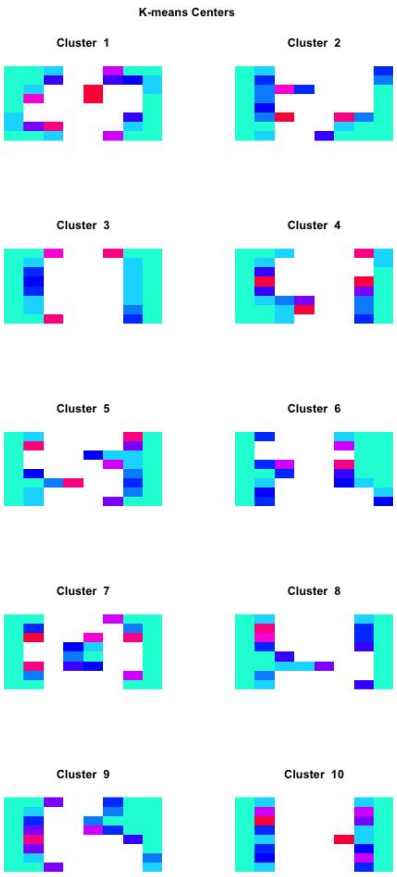
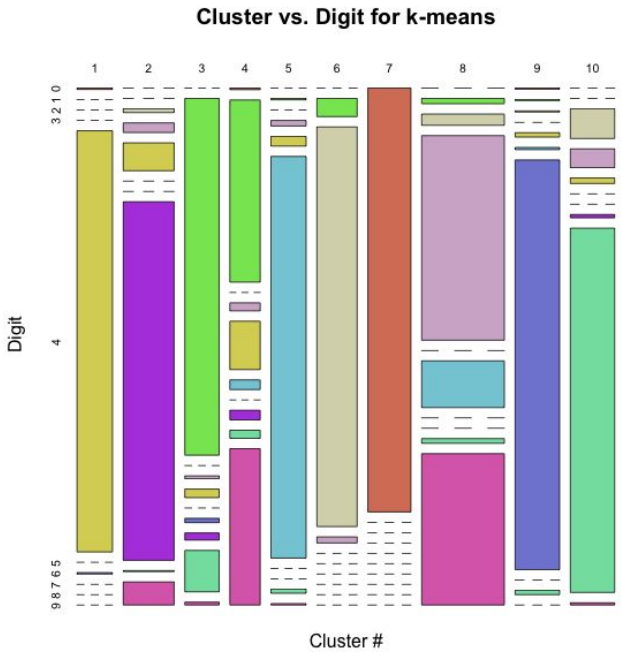
CS 7641 Assignment 3: Unsupervised Learning

Clustering

Handwriting recognition is a classic classification problem. I retrieved my handwriting dataset from here. In this particular dataset, the classification space was limited to digits 0-9 and the features were already preprocessed. There are 64 features which correspond to the 8x8 pixels of an image of a handwritten digit. Each pixel has a feature value of the intensity of the coloring in the range [0, 16]. This dataset came with a pre-split test set which I will use for my neural network experiments later. I started by running k-means on the 64 pixel features with k=10. Since each pixel already had a predefined range ([0, 16]), I did not scale the data and used euclidean distance as my distance function. For this and all future runs of kmeans, I ran 200 random restarts. To the left is a visualization of the 10 centers generated by the algorithm.

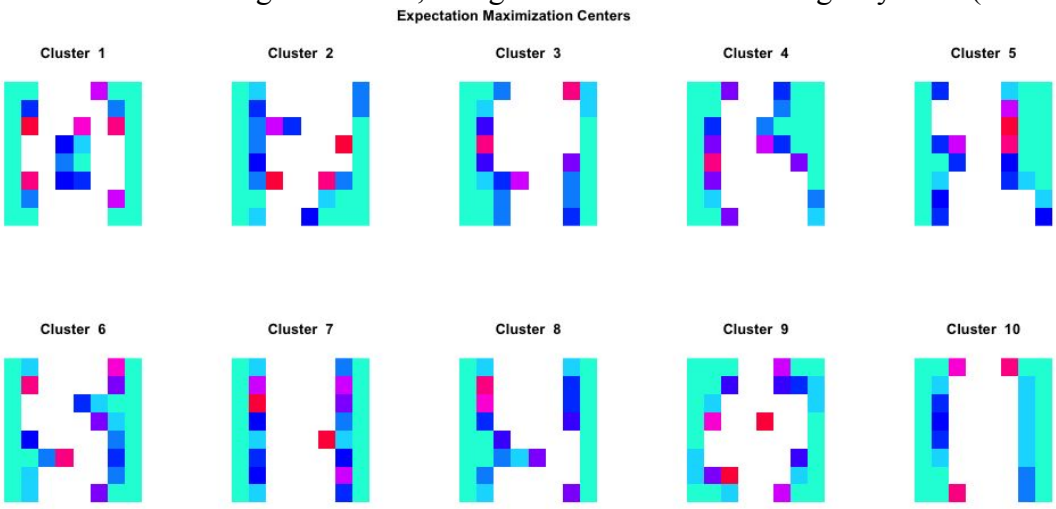


It is easy to visualize the centers as the digits themselves. Cluster 2's center looks like a 7, cluster 5's like a 5, cluster 7's like a 0, and cluster 9's like a 6. A visualization of the contingency table for cluster assignment vs. digit to the right shows the distribution of the different classes in the different clusters. It verifies the visual inspection of the centers since cluster 2 is dominated by 7, cluster 5 by 5, cluster 7 by 0, and cluster 9 by 6.

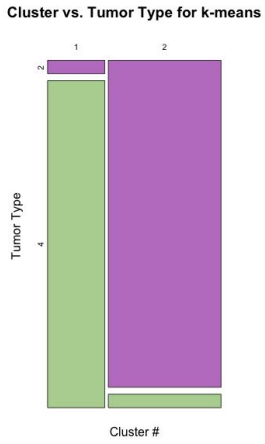


So all and all, k-means provided some interpretable clusters, but did not perfectly compare to the actual classes. It could be possible to run k-means with more clusters and potentially pick up different "styles" of say 3s, 9s, etc., however, I will refrain because I will later show that k-means with a combination of a dimension reduction technique greatly improves the mapping of a clusters to classes. I will quantify this by comparing the different clustering approaches using the adjusted rand index.

Next I ran expected maximization (EM) on the pixel features with k=10. I used the mclust package in R. Their algorithm for EM differs from that discussed in lecture in that instead of taking 10 points at random as the initialized centers, it uses hierarchical clustering for the initialization process and does no random restarts. The generated centers are displayed below, and appear very similar to the centers generated by kmeans in that we can actually visualize some of the centers as digits. As well, EM generates a similar contingency table (not visualized).



For my second dataset, I decided to analyze a classic cancer diagnosis dataset that uses a number of tumor attributes to predict whether a tumor is benign or malignant. I retrieved the dataset here. There were originally 699 observations, but 16 had missing attributes. I removed these 16 observations to create a total of 683 observations. I started by running k-means on the 9 tumor attributes with k=2. Since each tumor attribute is on a [1, 10] scale, I did not scale the data and used euclidean distance as my distance function. The table at the bottom of this page shows the centers generated by kmeans. The contrast between the two centers is: large measurements versus small measurements; every attribute in cluster 1's center is larger than its counterpart in cluster 2's center. To the left is the contingency table for cluster assignment vs. tumor type, which shows good class separation based solely on the clustering output. So all and all, k-means provided some interpretable clusters and does a very good job of creating clusters that partition the underlying tumor classes (i.e. malignant tumors are more likely to have larger tumor attribute measurements).



Cluster	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses
1	7.173913	6.8	6.734783	5.739130	5.478261	7.930435	6.108696	6.039130	2.569565
2	3.055188	1.298013	1.428256	1.353201	2.094923	1.317881	2.092715	1.260486	1.112583

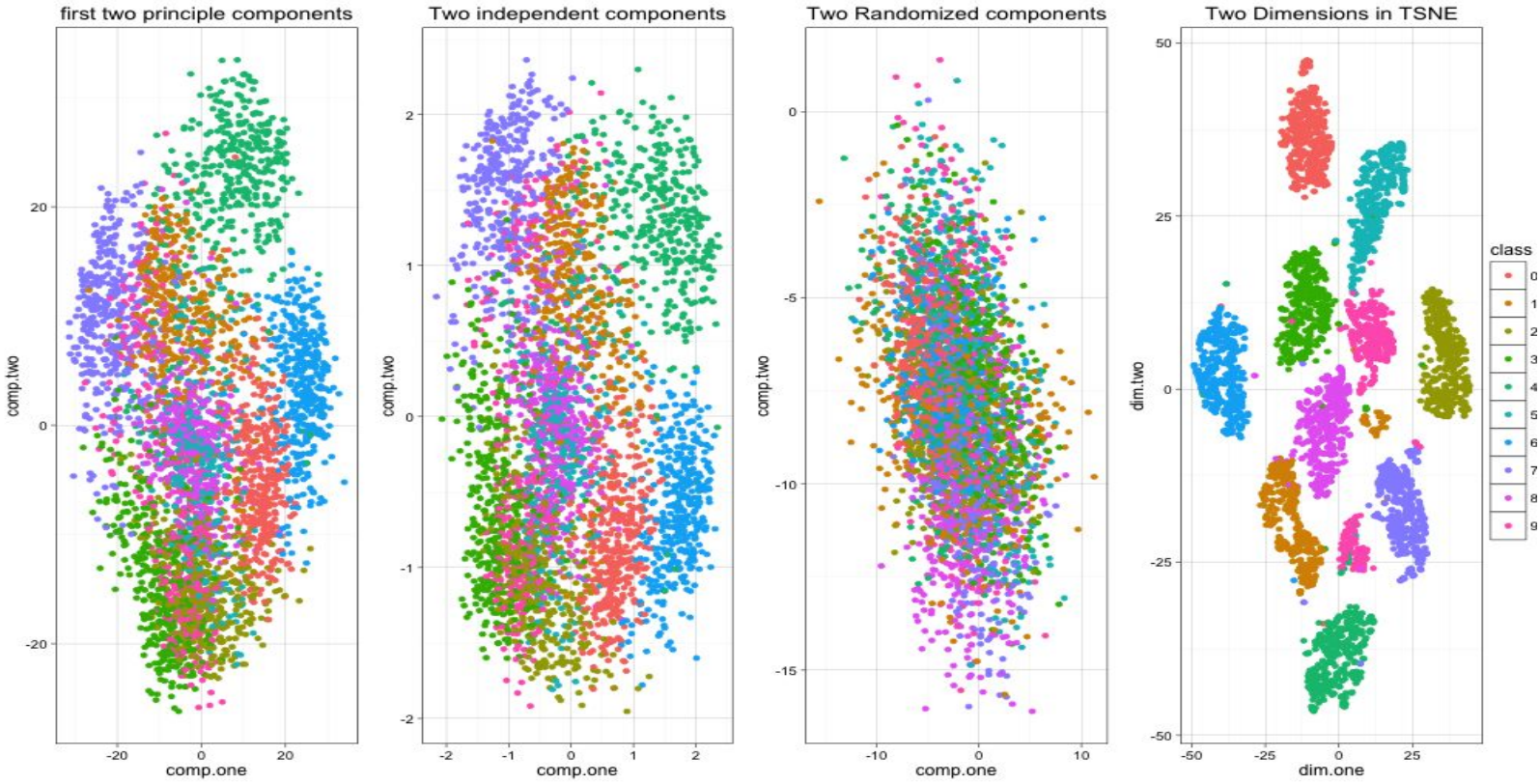
Next I ran expected maximization (EM) on the pixel tumor attributes with k=2. The table below shows the generated centers. EM generates similar cluster centers as kmeans. In this case the large attribute cluster is cluster 2, and cluster 2's center has larger values for each attribute then cluster 1's center. However, since EM is a soft clustering algorithm, EM's large attribute cluster is less extreme (smaller in attribute size) than kmean's large attribute cluster and EM's small attribute cluster is also less extreme (larger in attribute size) than kmean's small attribute cluster. This is because when generating centers, EM considers every observation in the dataset, therefore even observations far from the center still influence (making smaller or bigger) the cluster's center (albeit with much less magnitude). Although not displayed, EM's clusters also do a good job of partitioning the underlying tumor classes.

Cluster	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses
1	2.804268	1.060132	1.237351	1.177744	1.918719	1.016714	1.946011	1.043053	1
2	6.107690	5.276739	5.226461	4.510447	4.571990	6.115233	4.969463	4.727139	2.216616

Dimension Reduction

I ran 4 dimension reduction algorithms on the 64 pixel features: principal components analysis (PCA), independent components analysis (ICA), randomized projections (RP), and t-distributed stochastic neighbor embedding (t-SNE). Below are the results for the first 2 components of PCA, 2 component ICA, 2 component RP, and 2 dimension t-SNE.

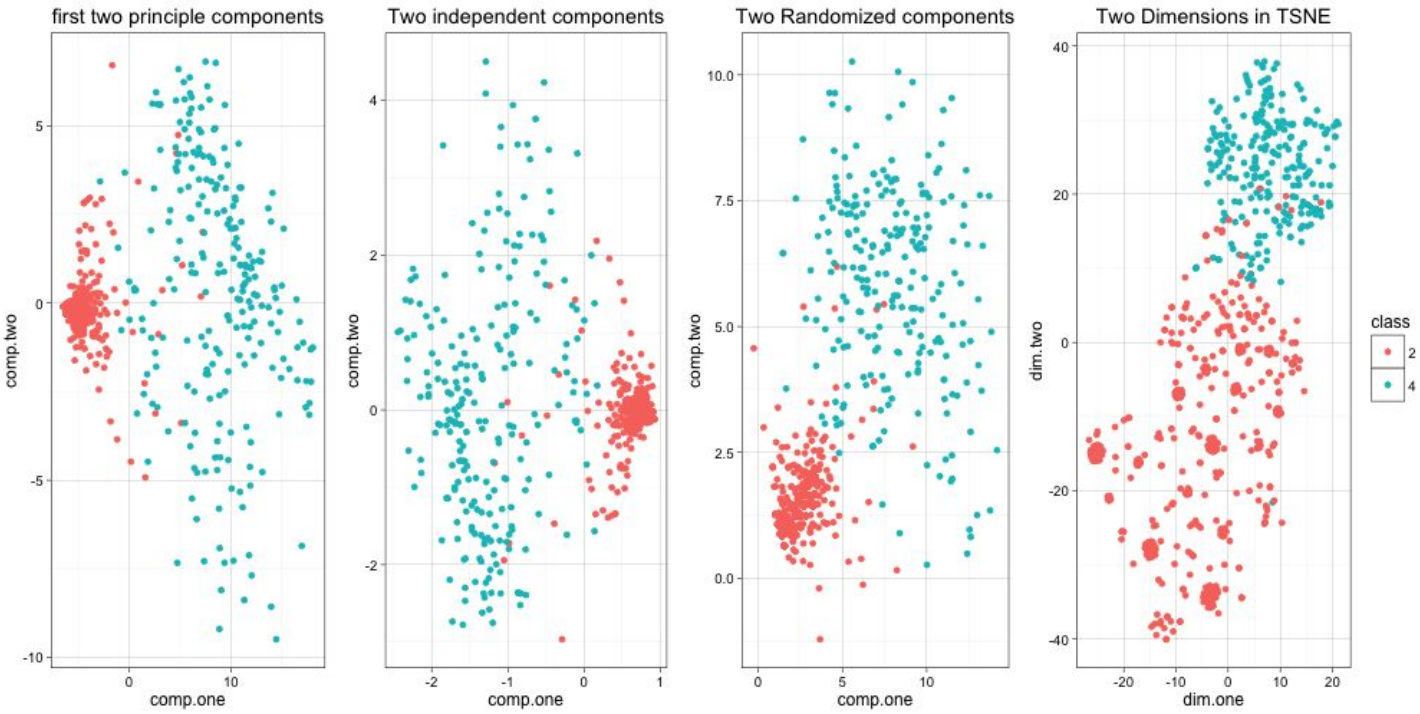
The image below shows that each dimension reduction algorithm contributes to creating a 2-dimensional plane where the various digit classes are linearly (or maybe using another kernel)



separable. t-SNE does a better job of separating the 10 classes in 2 dimensions than PCA and ICA who do a much better job than randomized projections. Why do we see this difference in "separable" performance in the dimension reduction algorithms? With regard to randomized projections, by mixing various pixels values in the 2 dimensions without any clue to the structure of the data, the algorithm actually obfuscates the difference in classes. PCA projects correlated variables into a set of uncorrelated components, thus we see something of a circular shape. ICA projects the variables into a set of independent components, thus we see something of a square shape (uniformly spread in each component). By maximizing variance or attempting to create independent components, PCA and ICA are able to preserve some of the structure of the data. As we saw with k-means in the clustering section, digits tend to cluster fairly well in euclidean space, thus t-SNE in which "similar objects are modeled by nearby points and dissimilar objects are modeled by distant points." (wikipedia) works extremely well at separating the classes in 2-D.

In the next section, I'm going to use these dimension reduction algorithms on the digits dataset in conjunction with k-means, EM, and neural networks. I will not use 2 dimensions for PCA, ICA, and RP, but instead select the number of dimensions using a number of different information measurements.

I also ran the 4 dimension reduction algorithms on the 9 tumor attributes. Below are the results for the first 2 components of PCA, 2 component ICA, 2 component RP, and 2 dimension t-SNE.

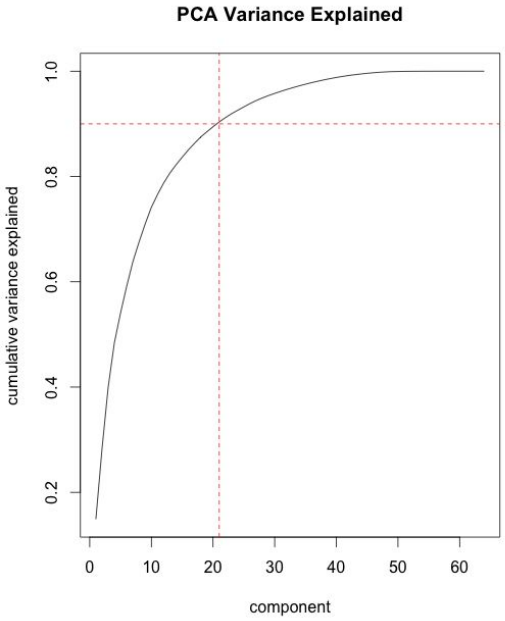


In particular interest in the above plots is that class 2 in the principal components plot has very little dispersion in component one or two. If you look at the variance of the 9 attributes for each class, class 2 has an average variance of 1.082571 across the 9 attributes while class 4 has

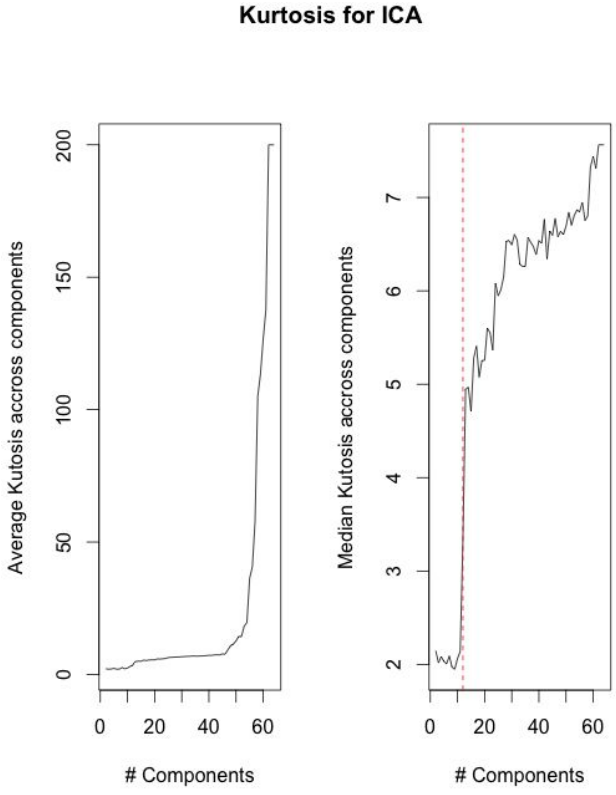
an average variance of 7.651895 across the 9 attributes. This explains the lack of dispersion in the principal components plot; there is little variance to preserve in regards to class 2. Said another way, class is much more homogeneous in its attribute values. Interestingly, randomized projections does arguable as well as PCA and ICA at separating the 2 classes. Why? As we saw in the clustering section that there is clearly a "large attribute" and "small attribute" cluster that map fairly well to underlying class structure. This means that each attribute likely has a value (like a split in decision trees) that is fairly good at splitting the different classes. Since randomized projections is weighing the contribution of different attributes randomly in terms of their loading on the dimension, picking any attribute and any contribution is likely to still result in class separation. t-SNE has perhaps the most obscure 2-D plot, largely because it's a nonlinear dimension reduction technique that attempts to preserve the neighborhood structure of the original dataset. That's why we see larger dispersion in class 2 despite its lack of variance in the original dataset.

Clustering on reduced dimensions (digits only)

The adjusted rand index is a measure of the similarity between two data clusterings; 0 being completely random and 1 being matching partitions. In this case, I will compare the output of k-means and expectation maximization on the various forms of the digits datasets (the full dataset and the different reduced dimension datasets) with the true class labels. In this way we can judge how valid the clustering methods are at capturing the underlying concept. Each trial of k-means or expectation maximization is always run with a **k** parameter of 10 (for the 10 digits). As I mentioned previously, I could have ran the algorithm with a different **k** parameter, but as we will see, the clustering results map extremely well with the class labels.

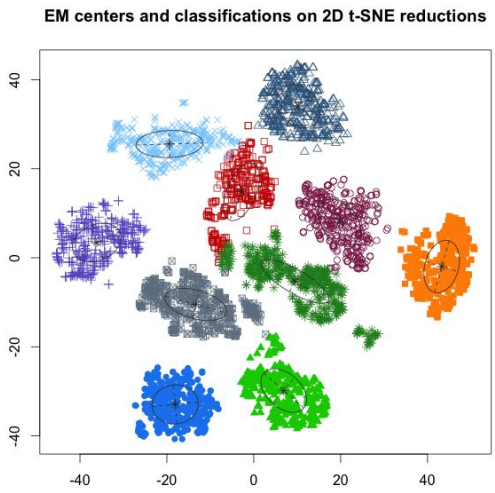


I chose 21 dimensions for PCA because by 21 dimensions, 90% of the original variance is explained. This is demonstrated in the plot to the left. I chose 12 dimensions for independent components analysis because it gave a reasonable median kurtosis value for the 12 components. This is demonstrated in the graph to the right. Using the highest average kurtosis across the independent components would have led me to use all 64 dimensions, so I resorted to looking at the median. High kurtosis (particularly above 3) implies a non-normal distribution. ICA attempts to



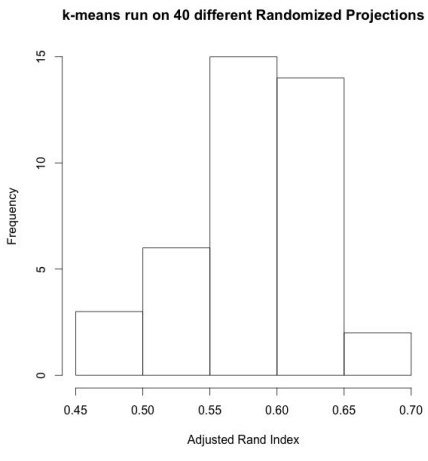
create independent non-gaussian components and by measuring the median kurtosis we get a sense of how well the algorithm performed. I chose 40 dimensions for randomized projections under the idea that randomized projections will require more components than PCA, so I thought twice as many was a good number. I chose 2 dimensions for t-SNE since the 2 dimensional plot showed such good separation of the different classes. Below is a table of the adjusted rand index for the various clustering algorithms. Again k-means was always run with 200 restarts and EM has no restarts due to the R package I'm using.

Algorithm	**Adjusted Rand Index**
K-means on raw data	.68
Expectation maximization on raw data	.66
K-means on PCA (21 dimensions)	.67
Expectation maximization on PCA (21 dimensions)	.76
K-means on ICA (12 dimensions)	.71
Expectation Maximization on ICA (12 dimensions)	0.78
K-means on RP (40 dimensions)	.51
Expectation Maximization on RP (40 ds)	.59
K-means on t-SNE (2 dimensions)	**0.92**
Expectation maximization on t-SNE (2 dimensions)	**0.93**



The best combination of clustering algorithm + dimension reduction technique is expectation maximization run on t-SNE in 2 dimensions. The centers and different cluster labels are visualized to the left. You can see the different bivariate gaussian shapes, most of which are spherical, but some of which (blue for example) look close to perfect circles.

Ignoring t-SNE who does a spectacular job separating the different classes, what happen? Randomized projections seem to always decrease the performance of both k-means and EM in regards to the adjusted rand index. Why? Well in this problem each pixel is important to the class label. Randomized projections may help reduce the number of dimensions, but it arbitrarily applies different loadings on the pixels values in the 40 dimensions, thus obfuscating the structure of the pixels that would lead to separation. In a problem space where we had potential noise variables, randomized projections may have done



a better job, but in this case where every pixel is important, the random mixing only seems to obfuscate the intended signal. However, this was just one set of random projections, so I decided to run k-means on 40 different trials of RP. The histogram of the adjusted rand index across the 40 trials is on the previous page. It goes to show that while in general RP does achieve the same performance of k-means on PCA or ICA reduced dimensions, it can achieve similar performance **if** RP randomly selects a "good" set of loadings the preserve the underlying structure of the data.

K-means's adjusted rand index is unimproved by PCA and only slightly improved by ICA, while EM's adjusted rand index is improved by PCA and ICA. Why? My theory is that EM worked better on the reduced dimensions since the influence of far away points on the centers was only on 21 or 12 parameters instead of 64 parameters.

Classification on reduced dimensions and clusters (digits only)

I ran neural networks on various data formats of the digits dataset. In particular, I ran it on a reduced 21 dimensions of PCA, reduced 12 component ICA, reduced 40 dimension RP, reduced 2 dimension t-SNE, all 64 pixel features plus the assigned k-means cluster, all 64 pixel features plus the assigned em cluster, all 64 pixel features plus the assigned k-means cluster on t-SNE, and all 64 pixel features plus the assigned em cluster on t-SNE. The results are displayed below.

Algorithms	**Accuracy on Test Set**	**Iterations till Convergence (backpropagation)**
NN	95.15%	≈ 100
NN + PCA	97.33%	85
NN + ICA	95.71%	197
NN + RP	95.00%	122
NN + tSNE	99%	≈ 50
NN (all 64 features) + K-means cluster on raw data	95.83%	98
NN (all 64 features) + EM cluster on raw data	95%	83
NN (all 64 features) + K-means cluster on tsne	98%	87
NN (all 64 features) + EM cluster on tsne	98%	97

In general dimension reductions improves performance and decreases the number of iterations needed for convergence. The only exception being randomized projections which have about the same performance as the original neural network on all 64 pixel features.

t-SNE provided such a good 2-D representation of the data (in which the naked eye could easily identify the different class clusters as show in a visualization earlier), that the neural network that learned the class boundaries on the reduced t-SNE data was close to perfect on the

test set. More interestingly (since PCA is an industry standard and significantly faster than t-SNE), however, is that a neural network run on 21 dimensions of PCA, performed better than a neural network on all 64 dimensions. Why? Well, there are less weights to estimate and non-important variables such as the upper left corner of the image (which in all training and testing instances has no pixel value) are removed through the process of reducing the number of dimensions and preserving variance. Furthermore, the neural network on PCA converged faster since it was learning a simpler concept and did not need to learn, for example, that the upper left corner had no bearing on the actual digit.

In addition, adding the cluster output of k-means or em to a neural network caused performance that was on par with a neural network run on all 64 pixel features, but decreased the time to convergence. Why? Well as started earlier, the k-means and em cluster outputs map fairly well to the underlying digits classes (in particular 4, 7, 5, 0, and 6 as seen in the contingency table for kmeans on page 1), thus the neural network was able to learn the "easy" digits using the clustering algorithm assignments, and was able to learn the "hard" digits using the actual pixel features themselves.

Conclusion

Dimension reduction and clustering on the digits and cancer dataset enabled a deeper understanding of the datasets. We were able to visualize different clusters in the data, see linearly separation in 2-D using the dimension reduction algorithms, and better understand characteristics of the feature-set that related to the underlying classes of interest. Normally, these techniques would be performed before running classification algorithms and would give good insight into what algorithms could/should be used. Finally, performing dimension reduction augmented the neural network classification algorithm by either increasing its performance or decreasing its time to convergence.