Assignment 3:
Unsupervised Learning and Dimensionality Reduction

Oindril Dutta | odutta3@gatech.edu

1 INTRODUCTION & DATASETS

The goal of this assignment is to take a deep dive into unsupervised learning and dimensionality reduction, exploring a few methods of doing so: Clustering, through K Means (KM) and Expectation Maximization (EM), which I "implemented" through SKLearn's Gaussian Mixture, alongside Dimensionality Reduction (DR), Principal Components Analysis (PCA), Independent Components Analysis (ICA) implemented using SKLearn's FastICA, Random Components Analysis (RCA) implemented using SKLearn's SparseRandomProjection, and lastly Latent Semantic Analysis (LSA) implemented using SKLearn's TruncatedSVD. As hinted, my library of choice for this project is SKLearn, and the datasets I picked are the same as assignment 1. My stock dataset is not a great dataset when it comes to learning, but it has shown to be very helpful in comparison between the different methodologies discussed below.

I started by implementing all the different methodologies listed above, doing no tuning for the different methodologies. To compare clustering, I decided to look at accuracy, since my datasets are labelled and I can get a % score. To compare all the DR algorithms, I compared across a few different metrics, namely the error, the kurtosis, and lastly the distance as an average across all the number of components per count of components. Also applied was a random seed for everything based on my ID and the main function to keep analysis consistent.

1.1 Stock

For assignment one, I use the data set of stock data - I'm going to reuse that data set. After working with stock data for a whole semester in CS 7646, I had an idea - instead of doing statistical models over the last n days to get an idea of whether the price will go up or down, why not train the model directly on the ebb and flow of the price for the last n days? So I created a custom generated data set that takes the last n days of closing prices and transforms them into a row, with a label based on whether the price of the next day goes up, down, or holds. Depending on that it gets a signal of BUY, SELL or HOLD. For this assignment specifically, I take the entire stock history of general electric and transform it into almost 14,000 rows of 60 day closing prices, and the label is based on the 61st day price. I do one-hot encoding on the signal labels for MLRose. Since my last time using this dataset in Assignment 1, I made a few changes. I've added a third label of HOLD which wasn't there before, which I believe contributed to the poor performance in that assignment. I also increased the number of days from 15 to 60 to make it a better fit for this assignment, that pertains to clustering and DR. It is not used for the neural network because all the comparisons across the neural networks are based on loss and training/testing accuracy,which this dataset is not the best at. Another note, I decided to make another variation of this data set because the prices across all of the rows are constantly increasing over time. As the total price of general electric raises over time, I decided to make an alternate version of the stock data set to see if it performs any better

across all these algorithms by normalizing each of the rows relative to each of the rows so that price changes over time don't affect the models ability to learn.

1.2 Yoga

For Assignment 1 & 2, I used a dataset of Yoga Poses - I'm going to reuse that dataset. Around 2000 rows of angles & magnitudes of human body joints doing yoga poses, labeled as either Warrior II, Triangle, Tree, or None. The closest analogy of this data set I can draw is the Iris data set, which I used as an inspiration to create this simple dataset. Both datasets' features are the physical measurements of the thing it's trying to classify. For example, the iris dataset measures petal count & lengths, I measure joint angles and magnitudes lengths.

The first 12 features of the data are the angles at the joints of the human body, such as the angle between the forearm and the upper arm at the elbow. The last 12 features of the data are the magnitude lengths between the center of the body and each joint of the body, which I calculated by getting the center of mass and then measuring the euclidean distances to the 12 joints. The labels are 0 - 3, with 0 as None (no pose, random pose) with 900 rows ~43%, 1 as Warrior II pose with 360 rows ~17%, 2 as the triangle pose with 343 rows ~16%, and 3 as tree pose with 459 rows ~24%. If you google these poses, you'll see I picked them because all of the joints of the human body in the pose are visible in 2 dimensions from a webcam.

For all the clustering and DR algorithms, I don't do any test splits on the data, but for the neural network I do a 75 - 25 split after shuffling all the rows across the training and testing set to ensure an equal split. I used ML Rose again for this assignment and I didn't get a chance to do any cross validation. Fitness curves were plotted over iterations to compare the different algorithms. MLRose also required one-hot encoding of the labels. To get more details exactly on this dataset, please refer to Assignment 2 for more details on this dataset.

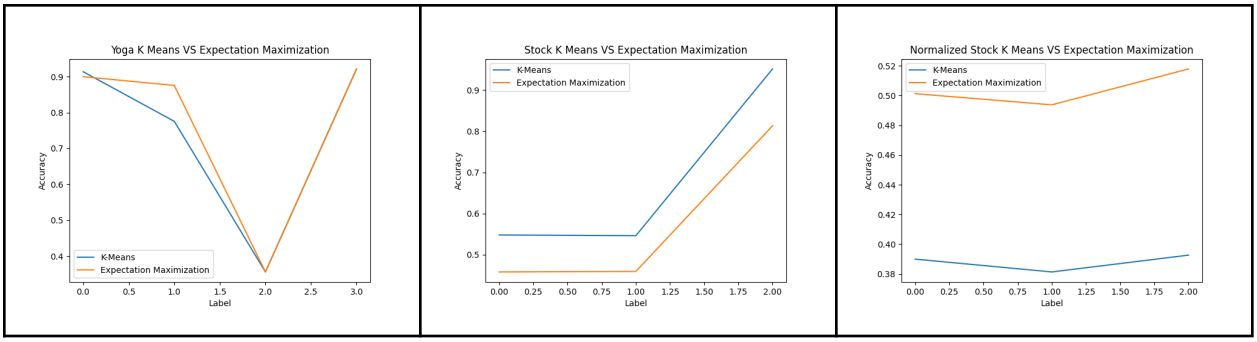
2 CLUSTERING

2.1 Methodology

After implementing EM and KM I ran it on the datasets. To compare the performance of the clustering, I decided to use accuracy, by counting the number of data points that fell into each label. The goal was not to find the best k for each clustering methodology, but rather to compare them against each other. I already know the best K for each data set because they are classification data sets.

2.2 Datasets & Analysis

I made a chart of the accuracy percentage per label for each of the data sets, along with a table with the raw data for these charts.



As you can see for the yoga data set, the clustering worked pretty well and EM seemed to have done a little bit better than KM especially for label one, which is the Warrior II pose. But both KM and EM had a very low accuracy rate for label 2, which is the Triangle pose. I believe it did not perform as well compared to the other poses because the number of rows for this pose is lower than the other two poses. Also because looking at the clustered data it seems to be very similar to the other two poses, and therefore bleeds into both of them causing a lower accuracy rate. For the stock data set, I was expecting under performance, but it seems like on the normalized data set expectation maximization has a better accuracy rate compared to KM, which is the opposite of the basic dataset where KM did better than EM. Both KM and EM are very accurate for one specific label. In other runs with different seeds, one label is always very high accuracy. I believe this is an anomalous feature of clustering with this dataset, because the cleaner normalized data set does not seem to have a bias for any specific label and performs equally as well for all of them. SKLearn did note that KM and GMM (EM) are pretty similar, so I'm not that surprised by the similarity of the results across the two algorithms. The datasets also don't have that much depth, but the difference for the normalized stock dataset has EM with a much higher accuracy. I believe this is because the nature of EM is to assign probabilities to data points instead of just assigning a cluster to each data point, which KM does, allowing to have a higher accuracy for the normalized stock data set. It's also a bit slower because of this more involved methodology.

Dataset	Clustering	Time	Accuracy	Results
Yoga	KM	0.13s	74.17%	{1: {1: 329, 3: 4, 0: 27}, 0: {0: 698, 1: 56, 2: 142, 3: 4}, 2: {2: 85, 3: 39, 1: 122, 0: 97}, 3: {1: 36, 2: 423}}
Yoga	EM	0.06s	76.32%	{1: {3: 324, 0: 4, 1: 32}, 0: {1: 788, 2: 93, 0: 5, 3: 14}, 2: {0: 110, 3: 122, 1: 111}, 3: {3: 36, 2: 423}}
Stock	KM	0.14s	68.16%	{2: {1: 3262, 2: 1016, 0: 1679}, 0: {1: 3229, 2: 943, 0: 1744}, 1: {1: 2021, 2: 44, 0: 59}}
Stock	EM	3.23s	57.65%	{2: {0: 1699, 2: 1532, 1: 2726}, 0: {0: 1720, 2: 1481, 1: 2715}, 1: {0: 1727, 2: 293, 1: 104}}
Normalized Stock	KM	0.64s	38.80%	{2: {1: 1726, 0: 2323, 2: 1908}, 0: {1: 1739, 0: 2256, 2: 1921}, 1: {1: 628, 0: 834, 2: 662}}
Normalized Stock	EM	0.96s	50.43%	{2: {1: 2986, 0: 1950, 2: 1021}, 0: {1: 2921, 0: 1928, 2: 1067}, 1: {1: 1100, 0: 652, 2: 372}}

To summarize, EM is generally more accurate due to it's probabilistic approach, and thus also slower in

general, the two exceptions to this being the basic stock dataset for accuracy and the yoga dataset for time.

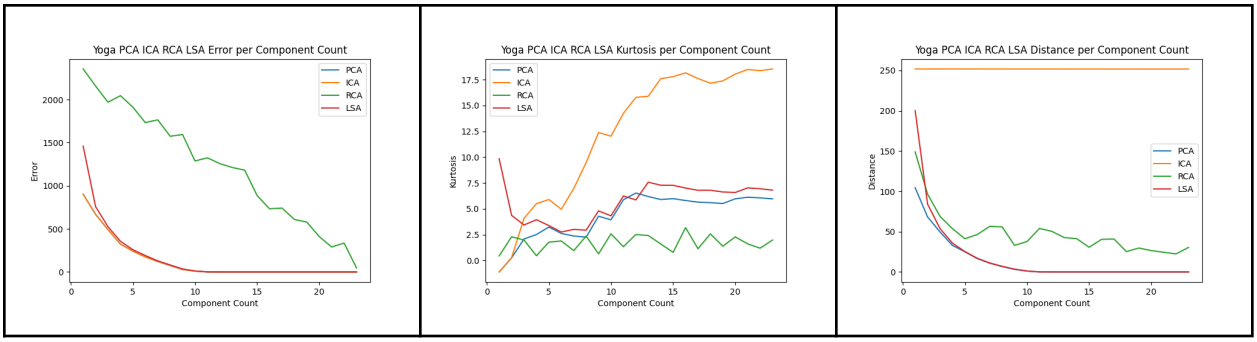
3 DIMENSIONALITY REDUCTION

3.1 Methodology

For dimensionality reduction, my general approach has been to implement every single approach, namely PCA, ICA, RCA & LSA, and then get a few specific key metrics for all of them across Component Counts to find the best Component Count per data set. I mainly compared Error - Sum of Squared Reconstruction Error, Distance - euclidean distance, and Kurtosis - non-gaussianity, to find the minimum number of components, for use in the NN dataset. I wanted to get to a point of diminishing returns for error, distance and kurtosis.

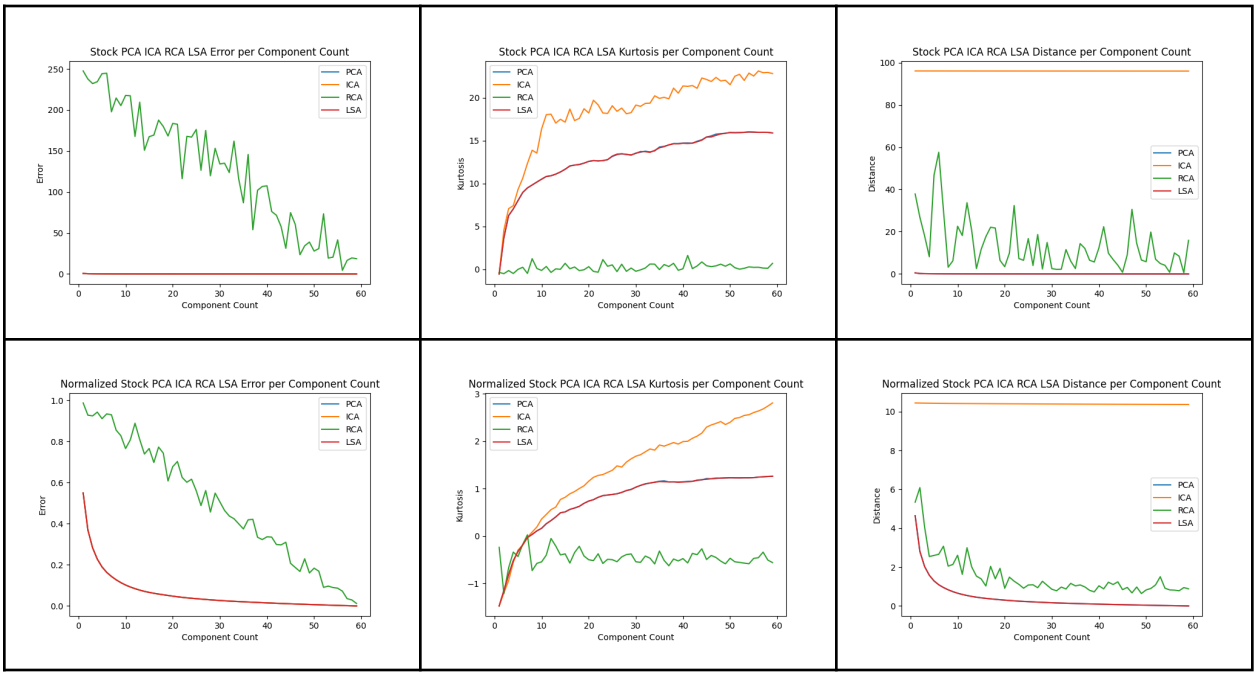
3.2 Yoga

For the Yoga dataset, I settled on a component count of 12 for use in the neural network; it seems to have reached the point of diminishing returns for error, distance, and kurtosis. Kurtosis could have gone a little higher, but not by much. The only exception to this being the error for RCA, which is fine since it is, by definition, random.



Curiously, PCA, ICA and LSA are very similar in general across error, distance, and perhaps kurtosis. In fact, I would go so far as to say that PCA and LSA are very similar, with kurtosis being the only curve that showed any visible differences. I believe that's because SKlearn LSA is similar to PCA, except for its ability to work with sparse matrices more efficiently compared to PCA. As expected, RCA needed more components to perform better, ICA optimized for kurtosis to find the independent features of the data, while PCA did the opposite.

3.3 Stock



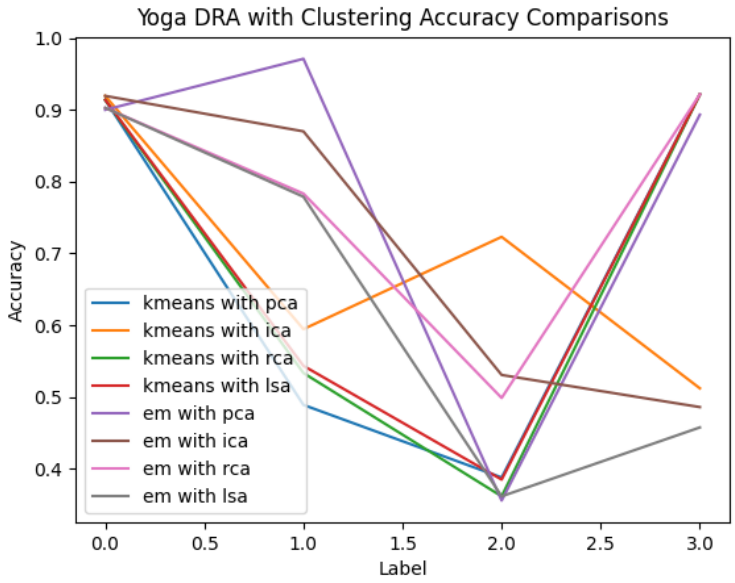
For the stock dataset, I decided to use a component count of 35 for the normalized data set and 45 for the basic data set. It is a pretty good improvement from 60 features. I was able to get a lower component count for the normalized data set because it is more easily reduced being normalized. There are a few other noticeable differences between the different algorithms performed across the normalized and the basic stock datasets. For example, there is a more clear curve of decreasing error in the normalized data set, whereas in the basic one it's effectively just a flat line. Adding components didn't actually reduce any error, Similarly for distance. Also, the rate of diminishing returns for the normalized stock data set was more quickly reached across all metrics, when compared to the basic data set, where more components were needed to bring it to a state of diminishing returns. Interestingly, the ICA kept increasing for the normalized data, but only upto 3, indicating the normalized data set of stock data is a true gaussian distribution. This is probably why it's not a good dataset for actual learning.

4. Clustering Dimensionally Reduced Data

4.1 Methodology

To get all 16 combinations of clustering and DR algorithms, I ran all of the data through two for loops for each of the methodologies of clustering and DR. First I applied the dimensional reduction on the data and then passed it on to be clustered. From there, I judged the algorithms by the accuracy rates for the dimensionally reduced clustered datasets.

4.2 Yoga



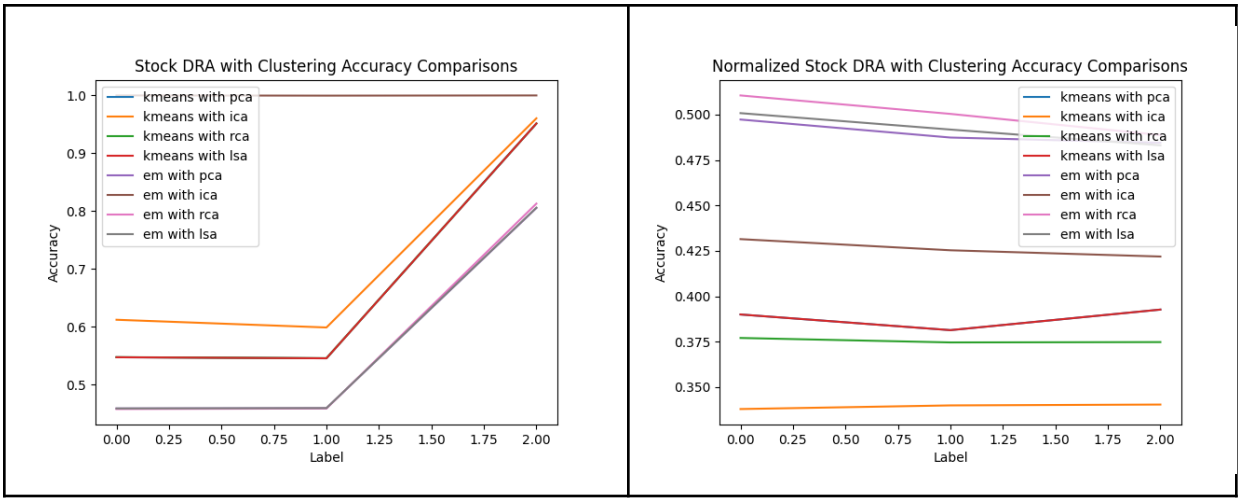
Surprisingly, there isn't much of a departure from the yoga data set clustered without any DR applied to it. However, there are a few specific features of interest to note - EM with PCA has really good accuracy for labels 1 and 3, above average in general when looking at the rest of the lines. Also, KM with ICA has the highest accuracy for label two where every combination struggled; conversely, it struggles with label 1 and 3 the most. I bring these two specific features to top of mind because they're effectively diametric opposites of each other when it comes to their intuitive meanings, and may be indicative of what PCA with ICA are doing here. Furthermore, the only two combinations that come close to KM with EM, and EM with RCA. I'm going to discard RCA, since it's effectively random, but to note here is that ICA seems to be very good at helping the clustering algorithm figure out label two, where it was sorely lacking when alone. After some initial thought, I believe this is happening because the clustering algorithms in general aren't able to pose label 2 well because it is very close to the other two poses, whereas pose label 1 and pose label 3 are very far from each other and therefore can be more easily identified in vanilla clustering. I believe ICA is doing a good job with KM to identify pose label 2 because it's finding the independent components of that pose to be clustered more identifiably. If you look at the graph, you'll notice that all the PCA combinations are absolutely terrible at figuring out what pose label 2 is, even worse than standard clustering. Inversely however, it seems like the algorithm with the highest accuracy in general is EM with PCA, as I called out above. Also one of the fastest iterations excluding EM with RCA - but again that's just random, but also fast.

Clustering	DRA	Time	Accuracy
kmeans	pca	.24s	67.80%
kmeans	ica	.21s	68.72%

kmeans	rca	.16s	68.26%
kmeans	lsa	.15s	69.09%
em	pca	.14s	78.00%
em	ica	.18s	70.15%
em	rca	.13s	77.66%
em	lsa	.17s	62.52%

4.3 Stock

Again, there isn't much of a departure from the vanilla clustered results for the stock dataset; it is the magnitude of accuracy that seems to be changing. Otherwise, the pattern seems to be the same with the massive exception - again - from EM with ICA. That seems to have 100% accuracy for the stock data set when it comes to clustering the signal label correctly. I am tempted to write it off as a fluke, but I want to run more tests as future work to see if EM with ICA is magically able to predict the stock market very accurately. This might again be linked to ICA's ability, with EM, to be able to flexibly classify and identify different labels that may have been too similar before by spreading them into their individual parts.



It seems like in general all of these took roughly the same amount of time around 6 to 8 seconds with varying accuracies from 30% to 70%. KM with ICA seems to be the slowest however, which isn't surprising due to ICA's methodology.

Dataset	Clustering	DRA	Time	Accuracy
Stock	kmeans	pca	6.89s	68.16%
Stock	kmeans	ica	12.43s	72.40%
Stock	kmeans	rca	6.08s	68.17%
Stock	kmeans	lsa	6.1s	68.16%
Stock	em	pca	8.4s	57.48%

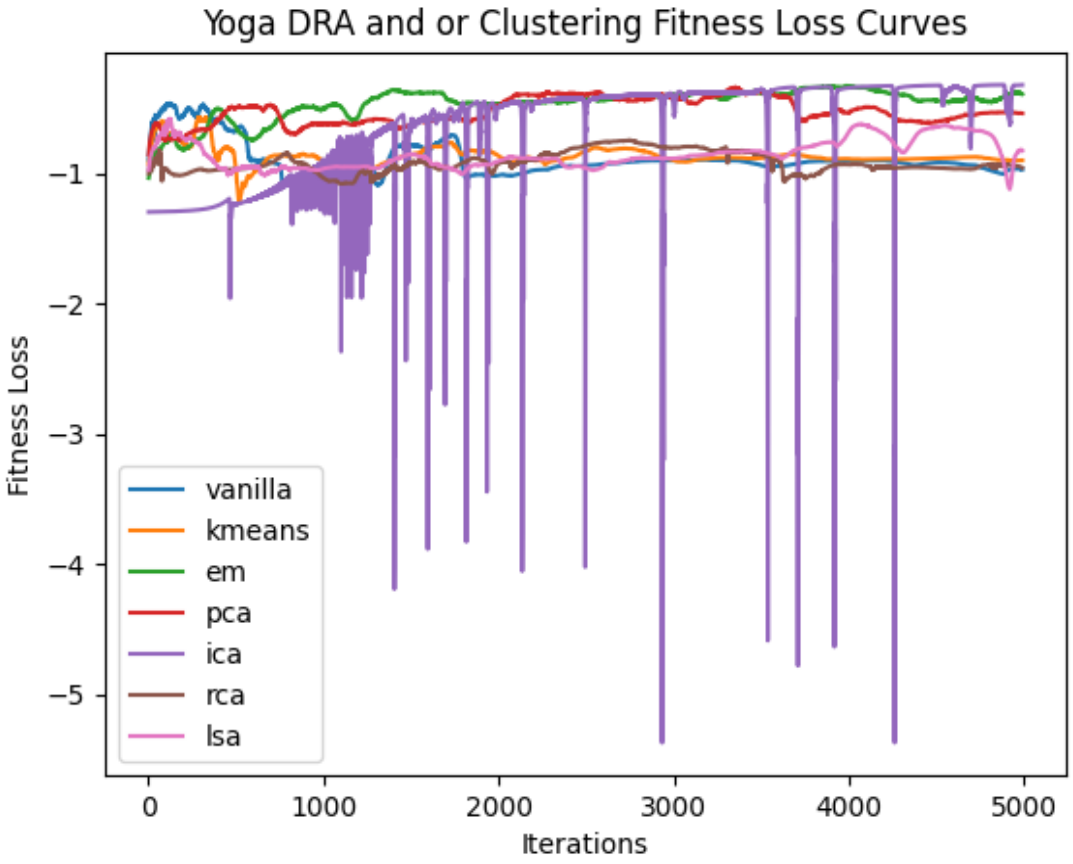
Stock	em	ica	6.59s	99.99%
Stock	em	rca	7.52s	57.68%
Stock	em	lsa	7.78s	57.50%
Normalized Stock	kmeans	pca	6.82s	38.80%
Normalized Stock	kmeans	ica	8.s	33.94%
Normalized Stock	kmeans	rca	6.81s	37.55%
Normalized Stock	kmeans	lsa	6.82s	38.80%
Normalized Stock	em	pca	6.97s	48.97%
Normalized Stock	em	ica	7.85s	42.62%
Normalized Stock	em	rca	6.57s	49.98%
Normalized Stock	em	lsa	6.94s	49.18%

5 NEURAL NETWORK

5.1 Methodology

For this specific part of the assignment, I pulled everything together and made three sequences of work. The first sequence is the vanilla Neural Network training. I basically copy pasted everything from Assignment 2, including the tuning for the yoga dataset, and ran it as the baseline of comparison. The second and third sequences are two for loops I ran for the Neural Network with an additional one-hot encoded cluster label added to the features, and one for the Neural Network with a DR applied dataset. I didn't combine them because that was not in the instructions, but I would be very interested in doing so for future work. I got some interesting data regardless. To compare the different methodologies, I plotted the fitness curve that MLRose provides from the iterations in training. For tuning, I use a learning rate of 0.001 and I decided to use 500 maximum attempts with 5,000 iterations to fully see how well each of the different methodologies did on the neural network. I'm unsure if more tuning could have done a better job for each of the algorithms, but in the spirit of a proper comparison, I decided to use the same tuning across all of them to see the best fit, and highest training and testing accuracies.

5.2 Yoga



Not surprisingly as led up to from all the previous analysis. ICA seems to have done the best when it comes to the fitness of the neural network and it also seems to have the highest training accuracy, but it has the lowest testing accuracy which suggests that it might have overfit on the data. The fastest algorithm seems to be LSA by 5 seconds, but in general they all seem to take the same amount of time, around 21-24 seconds. Weirdly enough, RCA took the longest. In terms of the algorithm that did the best however, it seems to be the EM clustered features one-hot encoded appended to the NN dataset. I believe EM did the best here because EM had the highest accuracy for clustering and so when one hot encoded and appended to the data set might have helped in your undoor enough work to better predict the pose. It also seems to have the lowest loss other than ICA which overfit.

Yoga	Time	Loss	Training Accuracy	Testing Accuracy
vanilla	24.29s	0.4559465792	83.70%	83.53%
kmeans	24.96s	0.5634350767	80.72%	80.43%
em	24.19s	0.3212204153	88.23%	86.43%
pca	23.17s	0.3352392047	89.46%	68.02%

ica	21.99s	0.3143535211	91.91%	32.95%
rca	27.37s	0.7406064802	67.53%	33.72%
lsa	18.64s	0.5762734558	81.24%	65.70%