Pranshav Thakkar

GT username: pthakkar7

# Unsupervised Learning

In this assignment we were tasked with applying two clustering algorithms, namely K-means and Expectation Maximization (I used a Gaussian Mixture for this) on two datasets. We also had to apply 4 dimensionality reduction algorithms on the datasets, and they were PCA, ICA, Randomized Projection and K-best Selection. In addition, we observed how the clustering algorithms performed after running dimensionality reduction. We also observed how our neural network from assignment #1 performed after dimensionality reduction and after clustering had been applied to the datasets after dimensionality reduction. Similar to assignment #1, I used the scikit-learn library in a python platform to run my code.

I used the same two datasets that I used in assignment #1 for this assignment. The first database is based on the game of chess where the player using white has a king and a rook left, while the player using black has a king and a pawn left. The dataset has 36 attributes that describe the possible states of the board and is classified with a ‘win’ or ‘nowin’ based on whether white is able to win the game based on the configuration of the board. This dataset has slightly over 3000 features. The second database is based on the game of tic-tac-toe where the player using ‘x’ is determined to have started first in the game. The dataset has 9 attributes that represent various states of the game and is classified with a ‘positive’ if ‘x’ is able to win the game or with a ‘negative’ if ‘x’ loses the game or the game ends in a draw. This dataset has slightly under 1000 attributes.

For the clustering algorithms, I decided on a cluster size of k = 2, because both of my datasets are binarily classified, with basically a yes or no outcome. This makes it very clear that the clustering algorithms should attempt to populate the instances in two distinct clusters representing the two different classifications.

For the dimensionality reduction algorithms, I decided to select nearly half of the features for both of my datasets, because I was curious to see how that would affect the results. I selected 20 out of 36 features for the chess database, and 4 out of 9 for the tic-tac-toe database. I decided to go with slightly more than half for the database with a larger number of instances and slightly less than half with the dataset with a lower number of instances, just to accentuate how it affects it.

## Analysis

### Clustering

For clustering, K-means is a simple algorithm that is similar to K-nearest-neighbors that we saw while we were doing Supervised Learning. Expectation Maximization is supposed to be a much more powerful algorithm than K-means, and I used a randomly initiated Gaussian Mixture to perform this. To measure the results of these algorithms, I used a few metrics from Scikit-learn: Adjusted Rand Index, Adjusted Mutual Info, Homogeneity Score, Completeness Score and the Silhouette Score.

| | KMeans | EM |
|---|---|---|
| Chess ARI | 0.005818104169322276 | 0.019417274153462373 |
| Chess AMI | 0.006102634894061353 | 0.02245149910382523 |
| Chess Homogeneity | 0.006327788994009074 | 0.022672770627381624 |
| Chess Completeness | 0.017252604912457715 | 0.04533651104513444 |
| Chess Silhouette | 0.39504165554060156 | 0.09219851042918989 |
| TTT ARI | 0.0031615229329602667 | 0.0015858250593446626 |
| TTT AMI | 0.00769620720279335 | 0.0037509073512742848 |
| TTT Homogeneity | 0.00901107489849434 | 0.004823352922629623 |
| TTT Completeness | 0.008450209104664791 | 0.004504833248630563 |
| TTT Silhouette | 0.10778060653100038 | 0.03411523061031808 |

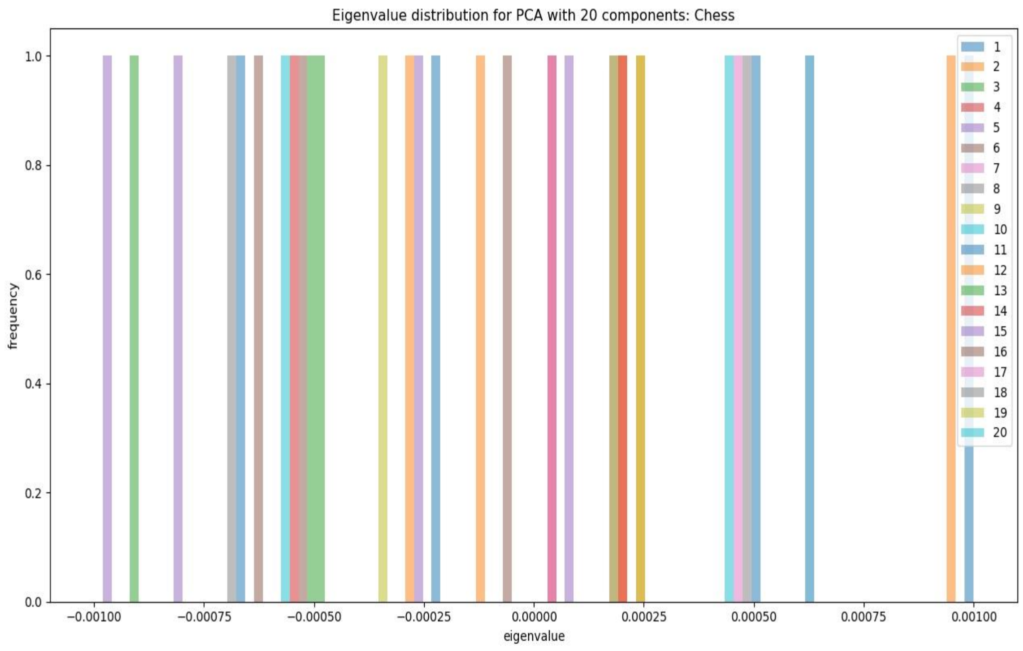
The perfect score for all of these metrics is a 1.0, which would mean that the clusters matched perfectly with the expected labels based on how they were classified. Clearly, the scores are very, very poor with the exception of Kmeans' silhouette score on the chess dataset, which is still pretty bad but is miles ahead in comparison. I believe this is a result of the datasets that I have selected. The way that they are set up (with attributes encoded to integers such as 0, 1, 2, 3, etc.) may make it difficult for the clustering algorithms to learn what the numbers mean and how they are significant. Hence, the scores are bad in terms of instances being correctly clustered.

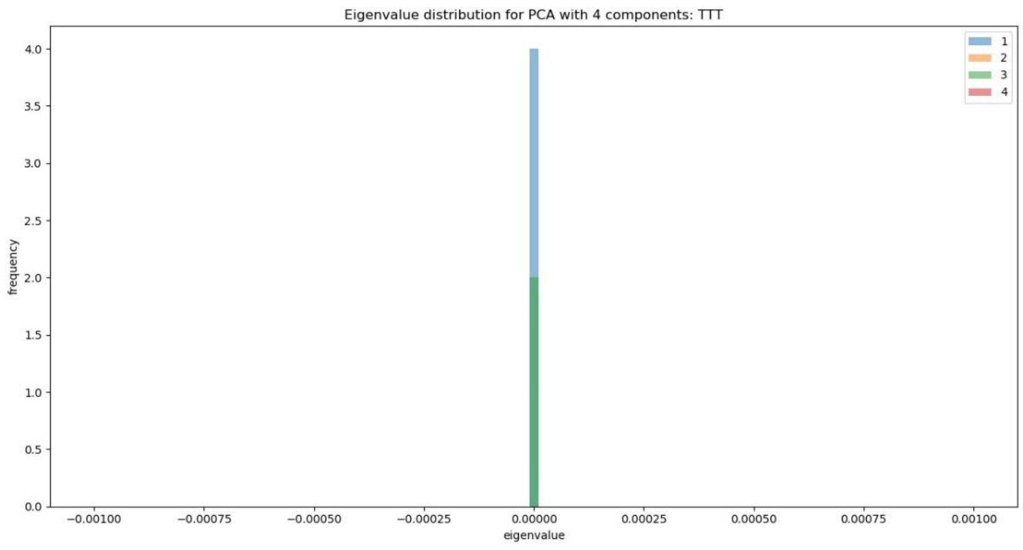
The Silhouette score is different from the other metrics as it compares the instances and the correct labels based on a formula, while the other metrics compare the correct labels and the predicted labels. The Silhouette score was generally the highest metric except in one case.

Overall, EM performed significantly better than K-means for my chess dataset except in the case of the silhouette. However, EM performed about half as well as K-means for the tic-tac-toe dataset, in all metrics. Again, I believe this to be a case of the problems being different, and the fact the EM had less material to work with for the TTT dataset.

## Dimensionality Reduction

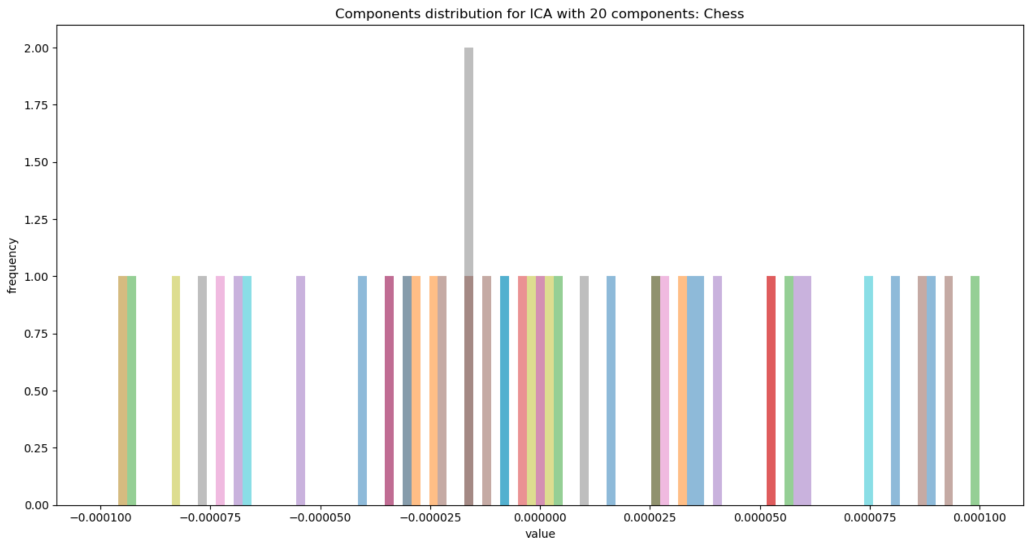
### PCA





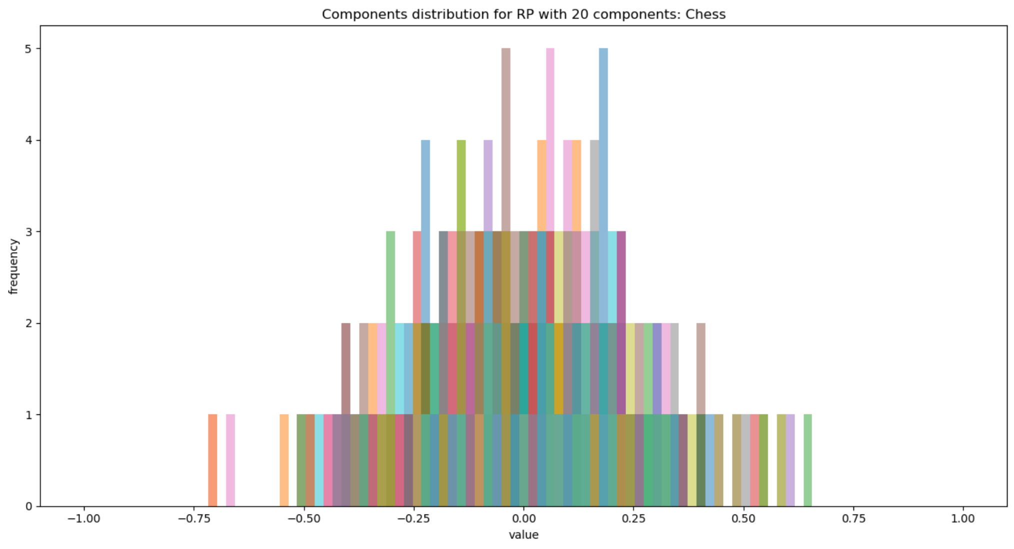
PCA seems to have distributed the data all over the place for the chess dataset and it seems like the data is just in one place for the TTT dataset. It may be that the TTT dataset is small enough already without having feature reduction done on it, and the doing so makes the dataset not very variant.

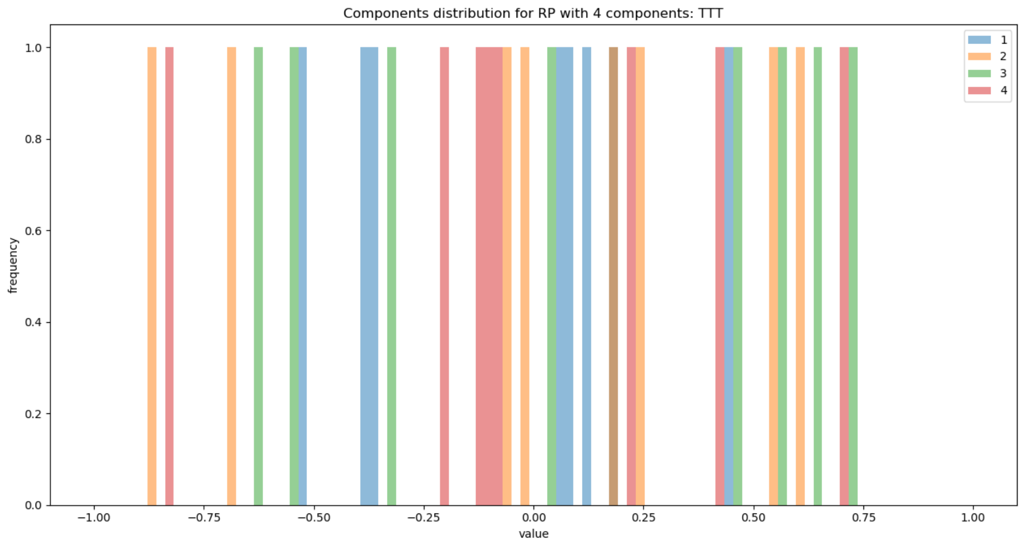
## ICA



Similar to PCA, the chess database is represented by components that are all over the place in ICA. This is supported by the fact that the kurtosis for this was 18.436. This leads me to believe that the algorithms are having a hard time with the specific problems and reducing the features, which makes sense since both of my algorithms have features that represent the state of their boards! Any loss in features would greatly take away from the proper information needed to understand the problems, and this is represented in how ICA and PCA show the distributions of their components, as they now no longer have much correlation.

## RP





RP seems to look much better than PCA and ICA for both datasets, especially with the chess dataset, where we start to see a Gaussian-type shape for the distribution. This also makes sense since these are random projections, and the order of the features does not matter. Obviously, with the specific problems that I have chosen, it does matter and that is why the results of PCA and ICA look the way they do, but here that fact is not highlighted as much. RP makes feature reduction on these datasets less of an issue than it really is.

**Clustering Pt.2**

Now, let's take a look at the results of running the clustering algorithms and their metrics on the datasets after they have gone through dimensionality reduction. Given the nature of the problems that I have chosen, and the fact that feature reduction most likely negatively impacts the dataset, it is no surprise that the clustering algorithms performed worse in some cases. EM took a beating in all cases, performing worse than it did before feature selection, and the ARI and AMI metrics even went into the negatives for the TTT dataset. The KMeans algorithm gave the same result for the Chess dataset after PCA, but gave a worse performance in all other cases.

PCA

| | KMeans | EM |
|---|---|---|
| Chess ARI | 0.005818104169322276 | 0.00022254385501963722 |
| Chess AMI | 0.006102634894061353 | 0.00012218269531153597 |
| Chess Homogeneity | 0.006327788994009074 | 0.00034839415749439467 |
| Chess Completeness | 0.017252604912457715 | 0.0003480882719469025 |
| Chess Silhouette | 0.4107794634305798 | 0.12608562264006737 |
| TTT ARI | 0.003991979856043702 | -0.0005068092699970158 |
| TTT AMI | 0.004529119087223443 | -0.0006790375017824757 |
| TTT Homogeneity | 0.005667709751813028 | 8.309306213473662e-05 |
| TTT Completeness | 0.005280627876054234 | 7.753361313471835e-05 |
| TTT Silhouette | 0.182355731139925 | 0.12066493821436777 |

ICA

| | KMeans | EM |
|---|---|---|
| Chess ARI | 0.004007256968699559 | 0.00022254385501963722 |
| Chess AMI | 0.002201404381352012 | 0.00012218269531153597 |
| Chess Homogeneity | 0.0024270455509765327 | 0.00034839415749439467 |
| Chess Completeness | 0.0025138312633982956 | 0.0003480882719469025 |
| Chess Silhouette | 0.05078990368665852 | 0.05407729780233709 |
| TTT ARI | 0.0031278125348101647 | -0.0005068092699970158 |
| TTT AMI | 0.003822829063393355 | -0.0006790375017824757 |
| TTT Homogeneity | 0.004909793442807826 | 8.309306213473662e-05 |
| TTT Completeness | 0.004574949436239911 | 7.753361313471835e-05 |
| TTT Silhouette | 0.18282784007113448 | 0.12164772416470268 |

RP

| | KMeans | EM |
|---|---|---|
| Chess ARI | 0.049010970189927325 | 0.04653643650428293 |
| Chess AMI | 0.03434829342706593 | 0.03253043349869477 |
| Chess Homogeneity | 0.03456666171770268 | 0.03274921315017751 |
| Chess Completeness | 0.03522475879307494 | 0.03342007287146747 |
| Chess Silhouette | 0.1366840318705211 | 0.03709477556221993 |
| TTT ARI | 0.013625489322184077 | 0.007050681133365674 |
| TTT AMI | 0.009068426514727057 | 0.004541231967742342 |
| TTT Homogeneity | 0.010536080556454077 | 0.0056820346502036755 |
| TTT Completeness | 0.009816508453518955 | 0.005292525203698146 |
| TTT Silhouette | 0.27267473035270534 | 0.08034606576909696 |

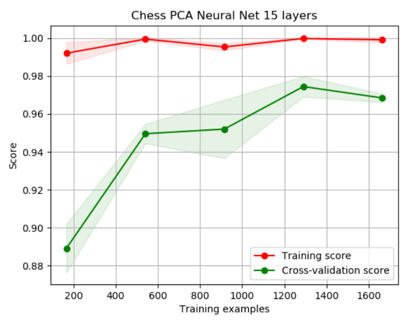
The results for Random Projections do not look nearly as bad as the results for PCA and ICA; in fact, they are better in some cases than the original clustering results. I understand these results as RP doesn't take into account how the features matter for these specific problems, and so the reduction of features may actually have made it easier for the algorithm to sort the features into the correct clusters.

**Neural Net**

I chose to run my 15-layer neural net on my chess dataset to see how it performed after feature reduction.

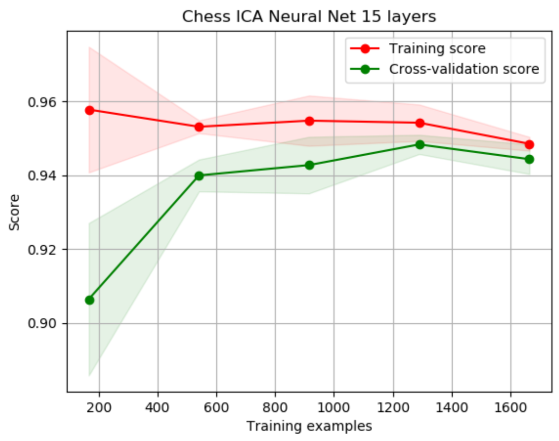
As a reminder, the neural net ran with a 0.9698275862068966 testing accuracy before feature selection.

PCA



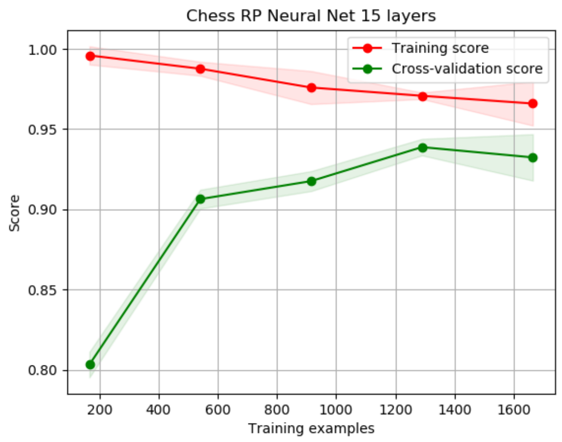
Testing accuracy: 0.5373563218390804

ICA



Testing accuracy: 0.5531609195402298

RP



Testing accuracy: 0.49137931034482757

Clearly, the testing accuracy for the neural net is MUCH worse than it was before feature selection, and the algorithm doesn't make a difference. We see from the training and cross-validation accuracies (which are very high) that they indicate a clear sign of overfitting. However, this result MAKES SENSE. This is due to the fact that each individual feature is very important to the dataset, given that they represent a position on the chess board and there is simply no easy way to get rid of that information or condense it. Thus, it becomes extremely hard for the Neural Net, or any algorithm for that matter, to predict the correct classification because we are simply missing crucial info. This results in a testing accuracy that can basically be compared to a flip of the coin.

Above all, what I have learned from this assignment is the ever-increasing importance of Domain Knowledge. We can't just randomly apply algorithms and expect them to give us good results; we have to know what we are trying to do with the algorithms and what we are applying them to. In this case, my datasets/problems are very specific problems that have to do with specific states. This makes it difficult to obtain good results from unsupervised learning algorithms like clustering, and it definitely made it clear to me that feature selection is not at all a good option, since each and every feature is equally important and vital to the algorithm's ability to learn from the dataset. Domain Knowledge should never be taken for granted. #NoFreeLunch