CS7641 Unsupervised Learning HW3

Aleksandr Plokhikh (aplokhikh3@gatech.edu)

Introduction, dataset selection, data preprocessing

The goal of the assignment is to study unsupervised learning algorithms such as clustering, and feature transformation + selection, and study how the last can help with supervised learning (NN), and how the substitution of the labels with the ones from clustering affects supervised learning and possible fixes preexisted mislabeling.

For the task, I take the same fruits [1], and phones dataset [2] which are interesting for several reasons described in A1, but here for A3, we have a new dimension of interest. Fruits have pretty determined classes of different fruits with genes of the fruits, but in the case of phones, the cell phones just divided into 4 price ranges, and we do not know how arbitrary is the division of classes, we can easily more classes, and do not break any logic, but it will be nice to see what is the reasonable number of price ranges(clusters there).

I did data preprocessing for both datasets (linearly scaled all features from 0 to 1 where 0 -is the minimal feature value in the set, and 1 – is the maximum). This is an extremely important step for A3. It is very important for determining distances for K-means, reasonable expectation calculations for EM, and very important for some feature transformation techniques especially crucial for PCA! For example for the phones dataset variance of the phones in battery capacity in mAh is 2-3 orders of magnitude higher than the difference in CPU clock speed in GHz, and there is an obvious difference in importance for customers. For example, an average consumer may not see the difference between phones with 3502mAh and 3500mAh difference but the corresponding difference between 3.5Ghz, and 1.5GHZ of CPU clock speed will be visible day and night especially if we take into account that phones with a 3.5Ghz CPU most likely will have more cores, and different architecture with wider pipelines which are not usually listed in phones spec list but have a very substantial multiplier on performance even further than the simple ratio of 3.5Ghz/1.5GHz.

If we do not implement scaling then the PCA on phones dataset will decide that the principal components are battery capacity in mAh, mass of phones in grams, and length of it in mm or some linear combination of them since there are obvious positive covariance of these parameters. They will dominate the process while they are not the most crucial features. Clustering will be also spoiled and will divide phones into clusters based on battery capacity, weight, and length ignoring another aspect with Euclidean or Manhattan distancing in K-means or in EM. The same situation will be in the fruit dataset but with other parameters like mass, and length of the fruit. Based on my knowledge I can parse reasonable scaling of the parameters but it ruins the idea of unsupervised learning, I can not pass domain knowledge to the algorithm, and just do the best most simple knowledge-independent scaling of each parameter from 0 to 1.

Part 1, Clustering

On the fruits dataset, most simple unsupervised standard square error (SSE) performance measure methods show that with an increasing number of clusters for K-means method of clustering gradually decreases, which does make perfect sense. The plot does not show any noticeable inflection point. SSE is not a good method for the database. The unsupervised Silhouette method on the contrary shows very prominent inflection on a number of clusters equal to 11 for K-means, and 10 for EM. Any further increase in cluster numbers does not make sense, especially for the EM algorithm since the metric does not go lower anymore. The supervised homogeneity score does show inflection at the number of clusters equal to 7 (the real number of clusters) for both methods, the homogeneity score is also high overall which says that both clustering method produces clusters very close to the right labels. The larger number of clusters offered by the Silhouette method may say that for larger fruit types (linear size and weight) very is some variation interclass variation in size which causes extra clustering. Small variations of big values in one class for K-means, and EM are as significant as many variations of small parameters in different classes. It must be the issue of data scaling but I did not have a right to offer a better scaling method without parsing additional domain of knowledge to the algorithm. K-means algorithm ran for a maximum of 2000 iterations and 10 restarts while EM ran for a maximum of 100 iterations and 5 restarts and even with a much smaller number of possible iterations EM took 531ms, while K-means only 65ms. EM is a much more expensive algorithm to run and the high dimensionality of feature data makes things even worse.

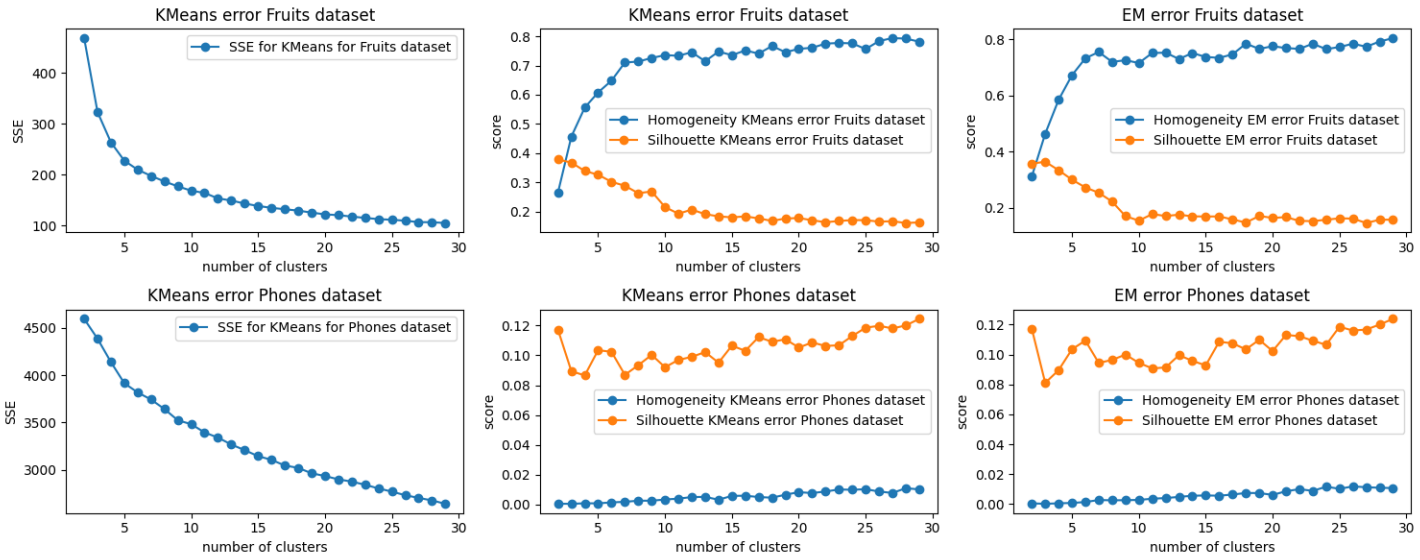

Fig. 1 K-means and EM performance of fruits and phones datasets.

On the phones dataset SSE, the graph has a prominent knee at n =5, and less prominent at n=9, so the most reasonable clustering according to SEE for phones is n=5 since it is simpler clustering we use Occam's razor here and does not multiply instances without need, also inflection point on n=5 is more distinct than at n=9 which look more like a small deviation from the trend. Silhouette method proposes clustering with 4, and 3 classes for K-means and EM respectively. Any increase in the number of clusters leads to an increase in the Silhouette score -not efficient. The mean decision here (for K-means SSE, K-means Sile, and EM Silhouette) is 4 which corresponds to the number of classes in the dataset so the number of classes ready does make sense here. The Homogeneity score almost monotonically rises with the rise of clusters but remains very low (less than 0.02) it may be said that both K-means and EM divide the dataset into some clusters that are not associated with price. The low Homogeneity level may be caused by noise in the data and the fact that the actual pricing of the phones is not always objective, but depends on the marketing, brand position, and current market situation, and we do not have such info in the database. K-means algorithm takes 32ms to run on the dataset while EM takes 92ms for 4 clusters spilled. There is a 3-fold difference in time but it is not as dramatic as in the case of the fruits dataset where we had 9 times difference for 10 clusters. Most likely this difference directly arises from the higher dimensionality of its data (34 vs 20) and the higher number of clusters to split in the case of fruit data, since EM is the "soft clustering" technique and for each point we need to calculate probability relation to each cluster and if we have roughly 3 times more clusters it means that we need 3 times more time!

Part 2 Dimension reduction techniques

PCA and ICA are well-established algorithms, and they perform equally well on both sets of data in terms of reconstruction error, a minor difference between decompositions occurs when the number of algorithm features becomes equal to the number of features and reconstruction error drops to the lowest values but this setting does not make to much sense since we do not reduce the number of features, noise and do not fight the dimensionality problem here which says that we need to exponentially increase the number of entities in training sample to get information out of it.

The reasonable number of features of the fruits dataset for PCA and ICA I chose to be 20 since at this setting the reconstruction error already drops below 10^-4 and it offers very reasonable features reduction over the default 34 dimensionalities in the original dataset.. At n around 30 reconstruction error settles at values of 10^-13 which is exceptionally low and tells about the very high efficiency of methods on the dataset In the case of the pones dataset the reconstruction error drops not that rapidly but more in a linear way and after a set of 17 features the reconstruction error does not fall dramatically so I chose this number to still have benefited over 20 dimensionalities in the original dataset. The reconstruction error of both methods on the phones dataset is orders of magnitude higher than on the fruits dataset which tells us that both PCA and ICA are not as exceptionally good here as on the fruits dataset. It may happen because phone databases have a lot of noise, not clear, and non-linear relations within the data. PCA takes 16ms of run time on

both datasets, ICA -13ms on the fruits dataset, and surprisingly less than 1ms on the phones dataset, it may only say that the ICA decomposition there is very straightforward even though the reconstruction error around 10^-3.

Reconstruction error for random projection error gradually decreases with the rise of dimensions number. The reconstruction error is many times higher than in the case of PCA and ICA methods for both datasets. For the fruits dataset, the difference is order magnitudes. The reconstruction error as in the case of PCA, and ICA drops to the lowest values when we reach the dimension of the original dataset. For all feature transformation methods, I used the same number of dimensionality (20 for fruits, and 17 for phones) for better apples-to-apples comparison. to have comparable noise, and convolution reduction capabilities in later parts of the assignment. The random projection method unsurprisingly takes less than 1ms to conduct calculations.

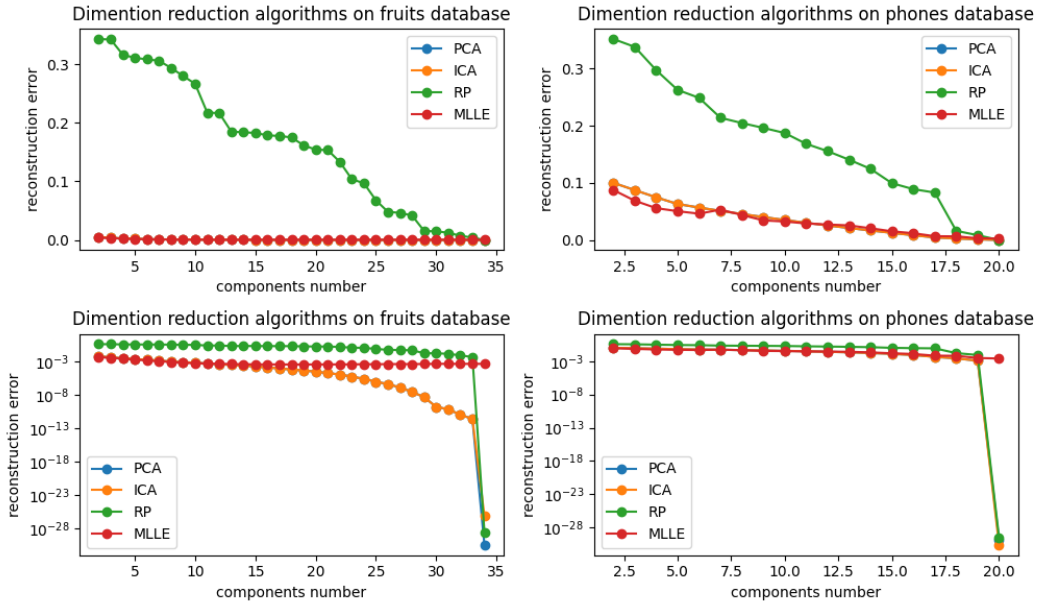

Fig. 2 Feature transformation methods performance

I tried all manifold learning techniques and all of them are very time-consuming, and often require orders of magnitude longer computation times than PCA or ICA. It happens for LTSA, MDS, Spatial Embedding, and TSNE, which is why they did not come on the shortlist as impractical. Hessian embedding was not only very slow but comparable with PCA, with ICA dimensions number equal to 20 for the fruits data set and 17 for the phones dataset we had to calculate embedding for a very large number of neighbors it is not only a very time-consuming but for fruits dataset calculation of the neighbor list already takes a significant portion of all entries and for 20 components the neighbors of the entry is already more than half of all entries.

Default LLE is the fastest manifold learning but still, it takes 420ms on fruits database and 1.15s for phones which is higher than PCA or ICA run times, while performance is poor. Isomap showed similar performance as LLE but ran 3 times longer than LLE. I chose MLLE with 50neighbour calculation since it increases calculation time very insignificant in comparison with default LLE (to 466ms and 1266ms for fruits and phones datasets respectively), while increasing performance. Still despite all efforts, MLLE did not beat PCA and ICA on my datasets. On the Fruits dataset MLLE went neck to neck until the number of components rose to 15 after that point PCA, and ICA continued to fast decrease reconstruction error while MLLE could not maintain the same pace. On the phone database, MLLE was as good as PCA, and ICA, and for low component numbers of 2-6 even offered better reconstruction errors but, such low dimensionality is hardly practical since they have a high reconstruction error of more than 0.05, and the number of components may not be enough for sophisticated further learning algorithms to retrieve information. It is worth mentioning that when we increase component number to the number of features in the original dataset the reconstruction error did not drop to the very low error it may be the cause of non-linearity and locality of the MLLE, it may not provide a very low reconstruction error on the whole space of the data due to the fact.

In short PCA, and ICA are equally good on both datasets, ICA takes less time to learn, and RP may be the fastest but a high reconstruction error makes its implementation not justifiable PCA, especially ICA on phones dataset does not take a lot of time to process. MLLE does not provide a performance boost compared to PCA/ICA but takes much more time to process. Other manifold techniques take even more time or show worse performance without significant speed-up gain.

Part 3, Clustering of processed datasets

In this part of the task, it is interesting to see how different dimensionality reduction techniques help with clustering or make it more difficult. Does it help or prevent finding the most appropriate number of clusters? Alo comparison will be presented on Fruits dataset since on the untreated dataset the Silhouette score very nicely showed the best clustering approach, and I will treat it as my best benchmark, and it will be also nice to see if feature transformation techniques will offer a right number of clusters equal to 7 (classes in original dataset) instead of EM+Silhoutte result of 11 clusters. All dimension reduction algorithms reduced components from 34 to 20 for better performance comparison.

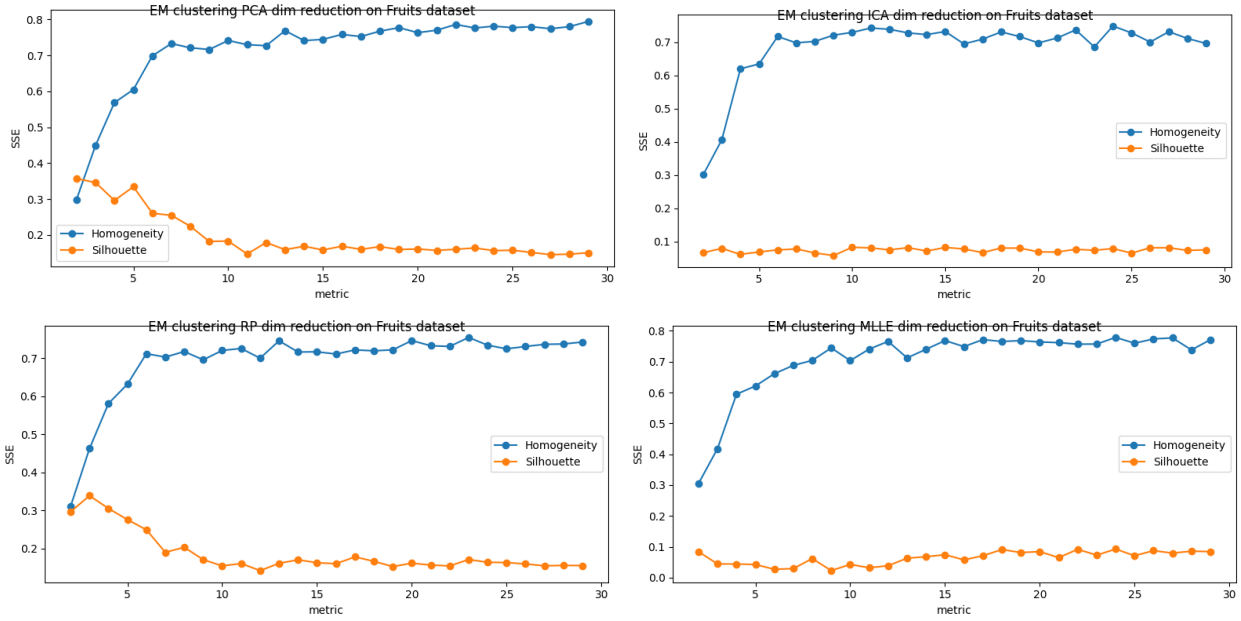

Fig. 3 EM clustering on fruits dataset after dimensionality reduction

The most mediocre result showed the MLLE features transformation method. Silhouette score does not clearly show optimal clustering, maybe n=9, or n=6 would be the ideal dimensionality count. Silhouette score even rises with an increase in cluster numbers, which is a sign of poor and inefficient clustering. While the homogeneity score is close to 0.75 for high dimensionality it does not show the same number of clusters as some labels in the original dataset equal to 7 and votes more for n=9. MLLE is not a winner here and does not grasp the essence of the set despite reasonable reconstruction error in part 2. EM algorithm with such data transformation takes the longest time to build the “clustering curve presented in Fig.3 (18.7 seconds) which says that the EM algorithm needs more iterations and results to come to a convergence.

The second least performing algorithm here is ICA it also does not have a good “knee” inflection on the Silhouette graph and “votes” more towards n=9 or n=4 which is close to MLLE predictions. Silhouette score here does not rise with the increase of the cluster number which is a good sign. The Homogeneity curve predicts 6 clusters which is a much closer number to the number of real classes in the data set equal to 7. The homogeneity score is not the highest overall but respectable and remains around 0.7 at high cluster numbers. The EM algorithm takes a bit less time to build the “clustering” curves – 16.7s which says that the EM algorithm comes to convergence a bit faster than in the case of MLLE pretreated input data.

RP is just a random linear combination of dimensionality hence the performance of clustering should be similar to the clustering on the original dataset but slightly worse due to the lower number of dimensionality and due to building linear relationships between components which most likely are not present in the dataset, and indeed than Silhouette graph

have a knee, as an original EM Silhouette graph on the untreated dataset but a bit less prominent and on a slightly higher clustering number equal to 12, instead of 10 as in original clustering. Silhouette score does not rise with the increase of cluster numbers, which indicates reasonable clustering. The homogeneity curve predicts 6 clusters the same way as in the case of ICA, which is still 1 less than the number of true classes in the data set. The homogeneity score keeps a minor uptrend at higher cluster numbers and reaches values higher than 0.7. The EM run time is even lower than in the case of MLLE, and ICA pretreatments and equal to 14.5 for building a full "clustering curve". The EM algorithm comes to convergence here faster than in the case of ICA, and MLLE algorithms.

PCA is the winner here. Silhouette graph has a very prominent inflection point at n=11, which is almost the same as in the EM clustering with original data where was equal to 10, it is also equal to the best K-means clustering on the untransformed database. The homogeneity score predicts the same number of clusters as true classes, the score rises even further with an increase of cluster numbers and reaches values of close to 0.8 at higher cluster numbers, which are signs of good clustering. The EM algorithm takes the lowest time to build the curve (11.3s) which is significantly lower than any other aforementioned features transformation techniques, and the gap to the closest competitor – RP is larger than the gap between 2nd and 3rd place and 3rd and 4th places. It indicates that it is much easier for the EM algorithm to come to convergence and produce good clustering than with any other clustering techniques. The run time of the EM algorithm is even lower than on the unprocessed dataset which was 13.7s for the full "clustering" curve, but such a comparison does not make sense since the original dataset has 34 dimensionality while PCA treated only 20 which reduces the calculation time of a single step of any clustering algorithm on the dimensionalities reduced dataset, but it hints that on average EM algorithm need fewer iterations and restarts to find good converged clustering than with any dimension reduction techniques.

I want to point out that the performance of the EM clustering on the Fruits data set increases in the raw MLLE, ICA, RP, and PCA, and the clustering run time monotonically decreases. The good preprocessed data gives better clusters and it happens faster as well, so we sooner reach the convergence. The second thing that I want to point out is the same quality outcomes as well as run-time results I see also on the K-means method on the Fruits database, and EM, K-means algorithms on Phone datasets. In all combinations of clustering techniques and datasets. Performance of the chosen algorithm increases in the raw MLLE, ICA, RP, and PCA, while the run time decreases, and we can conclude here that the combined podium looks like this: MLLE-4th place, ICA- 3rd place, RP- 2nd, PCA-1st for all important metrics (performance, and runtime) regardless of the weight of the metric in the total score.

Part 4, Neural net performance on density reduced datasets

One of the main points of feature transformation and dimension reduction is to help other learning techniques with dimension problems which says that we have an exponential rise of required data entries with the rise of feature count if we want to extract information from our dataset.

I performed this task on the fruits dataset since in=t has more classes 7 vs 4 in the phones dataset hence it should be a more difficult task for network and underlining dimension reduction algorithms to produce good results. Also, the fruit database has many (34) components and not many entries (898) to properly satisfy the dimension of the dataset so it is a proper candidate for dimensionality reduction pretreatment. I used a 25x25 network with a constant learning rate of 0.01 and "relu" activation function since it was the best parameter for the NN on the untreated database and our dimensionality reduction algorithms need to fight against this benchmark. All dimension reduction algorithms reduced components from 34 to 20 for better performance comparison.

The learning curve of NN on an untreated database clearly shows dimension problem of the set-learning process is "rough" with two falls on the validation sample learning curve which indicates that NN learns noise as true data. There is not much room for improvement with the increase of the entries data since the validation curve almost reached the train learn curve. It also indicates that NN learns a lot of noise, and each individual feature may not be ideal. The weighted average f score of the NN on the test sample is 0.8876. Overall NN shows all the signs that it needs dimensionality reduction.

PCA algorithm on the dataset proved to provide the lowest reconstruction error on pair with ICA and was the best helping algorithm. The validation sample learning curve has only one small performance drop with an overall rise of the f-score during the learning process. It is a sign that NN learns real information and less noise. Overall performance of the validation sample curve is higher than the original untreated one which is also a good sign of PCA usefulness here. The tangible difference between validation and training samples curves indicates that NN has good room for performance growth with extra training data but even with the current training set the performance of the NN on the test sample is higher than that of NN on untransformed input data and weighted average f sore is equal to 0.9214, which prove the usefulness of dimensionality reduction for fruits dataset.

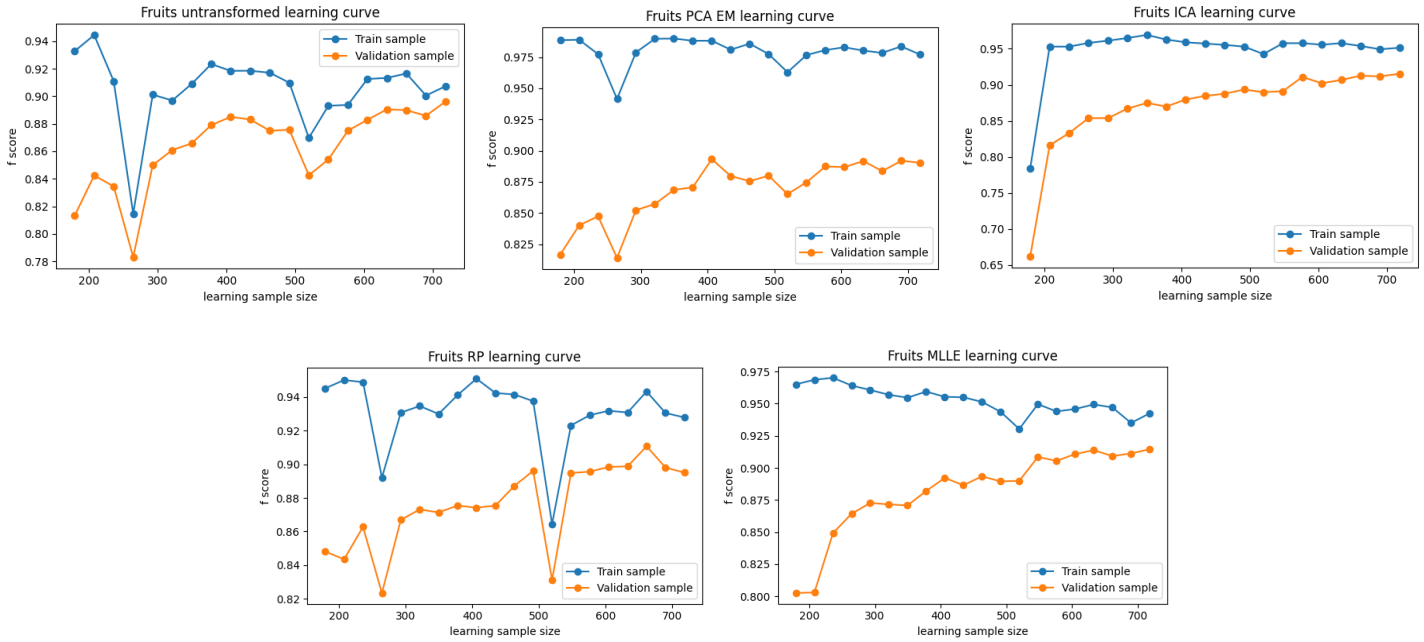

Fig. 4 Learning curve of NN on density reduced fruits datasets from left to right (untreated set, PCA, ICA; RP, MLLE)

ICA was one of the best in terms of reconstruction error but did not help EM clustering. The validation sample learning curve is very smooth and uprising which indicates that ICA does a good job eliminating noise dimensionality. The f-score on the validation sample is higher in general than the one on the original NN which indicates better performance of the NN with such dimensionality reduction treatment. Unfortunately, NN does not have much room for performance increase with additional training data since the gap between training and validation sample learning curves closes up at high training sample sizes, but it still manages to beat the original NN on the test sample with the weighted average f-score of 0.9124 which is slightly behind the PCA performance.

RP learning curves very closely resemble the ones for the original NN including overall shape, drops, and numbers, and the performance on the test sample is also very close- the f-score is 0.875 -significantly lower than in the case of the original NN (lower info for NN most likely influenced this small score reduction). The results do make sense since RP creates just random linear combinations on features, it does not make some oriented sophisticated features transformation hence we should not expected some radical changes in the performance of the new NN, and we did not get it.

MLLE showed a reasonably low reconstruction error but failed on the clustering test. The learning curves are smooth like in any “sophisticated” feature transformation techniques such as PCA, and ICA, which tells us that MLEE does help NN not to learn noise in data and focus on reasonable information. The NN trained on MLLE has limited room for improvement with additional data (based on the proximity of train and test sample learning curves). Unfortunately, MMLE has only a marginally positive effect on NN performance on the test sample where it showed a weighted f-score of 0.889 over 0.8876 for the original NN, such boost is statistically insignificant taking the size of the test sample equal to 180 entries and does not justify the time spent to dataset transformation (466ms) while the NN took only 93ms to train.

Timewise dimensionality reductions do not simplify the NN significantly since it removes only (34-20)*25=280 connections, while the original NN has 34*25+25*25*(25-1)*25*7=16025 connections. All NN train in about 91-93ms including the original one, except PCA “pretreated” NN which takes only 71ms to train which indicates that NN comes to convergence faster and this fact correlates well with the fact the PCA NN showed the best performance and largest area for improvement.

Part 5, NN on dimensionality reduced datasets with unsupervised labels.

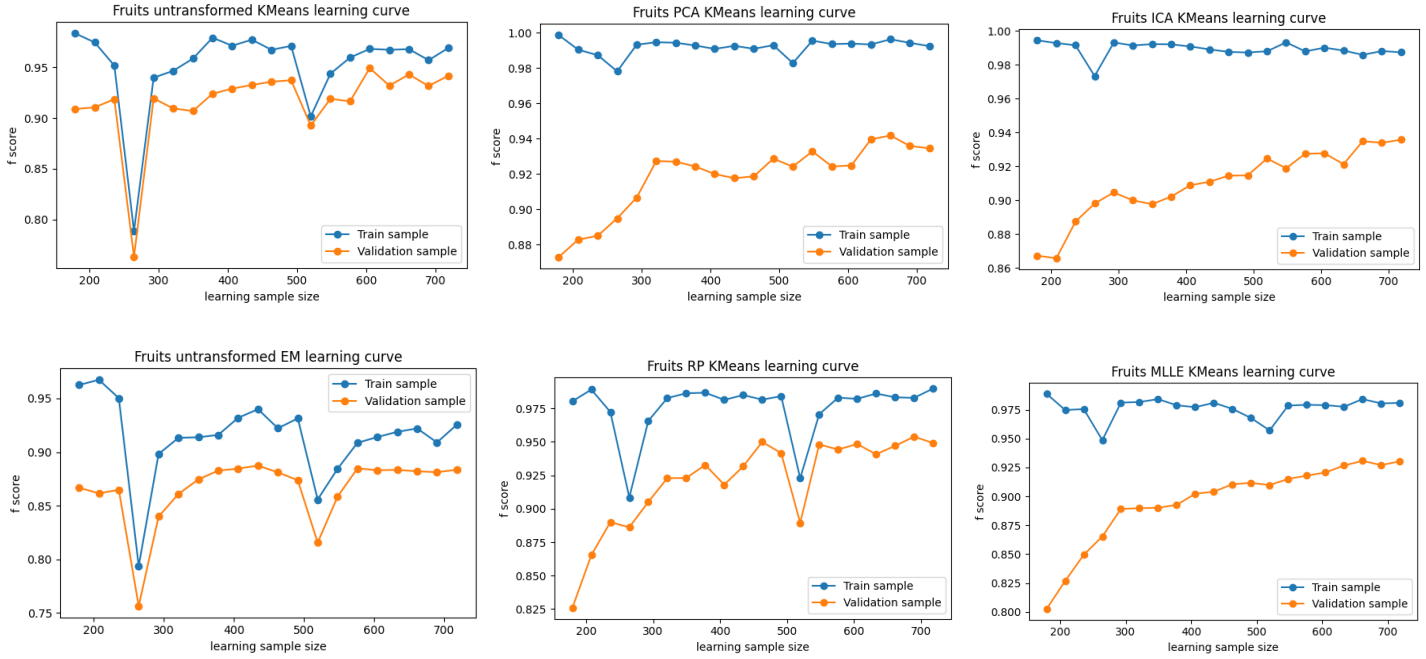

Fig. 5 Learning curve of NN on density reduced fruits datasets with labels substituted by K-means, and EM clusters

This part of the task was executed the same way as the previous one but in this case, all labels were substituted by clusters. I used 10 clusters model for both EM and K-means clustering methods as the most appropriate value according to its Silhouette score (see part 3), to make more apples to apples comparison between clustering techniques and make the problem harder than in the case of default 7 label classes.

The NN learning process with new labels but with untreated features very closely resembles the process with default labels with the same two fall on the curve which corresponds to the peculiarity of the splitting, and cross-validation process with a parsed random seed. The cross-validation f-score here does not rise that much during the learning process indicating that NN trains faster than with the original labels, there is also limited room for performance improvement dictated by train set learning curve position like in the NN with default labels. The train, validation, and test sample performance is at least as good as with the default labels, it is an interesting fact given into account that now we have a higher number of classes and a harder task to guess the outcome. It is also interesting that K-means clustering showed much better performance than the EM method on all train, validation, and test samples, and yielded a weighted average f-score score of 0.9497 while the corresponding f-score for EM clustering is 0.877 which is on par with the NN trained on default 7 labels. Such high performance may indicate that the default labeling is not ideal, and may have some mistakes.

K-means consistently outperforms EM on all feature reduction techniques. Originally I thought that it had something to do with abrupt but fast RELu activation faction, which may prefer hard clustering, but smooth logistic function did not make EM winner, I also proposed that EM may not have enough iteration so I increased maximum iterations from 100 to 2000 like in case of KMean, but it did not change the run time, and performance of the NN which indicates that EM converges faster than reaching hard iteration limits. Maybe it is something to do that fruits are actually very different, have different genes, and hard clustering such as K-means should work better, than the soft one. Here we just left to enjoy better accuracy and faster run time of the more simple K-means method. Solid win of the algorithm.

PCA features transformation/reduction, makes the learning process, and cross-validation learning curve smoother for both EM and K-means clustering, which indicates that NN does not learn nose too much. There is a huge area for performance improvements since the f-core on the training sample is around 0.99 and 0.97 for K-means and EM respectively. The weighted average f -score is 0.9888, and 0.9102 for K-means and EM clustering, which is a very significant performance boost over untransformed components space. PCA proved it's high efficiency here.

ICA transformation makes the learning process even smoother with a constant gradual increase of performance with an increase in the training sample size. The NN has a very large potential for improvement - the f-core on the training sample is around 0.98 and 0.93 for K-means and EM respectively, which is less than in the PCA case but still very high, and higher than for NN performance on the untransformed database, and it transforms to the test sample performance right in between the NN trained on untreated features set and PCA with average weighted scores of 0.9556, and 0.8953 for K-means, and EM respectively.

RP learning curves closely resemble the curves of the NN trained on the untransformed dataset. We already saw it similar before, and it most likely arises from the fact that RP has the smallest transformation of the input data space and just makes random linear combinations of initial components. The NN still has some reasonable room for improvement with increasing training size – on par with the NN trained on the original components set. The performance of the NN on the test sample is slightly worse than the result of "vanilla" NN – the weighted average f-score is 0.9108, and 0.8480 for K-means, and EM clustering respectively. This result is expected – RP does not really remove noise from the data, just reduces the volume of information which damages the performance.

MLLE does make the learning process smoother here, but the outcome is debatable on K-means clustered data frame performs exceptionally well with the weighted average f score on the test sample of 0.9888 -on par with PCA, while with EM clustering the f-score drops to only 0.8331 which is significantly less the result of the "vanilla" NN trained on the untransformed dataset. Most likely the dataset does not have significant non-linear relations which the methods look for.

Overall all feature transformation techniques behave the same way as in the case if we use default 7 class labeling from the dataset, all relations are preserved, which is expected, Nice to see that clustering increases the performance of all NNs despite the increase of the label class count. All NN with pretreated features have a higher potential for performance increase than the original NN which is trained on an untransformed features dataset.

Time-wise all NNs took comparable time to train from 61ms to 131ms to train the neural net. And the run time is highly correlated with the performance of the NN. Nets with the higher performance took less time to train – they simply converged faster, and more reliably. For example, the shortest train time of 61ms corresponded to K-means PCA NN which showed the best f-score on the training sample of 0.9888, while the closest to train NN is EM MLLE one which took 131ms to train, and had the worst f-score on the test sample of 0.8331.

Conclusions

K-means hard clustering was faster and better than more complicated EM, especially in the objective task of label creation for NN, K-means also possibly fixed mislabeling in the original dataset, and showed great performance for label creation. PCA features transformation method was the best almost everywhere and proved to be a viable technique to boosting NN performance, RP did not have any improvements and did not change any performance, and behavior of NN to much, and expectedly had the highest reconstruction error. All the behavior comes from the fact that RP is just a truncated random linear combination of components without sophisticated changes under the hood. Manifold techniques were very time-consuming, and the best technique from all of them on my dataset did not offer any performance improvement over PCA or ICA in any performance metrics such as reconstruction error, f-score, or run time. The dataset most likely do not have prominent non-linear relationships.

Citations

[1] https://www.kaggle.com/datasets/muratkokludataset/date-fruit-datasets
[2] https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification