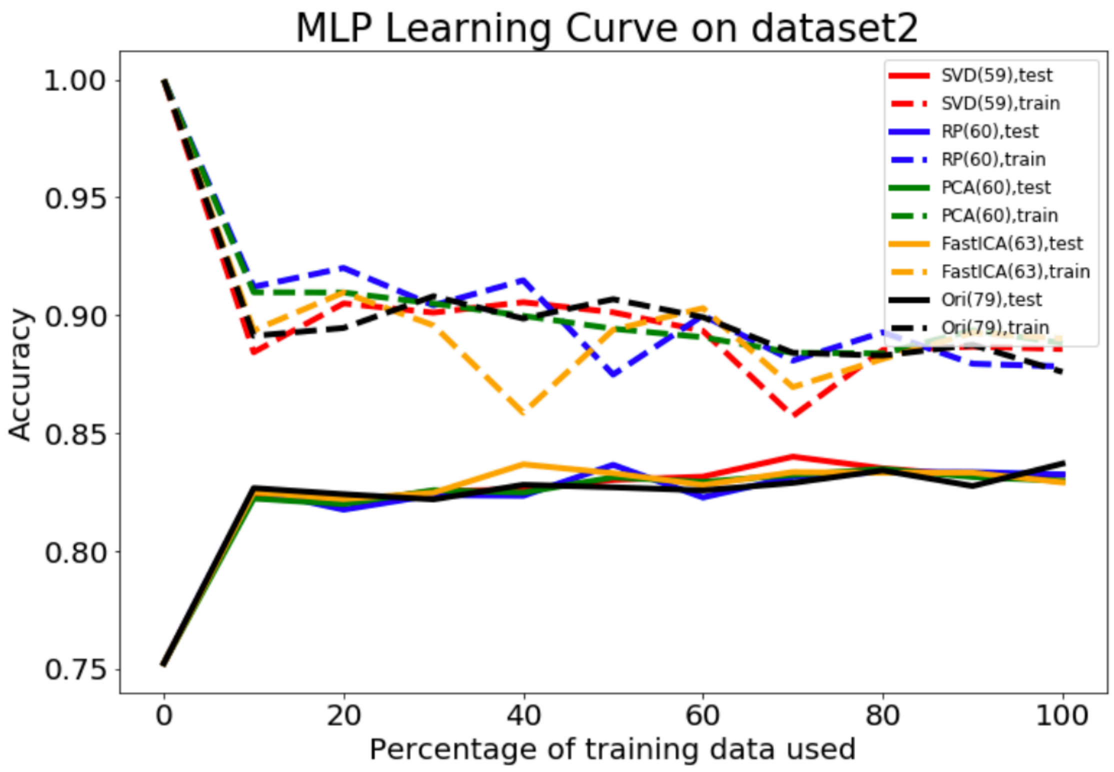
```
               sum(dist)                sum     std    MutualInfo    homogeneity    time
-----------  -----------  -----------------  ------  ------------  -------------  ------
SVD(59)          17234.6    21083.5           0.506             0            0.01    0.05
RP(60)           18333.4    -1090.18          0.4413            0            0.01    0.06
PCA(60)          17394.8       -7.28306e-14   0.2464            0            0.01    0.06
FastICA(63)         59.0783    -1.20792e-13   0.0127            0.01         0.02    0.08
Ori(79)          17932.7   158550             0.2983            0.01         0.01    0.07
```

EM:

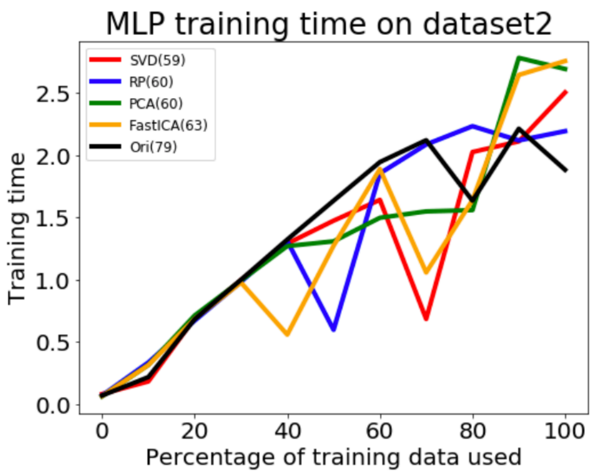
```
               mean(logP)                sum     std    MutualInfo    homogeneity    time
-----------  ------------  -----------------  ------  ------------  -------------  ------
SVD(59)           47.7895    21083.5           0.506             0            0.01    5.94
RP(60)            53.3568  -42150.6            0.5432            0            0.01    9.76
PCA(60)           47.601        -1.56142e-12   0.2464            0            0.01   10.9
FastICA(63)      205.669        -3.29181e-13   0.0127            0.01         0.02    2.31
Ori(79)           94.7666  158550              0.2983            0.01         0.02   15.57
```

Clustering methods do very bad in noticing the characteristic we care (the label). After adjusted against chance, the mutual information is almost 0 between predicted cluster and labels. ICA performs less terrible than other methods, which tells us the story again that in practice we should choose algorithms most suitable for the problem.

4. Neuron Network on Re-projected Dataset2



The MultiLayerPerceptron with the structure optimized in assignment 1 is trained on re-projected dataset2. It's performance on original 79 features dataset is used as control (colored black).

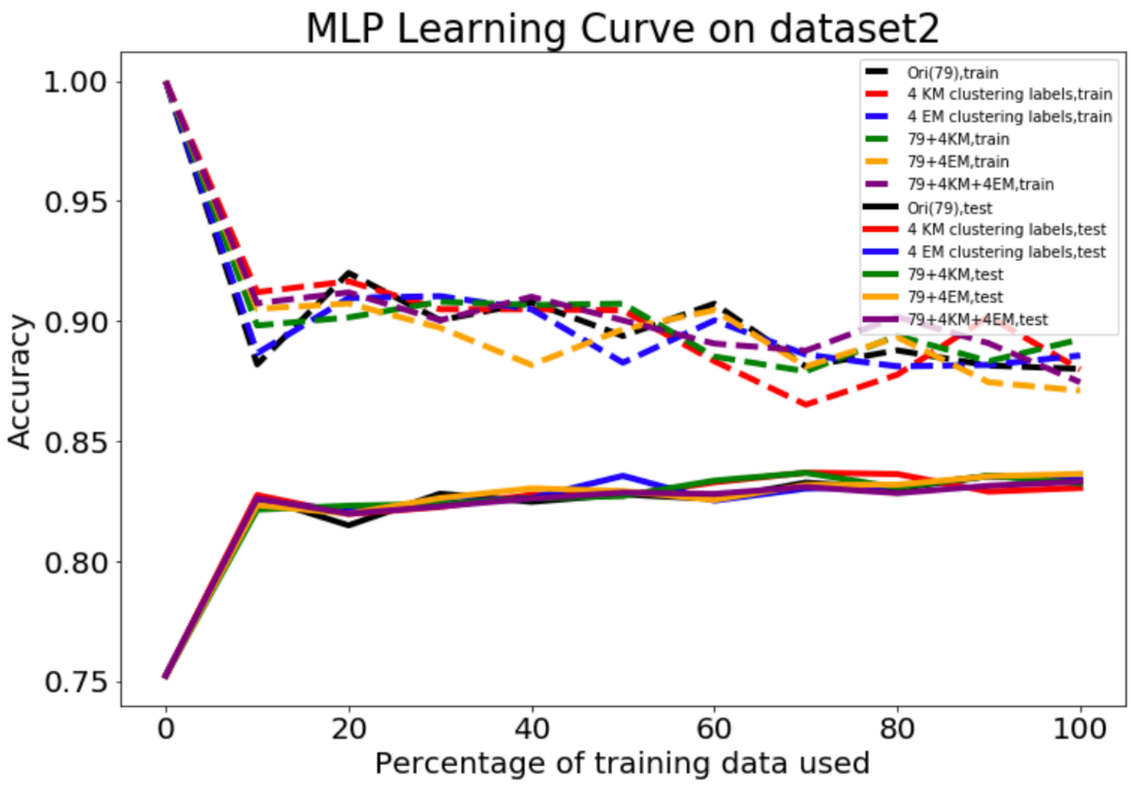


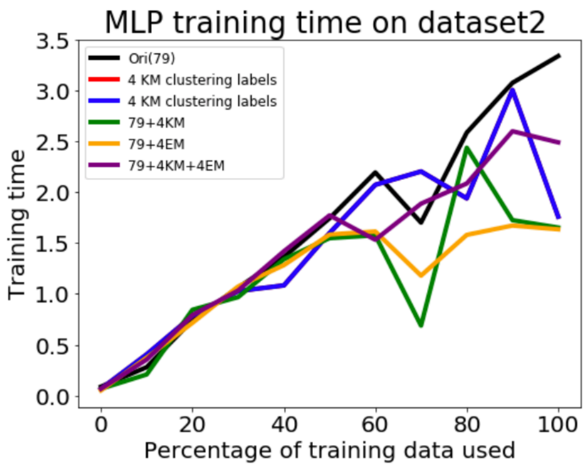
Information Loss:
Compared to original data, transformed data didn't suffer much from information loss, as conservative strategy is taken in choosing n_components (95% variance).

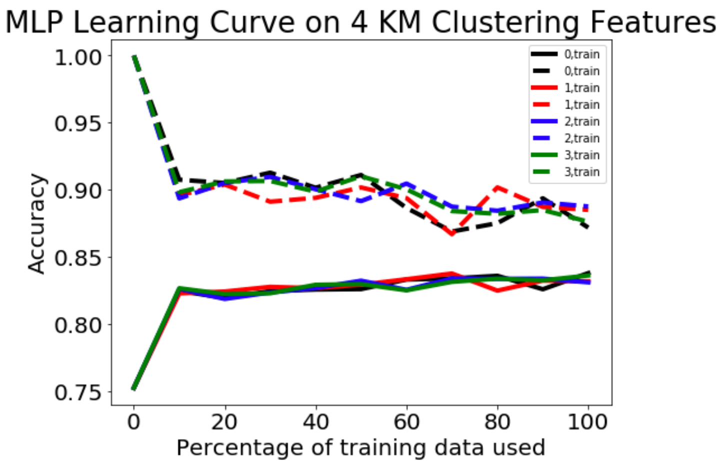
Time cost:
Transformed data didn't improve much in MLP's training speed. But it might because the compression power isn't big enough to elaborate the difference here.

5. Neuron Network on Re-projected Dataset2 and Clustering Features

MLP built on 4D (KM/EM clustering on PCA, ICA, RP and SVD), 83D (original features + KM/EM clustering features) and 87D (original features + KM + EM clustering features). Black curve (original features) is used as control.







Information Gain:
Purple line (87D) didn’t perform better than reference. However, it’s interesting that only 4D clustering features can do just as good as the full dataset. In fact, even MLP built on 1 clustering feature (either one, as shown in the right graph) can do similarly good, which is hard to believe and maybe have something to do with nature of redundant expressiveness about MLP.

Time cost:
Surprisingly, MLP training time didn’t scale down with reduced input numbers (4D clustering features take similar time with 87D all-inclusive data), which implies that a big chunk of data isn’t used between iterations. It would be helpful to look at how other supervised learners respond to differently projected and clustered data.

Reference:
(Plz check the readme.txt file)