KU Leuven

Data Mining and Neural Networks

Assignment 3

Zachary Jones
Master of Statistics

January 6, 2020

Contents

1 Redundancy and Random Data

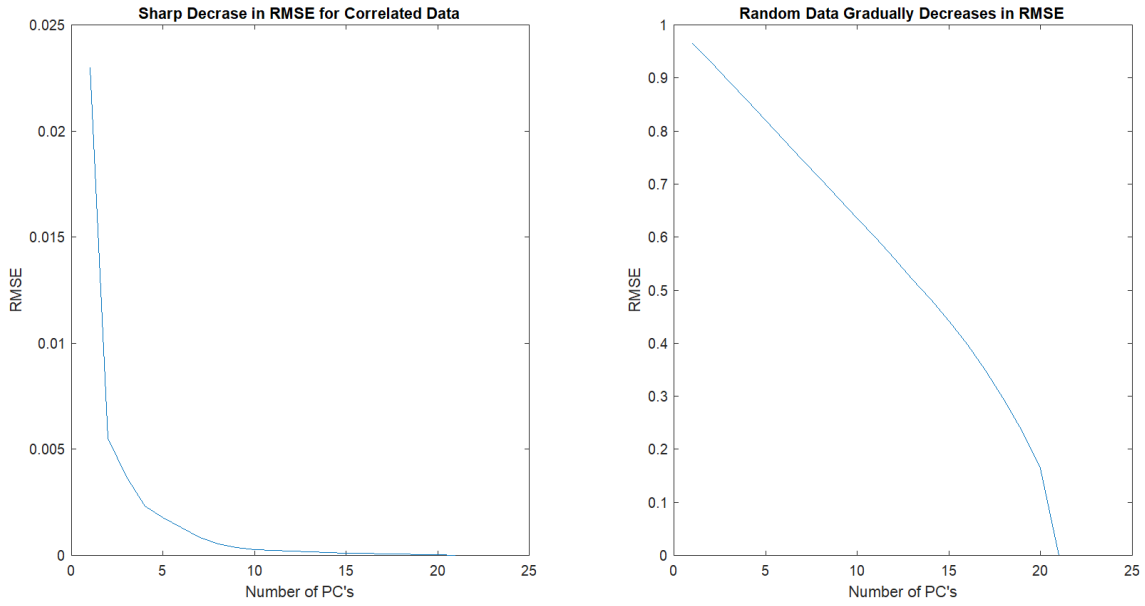


Figure 1: Comparison of Eigenvalues for Choles dataset and Random Generated data of same dimensionality

1.1

Using a manual method we are able to reconstruct the PC's and across all iterations of random vectors a similar plot to that of the right in figure 1 appears. A gradual decrease in reconstruction error as the number of eigenvalues included reaches the full dimensionality of the dataset, where it sharply falls. This is because the data is highly independent and the proportion variance explained by the eigenvalue decomposition of the covariance matrix is nearly constant. When the number of principal components is equal to the dimension of the covariance matrix the reconstruction is perfect and the error is zero.

1.2

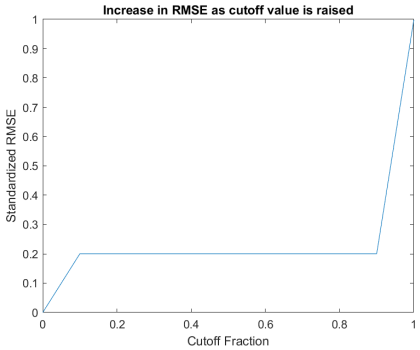


Figure 2: The Choles Dataset Recomposition Using MaxFrac and processpca()

For the eigenvalue decomposition and reconstruction of the Choles dataset with RMSE illustrated in figure 1 on the right, we see a quite different trend. The data is correlated and so the principal components do not all explain the same amount of variance. This is because the process of performing PCA corresponds to projecting the data onto a subspace with the highest possible variance, larger eigenvalues (the lower ones in this example) correspond to higher proportions of the variance explained by the reconstruction of the covariance matrix, and thus a better reconstruction of the original matrix as well.

1.3

We can see in figure 2 that we get similar results with processpca() and maxfrac, the plot plateus as no new eigenvector is larger than the cutoff.

2 PCA on Handwritten Digits

2.1

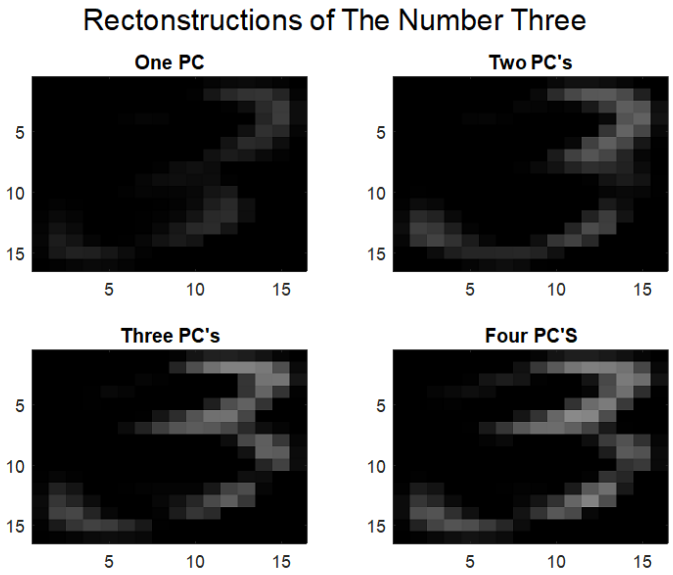
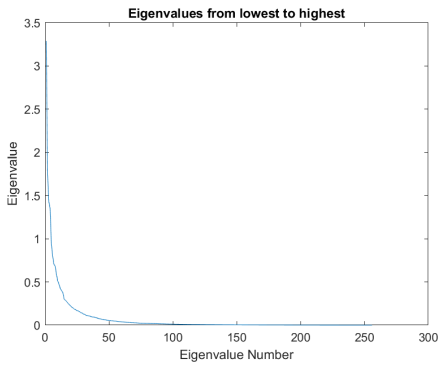


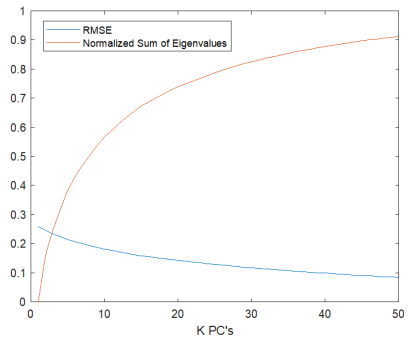
Figure 3: Reconstruction of digit '3' with one, two, three, and four PC's gives higher resolution

The cell containg the digit '3' has an average value of 0.3026, which being quite small corresponds to a nearly dark image. As expected given the previous section, as the number of principal components used in the decomposition and reconstruction increases so does the amount of information retained, which corresponds to a higher resolution image. As can be seen in figure 4(a), the value of the eigenvalues sharply decreases at an early stage, indicating that a large portion of the variance is actually explained with lower dimensional data, eventually the value drops to nearly zero. If all 256 dimensions are used, there is no information loss in the reconstruction and the error should be zero. In practice there are small computational errors that can keep this from being the case.

To further illustrate this point, in figure 4(B) shows the clear relationship between the sum of all eigenvalues up to a given index and the RMSE. Note the sharp increase in the sum of the eigenvalues as it corresponds to a strong decline in the RMSE. At any given point we can choose not to include further eigenvalues, and the RMSE will correspond to the remaining sum which will be small. Best practice for choosing the number of eigenvalues can either be to look for an 'elbow' in the graph or to use Horn's Parallel procedure.



(a) Eigenvalues sharply decrease in value



(b) The RMSE decreases as the sum of the eigenvalues increases

Figure 4

3 Density-Based Methods

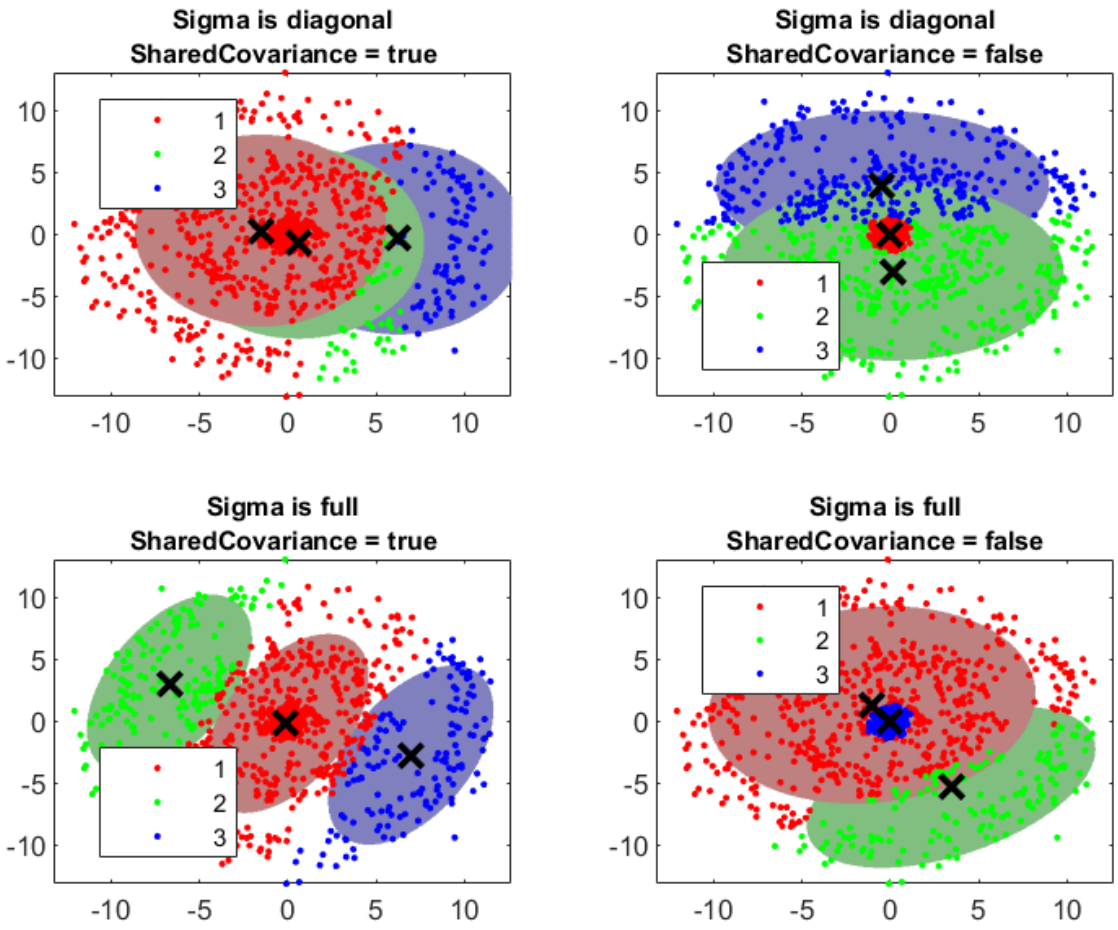


Figure 5

3.1

In the Gaussian Mixture Model, data is assumed to follow a normal distribution with either equal, or unequal covariance matrices, and points are assigned to a given cluster through a maximum liklihood method found with the EM algorithm that minimizes the negative log liklihood of the cluster given the datapoint. Chosing the most likely cluster given this method also limits clusters to an ellipsoid measure as the most probable values of a gaussian distribution will be centered around the mean value.

3.2

As is often with clustering, a visual explanation is often preferred: The clusters here simply do not seperate the rings due to the fact that the algorithm is limited to forming eliptical clusters. One encouraging result, however, is in the GMM solution with full and unequal covariance matrices. Because the variances are different, it is able to form clusters inside of clusters, potentially yielding a more appealing solution with some finetuning.

4 DBSCAN for Clustering

4.1

DBSCAN is a fairly straightforward algorithm that clusters points using two hyperparameters ϵ specifying the radius of a neighborhood with respect to some point, and *minPts* which specifies the minimum number of neighbors required for a given point to be considered a core point. DBSCAN iterates through each point seeing first if it is a core point, and if so adding all points reachable from it (within distance ϵ) to a single cluster. Clusters are formed by collections of core points and all points reachable from those core points, if the point is not a core point and not reachable from any other core point, it is considered an outlier and not included.

4.2

The first step in performing and examining the DBSCAN algorithm is managing the two hyperparameters, ϵ and *minPts*, which govern our solutions. One approach might be to find the minimum distances between points within each individual cluster and the minimum distance between separate clusters. If the neighborhood is smaller than the first minimum, then no point can be a core point and so there is a lower bound. However, this approach becomes a problem when trying to differentiate between clusters, as they can also be quite close to one another (especially if outliers have not been removed). One way to mitigate that problem would be to "clean" the dataset using a boxplot approach to detect outlying datapoints and remove them. While this approach can potentially yield reasonable results, it is best practice to avoid removing datapoints if at all possible. Hence the method depicted in figure 6 The idea behind this approach is to loop through the data, and calculate the sum of the three closest distances to any given point, three being the number of clusters we intend to find. Then, we order these distances from least to highest and search for an inflection point. As seen in figure 6 The distances between datapoints gradually increases until the distances between important data and noise starts to become apparent at the farther end of the lest at which point the distances rapidly become further. It is precisely at this point that we want to cutoff our definition of a local neighborhood.

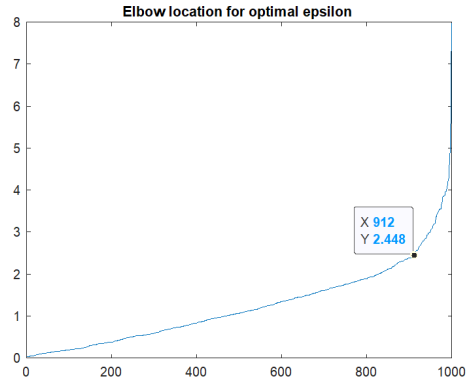


Figure 6: elbow indicating best value for epsilon

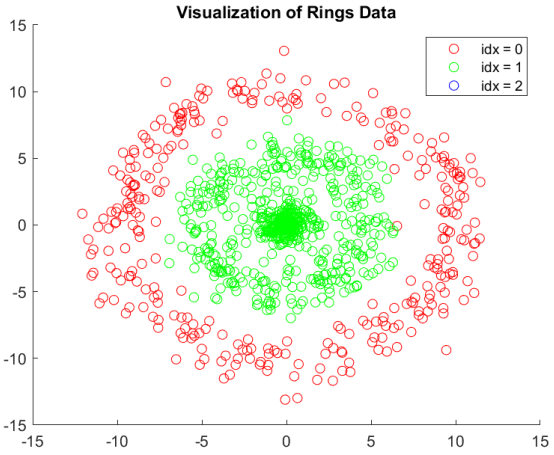


Figure 7: We can see here a clustering assignment of disparate rings using dbscan.

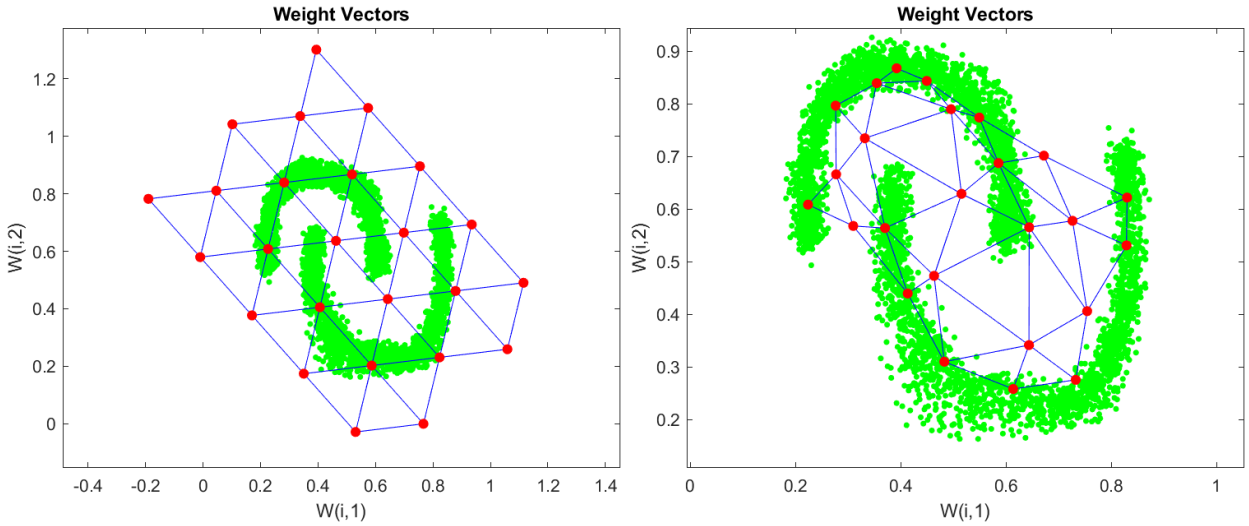
Then, focusing on the other hyperparameter *minPts*, we want to insure that the algorithm "sees" different densities of points at different clusters in order to distinguish between the two, as areas with different densities should be classified as different clusters. We can construct a heuristic measurement of density using the number of points and the area of the cluster. $\rho = \frac{n_i}{A_i}$. However, to make life easier, we simply divide by the "spread" of the data and approximate the density using the following.

$$\rho = \frac{n_i}{\sigma_x * \sigma_y} \propto \frac{minPts}{\epsilon^2}$$

Solving for minPts we get that class 0 has a characteristic value of 1504, class 1 of 137 and class 2 of 40. Plugging in these values it is clear that it is far easier to distinguish class 0 by setting the value of minPts high above 137.

5 Self Organizing Maps

5.1



(a) Initialization of a hextop grid to form a SOM

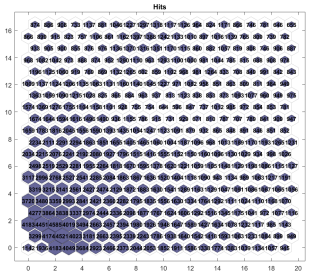
(b) SOM weights and neuron placement after training

Figure 8: The process of training on the Banana.mat dataset

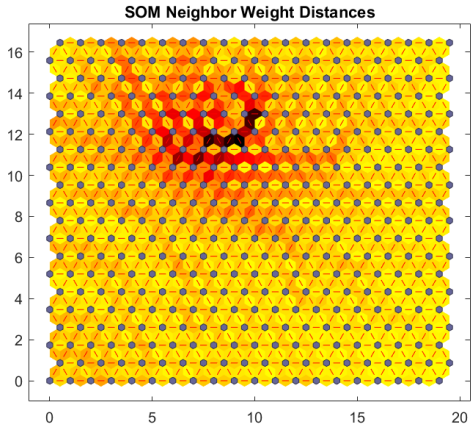
We examine first the network structure of a self organizing map trained on the Banana.mat toy dataset. Upon initialization, we use a hextop network topology, although both euclidean and random topologies yield similar results. Euclidean distances are used in this iteration, as this is a real shape and the variance carries important information about the distribution. In figure 8(a) the first thing to notice is that the hexagonal grid is placed over the entire distribution with the same distances between all weights. If the grid was too small or the points placed too close together the training process might "collapse" onto a local value. In figure 8(b) it can be seen that the traing procedure straightforwardly moves neurons towards the distribution. Interestingly, the interior surface of the distribution is what is mapped, implying that the interior neurons "pulled in" the outer neurons. This is becuase the closest neurons to the distribution become the "Best Matching Units" (BMU) in the vector quantization algorithm and pull neurons close to them as they are all updated together.

5.2

Next we turn our attention to the less trivial covertype dataset, in which we attempt to predict the type of forest cover using 54 seperate variables. Immediately a problem is apparent, the dataset not only high dimensional, but there are 581,012 different samples to analyze. Overcoming the first difficulty is one of the primary reasons behind creating a SOM, to make lower dimensional representations of difficult datasets. The second difficulty, however, is more challenging. As mentioned in the previous exercise, if the neural net doesn't "cover" the space by equally spacing the weights it can be attracted to a local minima. Not only that, but if a network is too sparse it will not effectively characterize the data. Imagine analyzing the Banana.mat dataset with an SOM containing only 2 neurons, in the best of all possible worlds they would be isolated at the vertices of the two crescents, but would do a poor job of classifying new data or representing the data as a whole. TO avoid this, a rule of thumb is applied, $N = 5\sqrt{M}$ where N is the number of neurons and M is the number of datapoints. Applying this rule to the forset cover dataset, we need $N \approx 5\sqrt{(581012)} = 3811$ neurons, or a 62×62 grid to properly create an SOM.



(a) Hitmap Showing the network has been "pulled" to one side



(b) Weight Distances, darker weights are less strong.

Figure 9: Caption

This, however, is unfeasable on many computers (mine) and compromises need to be made. One possibility is to train the SOM on a random selection of a proportion of the data before generating clusters based on the distances between the neurons. One can then classify the rest of the dataset by first applying the trained SOM to the new datapoints and then comparing the output to the previously computed cluster centroids, assigning each new output to the cluster that is closest in euclidean distance.
That procedure, however, was not done here. Istead a 20×20 grid was trained on the forest cover dataset. As can be seen in figure 9(a), the hitmap and many of the neurons are concentrated in the lower left hand side of the SOM map, indicating a large collection of neurons in the same area. This observation is confirmed in figure 9(b), where we can see the weights in the lower left are extremely strong, implying that the neurons are close together. The performance of the clustering algorithm can be measured using the Adjust Rand Index (ARI), which indicates how much better the clustering procedure is than random chance. Values near zero imply that the clustering is not much better than random chance, whereas values near one are much better. While there is no canonical cutoff or lower threshold for accepting a clustering algorithm, it be compared across different models, where models with higher ARI are preferred.