Anish Moorthy *Project 3: Unsupervised Learning*

Introduction

This paper explores the topics of unsupervised learning and dimensionality reduction, and is split into three parts. In the first part, I cluster each dataset using the K-Means and Gaussian Mixture Model algorithms, and evaluate the results using several metrics. In the second part, I use the PCA / ICA / Randomized Projection / Variance-Threshold feature selection algorithms, and report various results such as (a) reconstruction quality (b) effectiveness of dimensionality reduction and (c) visualizations. Furthermore, I re-cluster each dimensionality-reduced dataset (using the same hyperparameters as above) and evaluate clustering fitness. In the final part, I train neural networks on a dimension-reduced (and possibly clustered) dataset and compare performance with Project 1

Dataset 1: CIFAR-10

As in Project 1, I use the CIFAR-10 image classification dataset. This dataset consists of fifty thousand 32x32 images, each drawn from one of 10 classes (the class distribution is balanced). This dataset presents an interesting challenge due to the relatively high dimensionality (3072) of the data, and furthermore because of the correlation between dimensions: a single dimension of the data (corresponding to a single channel of a single pixel) reveals almost nothing of the answer. It is the *combination* of dimensions where information lies.

Therefore, this dataset is especially interesting from the dimensionality-reduction point-of-view. An effective dimensionality-reduction algorithm should capture the dependencies between different dimensions of the data, which are highly correlated

Dataset 2: IMDB Sentiment Analysis

I also re-use my other dataset from Project 1, drawn from the domain of natural language processing. This dataset consists of sixty-thousand movie reviews, each labeled “positive” or “negative.” Although natural language has a rich sequential structure, to simplify the problem I preprocess the data in a way which destroys said structure in order to simplify the problem. Namely, I represent each instance as a bag-of-words vector of the 10,000 most common words.

Being preprocessed in this way, this dataset also presents an interesting target for dimensionality reduction. Many of the most common words are, of course, uninformative words such as “and” or “the” or “a” which appear at a relatively constant rate throughout the dataset. An effective dimensionality reduction algorithm should be able to determine that the dimensions corresponding to such words are simply noise, and discard them. Furthermore, reductions would ideally be able to accordingly weight dimensions corresponding to words such as “great” or “bad” which are good predictors of a datum’s label.

Part One: Clustering

Due to the high dimensionality of both datasets, it is infeasible for me to evaluate clusterings via direct graphing. However because my datasets are labeled, it is possible for me to evaluate the “closeness” of labelings with the true labels, in a variety of senses. I use three metrics to do so

1. *Adjusted Rand Score*: This metric evaluates similarity of two labelings based on the number of datapoints which are in the same/different clusters in both.
2. Homogeneity: An information-theoretic metric. Given two labelings “True” and “Predicted,” this measures the average information which a “Predicted” label reveals of the “True” class
3. Completeness: Another information-theoretic metric which is the “dual” of Homogeneity. This measures the average information which a “True” class reveals of the “Predicted” label.

Each is a similarity function ranging from 0 to 1, thus I usually graph each at the same time. To choose *k* for my clustering algorithms, I clustered the training set using many different # of clusters (which is what *k* represents of course), used said clusterings to predict a validation set, and chose the value which corresponded to a peak/plateau in the scores.

Note that increasing the number of clusters tends to increase scores in my results below. A valid concern is that the clusterings might just be random and are increasing in score only due to increasing in size (a type of overfitting). However due to the information-theoretic definitions of Homogeneity/Completeness, a random clustering will actually correspond closely with scores of around 0. Thus the observed increases actually do represent an increase in cluster/labeling fitness.

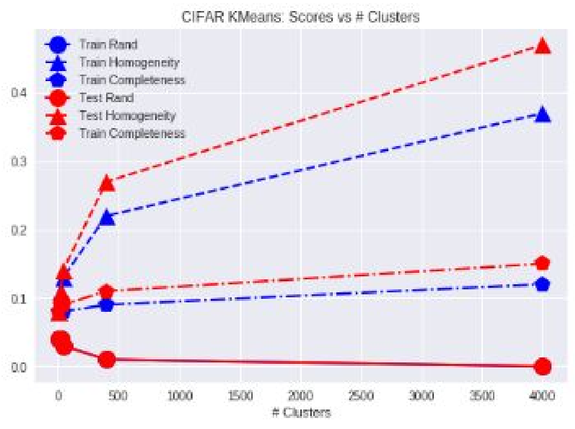
The distance functions used in K-Means/GMM are a source of domain knowledge. For both datasets, I tested performance with different distances (L1, L2, Cosine).

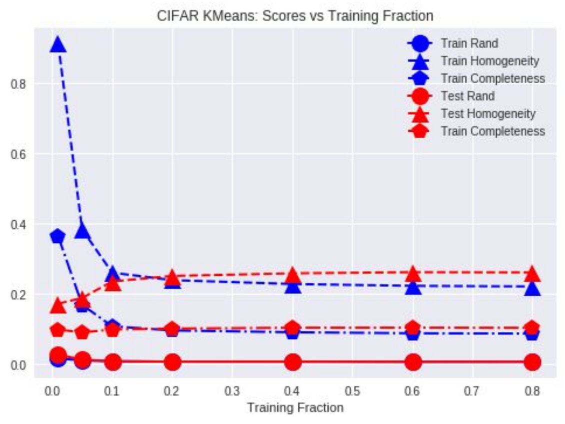
1. For CIFAR-10, I found that L2 (standard Euclidean distance) worked best. L2's advantage over L1 is best explained by the fact that I normalized each dataset, thus L2 more heavily penalizes large deviations from the mean. Cosine distance could theoretically work, but in practice was less performant than L2. This is probably due to the fact that the "averaging" step itself of K-Means and GMM corresponds to a minimization of the average L2 norm: ie L2 is the most theoretically-justified norm to use.
2. For the IMDB dataset, I found that L1 (Manhattan) distance was the optimal metric, despite the theoretical advantages of L2. The L1 norm between two bag-of-words vectors is very intuitive, being the number of dimensions in which the two vectors differ. Though a crude measure, this distance function evidently is the best to capture the space structure.

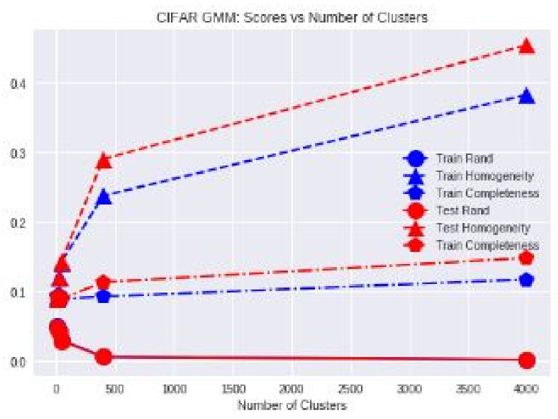
Note 1 : Unless otherwise noted, I use a 70/15/15 percent train/val/test split everywhere in this report

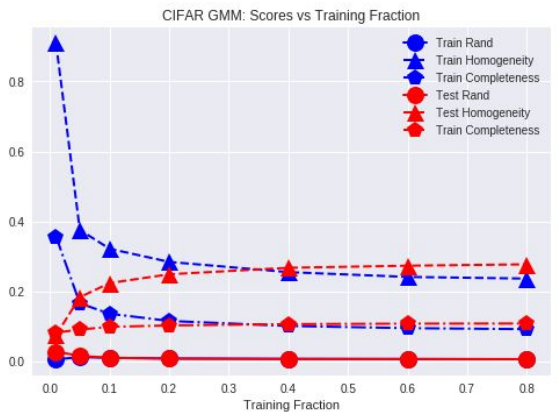
Note 2: Rand/Homogeneity/Completeness = RHC scores

Clustering :: CIFAR-10 - Test RHC Scores for K-Means/GMM are (.01/.27/.11 & .01/.29/.11)









First of all, we notice that the graphs are extremely similar between K-Means and GMM. The explanation is that both algorithms are actually just "hard" and "soft" versions of the same algorithm. In this problem, where inter-image distances are very large (even after normalization), GMM becomes increasingly "hard" since the vast majority of weight for a given point will be assigned to the closest cluster: essentially converging to the "hard" K-Means

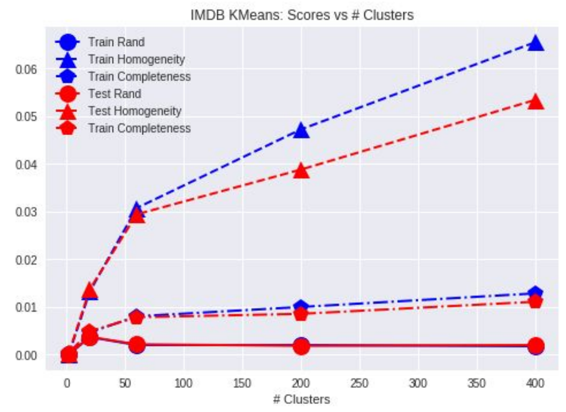
As # of clusters increases, rand score flatlines. This is to be expected: due to the way that rand score is defined, as the number of clusters increase it is increasingly likely which two elements in the same class will be assigned different clusters. This is why I evaluate with an ensemble of metrics: homogeneity and completeness reveal that the more available clusters allows the clusterings to approach the true labeling more closely. Two factors prevented me from further increasing the number of clusters.

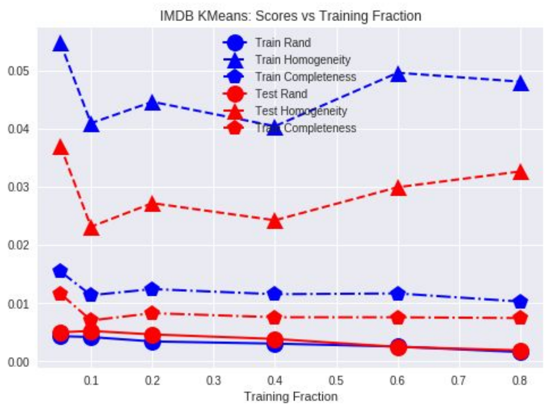
1. Computation, which scales linearly in *k*: 400 clusters required 10 minutes to run, and 4000 required 95. Even 4000 clusters was impractical, I only provide it for illustration and actually used 400 for further experiments
2. Potential for overfitting: even though both train and validation scores are increasing, I was wary of using too many datapoints and overfitting by essentially allowing each cluster to represent a single datapoint. Thus I chose *k* at a relatively low, computationally feasible value

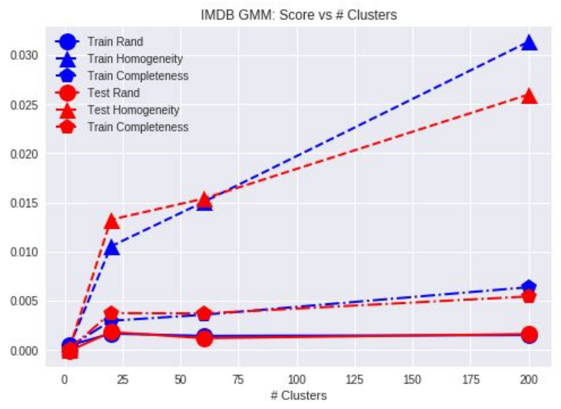
The curve of scores to fraction of the training set used for clustering is even more interesting. The overall shape is relatively standard, but the scores flatline extremely quickly. Only 20% of the training set is needed to achieve near-optimal performance. This actually mirrors my results from Project 1, though the flatline was less pronounced there. I have no direct explanation, but it is evident that even this small amount of data is enough to capture the contours of the dataset (in a rigorous sense: ie generating a Voronoi diagram similar to the true distribution) well enough.

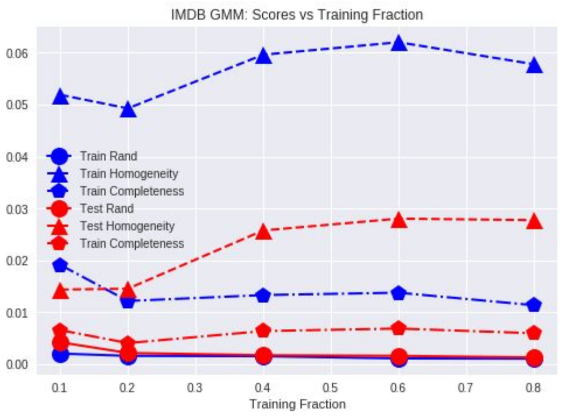
Given both results above, a path for even further performance improvement would be to cut the size of the training set and increase the number of clusters by the same factor. Because training time is linear in both, the effects on time would essentially cancel out but (extrapolating from the first graph) both training and validation scores would increase, ie the clustering would be more accurate.

Clustering :: IMDB - Test RHC Scores for K-Means/GMM are (0/.04/.01 & 0/.03/.01)









Firstly, I should note that the scores obtained are approximately an order of magnitude lower than those obtained on CIFAR-10. In fact the clusters are not much better than random, though we may still observe clear trends in the scores. Here we see a greater degree of difference between K-Means and GMM algorithms: this is most likely due to the fact that the average intra-cluster distance is much lower than with CIFAR (about fifty times lower to be precise). However, this works against GMM: the inter-point distances are small enough that there is actually too much randomness in GMM (ie the class distributions for points don't converge quickly enough), leading to GMM achieving only about half the performance of K-Means by all metrics.

Due to the increased dimensionality of the data, running either algorithm with thousands of clusters simply took too long to terminate. However as in the CIFAR-10 dataset, we observe that adding more clusters tends to increase performance. Thus I chose *k* to be 400 and 200 for K-Means and GMM respectively (with this config GMM takes about 15 mins, and K-Means takes ~7mins)

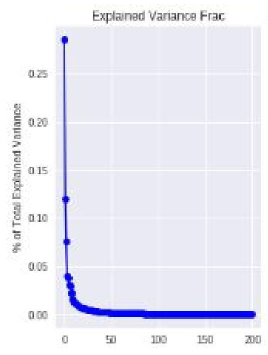
The performance vs fraction of data curves for GMM and K-Means are substantially different from the CIFAR-10 dataset however, which is very interesting. The gap between train and test homogeneity cores remains relatively large, indicating that our model is overfitting to some extent: furthermore, said gap reduces somewhat. We can draw two main conclusions from the above results

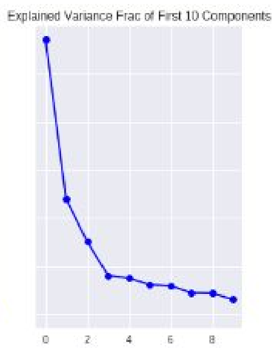
1. To even have a chance of achieving acceptable performance using K-Means/GMM, it is likely that a much, much larger dataset is needed.
2. The IMDB dataset is noisy in that many of its components don't matter, and the effects of those which due (ie the distances) are obscured by distances between irrelevant components. In fact we already knew

this (see my discussion above), but it is good to have a confirmation via actually running an algorithm on the dataset.

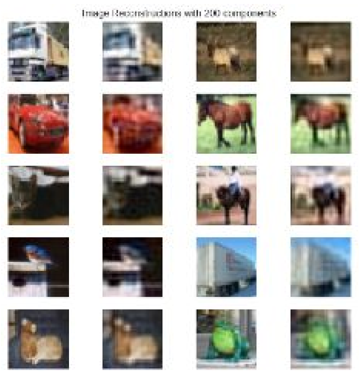
Part Two (A) :: Dimensionality Reduction on CIFAR-10

PCA :: CIFAR





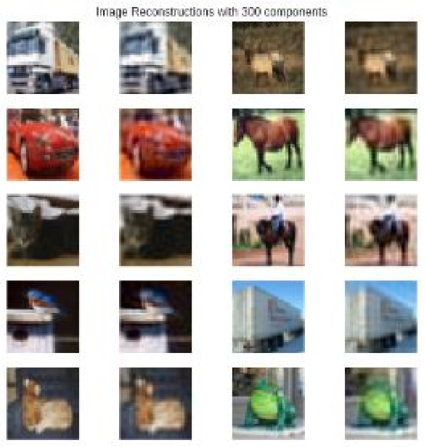
First 20 PCA Axes

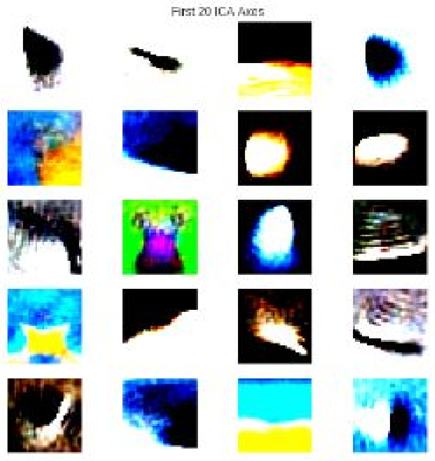


PCA on the cifar dataset produced some interesting results. Firstly, we observe that just a few eigenvectors have eigenvalues of any appreciable magnitude (in fact, this is a common phenomenon in SVD decompositions of images). Reconstructions of images (right column) from the validation set (left column) from only 200 dimensions are, astonishingly good: the reconstructions are blurrier, but capture the general concept extremely well. The PCA eigenvectors seem to correspond to shapes and colors: it appears that PCA yields a generalizable dimensionality reduction strategy for images

K-Means and GMM clustering on PCA-reduced data gave test RHC scores of **(.02/.18/.19 & .02/.16/.19)**, showing that C is increased at a significant cost to H. The RHC between clusterings on (a) dataset w/o PCA and (b) PCA'd dataset are **(.01/.31/.38 & .01/.26/.28)**: thus the clusterings are somewhat similar. This difference is probably due to the heavy dependence of cluster centers on initialization in boh K-Means and GMM.

ICA :: CIFAR

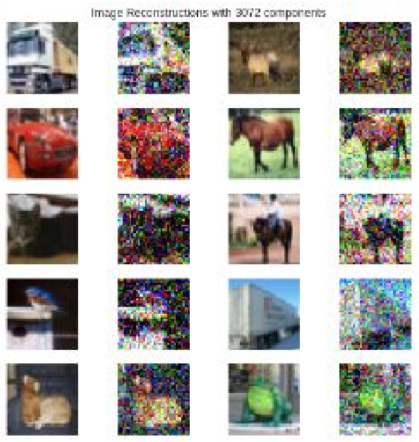


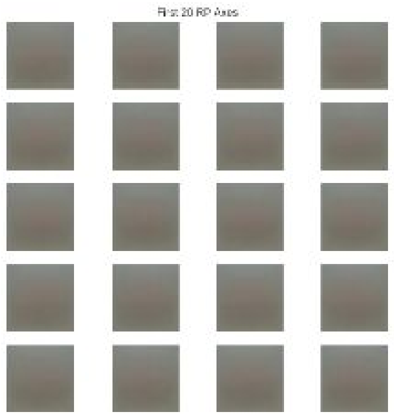


ICA produces results which are strikingly dissimilar from PCA. It maintains the same sort of dimensionality-reduction power (the above reconstructions are from 300 components), but the axes appear much... odder. Traditionally one finds that the independent components of images are edges, but here again they are to be shapes, though in much brighter hues. I hesitate to call this an overfit, as the reconstructions are drawn from a validation set and perform well: instead, I would conclude that the CIFAR-10 dataset is probably small enough that a different, more specific set of independent components is discovered on this dataset

K-Means and GMM clustering on ICA-reduced data give RHC scores of **(.02/.15/.08 & .00/.01/.01)**. The RHC between clusterings on the ICA'd/non-ICA'd dataset are **(.01/.24/.34 & .06/.17/.35)**: thus the clusterings are somewhat similar. On all metrics ICA seems less performant than PCA and RawClustering, which I believe is due to the small, seemingly uniform coordinates of each datapoint in ICA (explained in final section), which slows the convergence of clustering.

Randomized Projections :: CIFAR



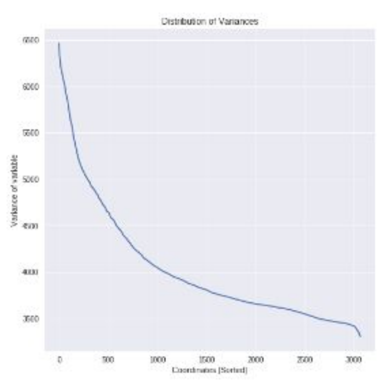


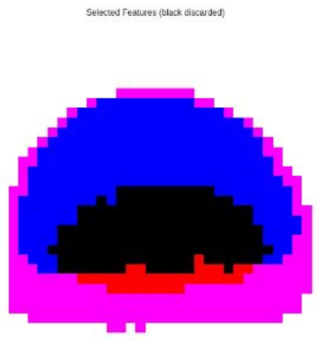
Randomized projections perform terribly (consistently across many runs): using 3072 dimensions (the original image size!), RP cannot even reconstruct images without significant noise. Appropriately, the axes appear to just be random noise. My explanation for RP's terrible performance is that pixels in images are highly spatially dependent, and thus having basis matrices where each dimension is independent of all the others actually *hurts* representation power since this actually makes it more difficult to construct images with contiguous regions of distinct colors which are *not* the "mean" image color

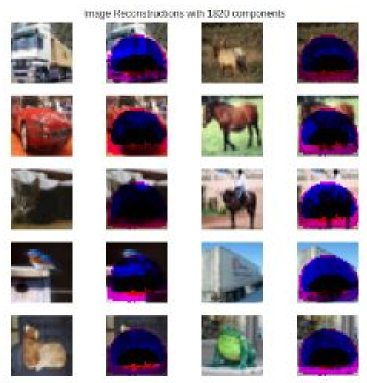
K-Means and GMM clustering on RP-reduced data give RHC scores of **(.01/.28/.11 & .01/.29/.11)**, which are almost-*identical* with the RHC of RawClustering: backing my hypothesis that RandProj does basically nothing. The RHC between clusterings on the RP'd/non-RP'd dataset are **(.17/.65/.65 & .16/.64/.66)**, indicating a high degree of similarity: whatever projections RandProj *does* to appears to maintain the geometry of the space (relative distances between points), leaving the cluster structure intact.

Variance-Threshold Feature Selection :: CIFAR

Here, we discard dimensions with variance less than a given value (I picked 3500, leaving 1700 dims)





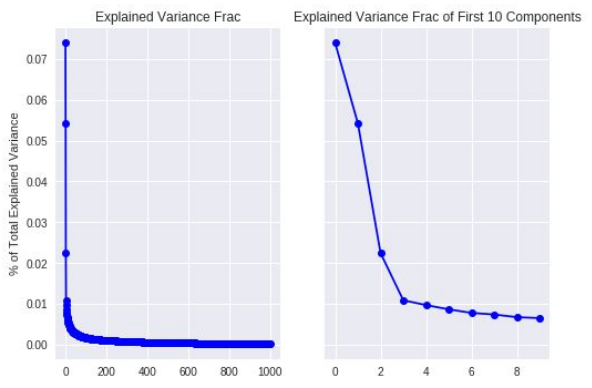


In the left column (apologies for small font), I plot the (sorted) variances of each attribute. In the middle is the constructed filter (white corresponds to each channel of the pixel being chosen, while black corresponds to all filters being discarded). Counter-intuitively, the filter actually focuses less on the center of the image and more on the background as the variance-threshold increases. This is clearly undesirable, since most of the semantic information of the CIFAR-10 images is found in the center.

K-Means and GMM clustering on FS-reduced data give RHC scores of **(.01/.27/.11 & .01/.28/.11)**, again identical with the RHC of RawClustering. Again, the reduction therefore maintains the geometry of the space. In fact, this tells us that Variance Selection isn't totally screwing up by discarding the center region of the image: this result shows that just considering the background is enough to get the distance between two data points. RHC between clusterings on the FS'd/non-FS'd dataset are **(.1/.59/.59 & .16/.61/.61)**, shows that the reduction preserves labeling similarity.

Part Two (B) :: Dimensionality Reduction on IMDB

PCA :: IMDB



Again, we observe the general phenomenon that most of the eigenvalues given by PCA are close to zero, and that their corresponding eigenvectors can be discarded. Using PCA to discard all but the 1000 most significant eigenvectors (from an original dimensionality of 10,000) seems to provide a good balance. Reconstruction is fairly good, with an average L1 distance of 2000 from a word to its reconstruction. The axes aren't quite as interpretable as they are in the CIFAR-10 case, but do seem somewhat related. For instance, one axis corresponds to the string "episode film great love movie one season series show time tv watch would year," which is probably corresponds to positive reviews of TV shows. Another axis string is simply "bad like movie watch," probably for negative reviews ("bad" appears in no other axes)

K-Means & GMM clustering on PCA-reduced data give RHC scores of **(.01/.03/.02 & 0/.01/0)**, on par with RawClustering. This shows that PCA is doing a good job of reducing dimensionality, but is overall neutral on projecting the data into a more favorable space. RHC between clusterings on the PCA'd/non-PCA'd dataset are **(.06/.1/.33 & .06/.19/.3)**.

ICA :: IMDB

Using ICA gave slightly better results than PCA. Even after reducing to 800 dimensions, the average datapoint-reconstruction L1 distance was just 1500. However, I encountered two main disadvantages. The first was that ICA was very slow, taking about 30 minutes to run: this is expected however, since it spends extra computational time attempting to find axes which are independent. A second disadvantage, however, was that the axes were much less interpretable than even IMDB PCA. Several axes were almost uniform across each data dimension, while others are coherent but don't correspond to "topics" as we expect the ICA of a text corpus to be. For instance, one axis is "albert alejandro cassandra concoct crackl ethan feroci finney flourish gamut gina hank haul hawk hoffman holden jenkin jewelri juici katharin lumet marisa masterson sidney steiger tomei tote tyron", which clearly corresponds to names (famous actors perhaps): obviously my expectations of immediate interpretability were unfounded.

K-Means & GMM clustering on ICA-reduced data give RHC scores of **(.02/.06/.02 & 0/.01/.01)**, again about equal par with RawClustering. RHC between clusterings on the ICA'd/non-ICA'd dataset are **(.05/.1/.35 & .06/.17/.35)**, completely average for this type of comparison.

Randomized Projections :: IMDB

Again, randomized projections performed so terribly they're almost not even worth mentioning: even with no actual dimensionality reduction, average data-reconstruction distance was nearly 8000. Again, I submit that the heavily sequential/spatially correlated nature of the data is the reason why randomized projection performs so poorly on my datasets, though if course it could be random choice simply isn't the greatest of strategies for dimensionality reduction :)

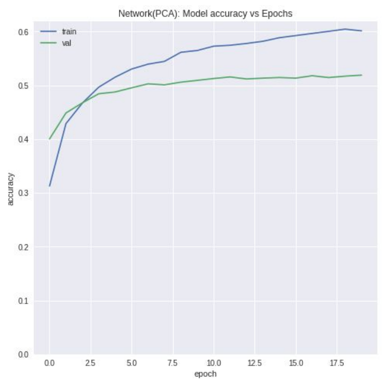
K-Means & GMM clustering on RP-reduced data give RHCs **(.00/.01/.00 & 0/0/0)**, another abysmal showing for RP. RHC between clusterings on the RP'd/non-RP'd dataset are slightly better, at **(.07/.1/.43 & .06/.16/.34)**, but is still below-average.

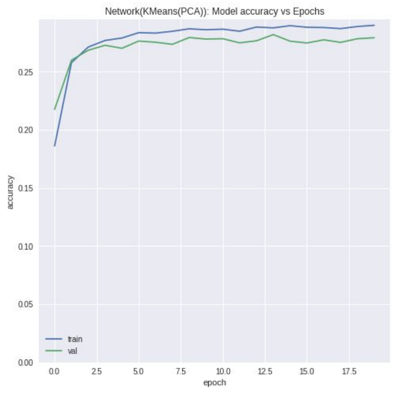
Variance-Threshold Feature Selection :: IMDB

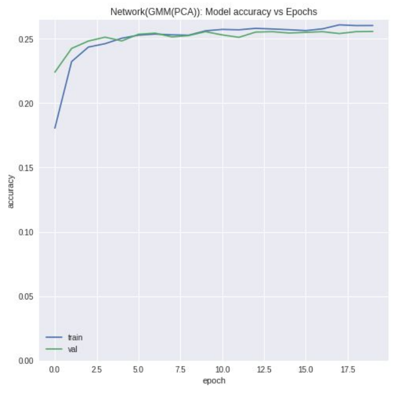
Selecting the variance threshold such that only 600 components are chosen actually gives great performance: average data-reconstruction distance is about 3500 (a count inflated because reconstruction simply substitutes zeros in many places), but the many of the selected dimensions correspond to high-information words such as "enjoy", "great", and "bad." By the nature of variance, the selector will naturally pick up on informative words which are commonly used to describe movies, showing that given the right domain even very simply methods can prove very effective

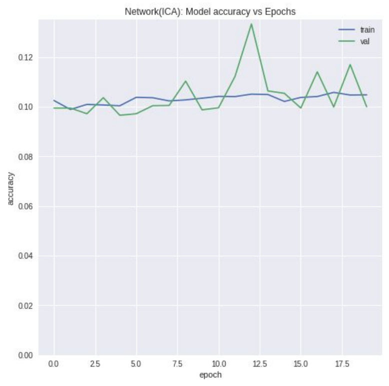
K-Means & GMM clustering on FS-reduced data give RHCs **(.00/.09/.02 & 0/.08/.01)**,which is significantly better than RawClustering. RHC between clusterings on the RP'd/non-RP'd dataset are also high, at **(.27/.64/.46 & .17/.59/.45)**. Again, these metrics would seem to show that even this simple feature-selection algorithm can adequately capture the geometry of the space.

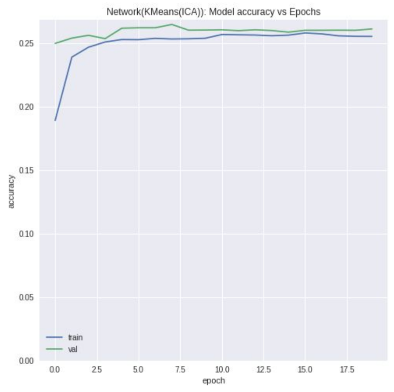
Part Three :: Neural Networks on CIFAR

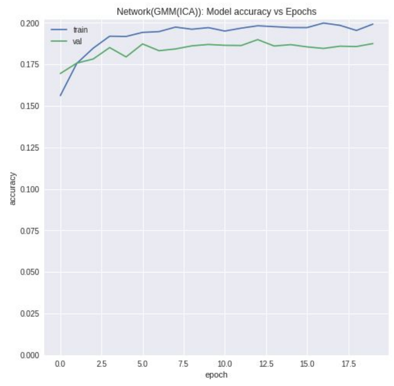


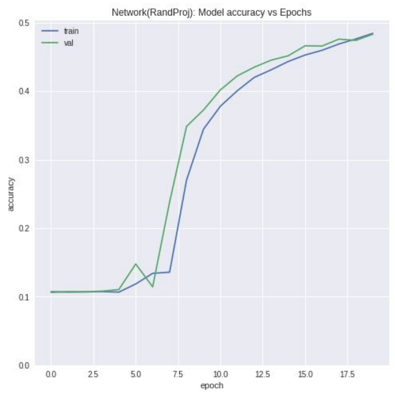


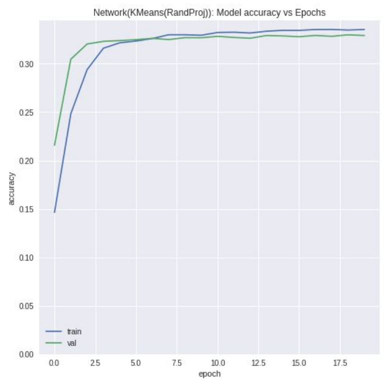


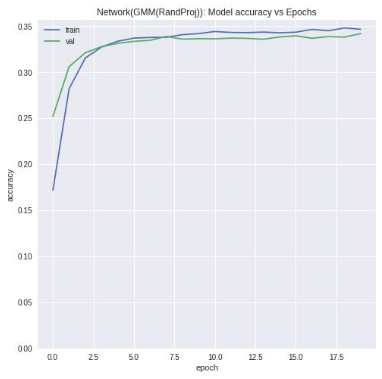


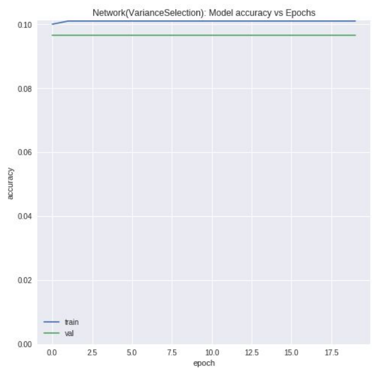


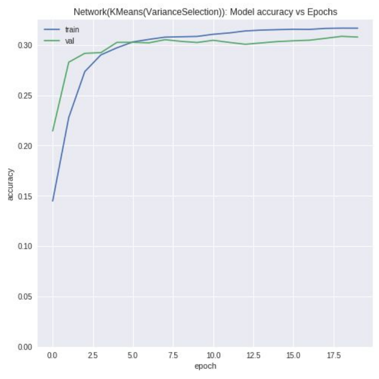


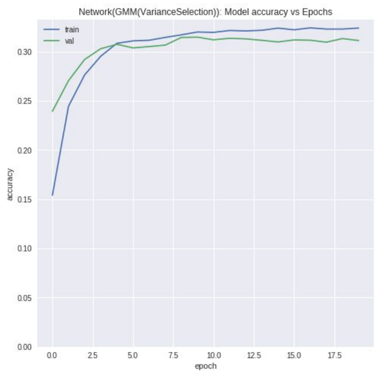












My apologies, **you can read the text on the above graphs by zooming in**.

For every single one of the above experiments a feed-forward network with hidden layer sizes of [50, 50] was used, along with various other hyperparameters (all with the same values as in Project 1).

There's quite a bit of good (and often extremely confusing!) information contained in the above graphs.

1. The first thing to note is training times were reduced more or less linearly with the dimensionality of the data, which wasn't such a big deal here, but would definitely make a difference in the real world with much larger datasets
2. The second thing to note is that the best accuracy I obtained in Project 1 was 35%, whereas here we hit validation accuracies of ~50% (via networks on PCA(Data) and RandProj(Data)). Thus, we observe that ***dimensionality reduction can significantly improve performance***

Onto the confusing parts. For PCA and RandProj, we observe that networks trained on the dim-reduced data have significantly higher accuracy than networks trained on clusters(dim-reduced data). This makes sense: replacing a datapoint with its cluster label loses information, information which the neural network might have been able to leverage. However, this trend is astonishingly reversed with ICA and Variance-Threshold Feature Selection!

Clustering is a function, and it is a theorem of Information Theory that functions can only reduce the information of a sample. Thus in ICA and FS, even when a network performs no better than random the information must actually be there, just obscured in such a way that the network can't easily learn it. I do not have a quick-and-easy explanation for the mechanism via which this occurs, but I have two theories

1. The geometry of the distribution (ie the underlying Voronoi diagram of the data) is too complex for the network to model. Since we are using a relatively small network, this is a possibility: clustering the data essentially corresponds to a dimensionality-reduction of the Voronoi diagram, thus projecting the data into a space where it is learnable by our network. Possibilities on why this only occurs with ICA and VarianceThreshold below.
2. Motivated by the odd ICA results on IMDB and the strange graphs, I re-ran ICA on CIFAR-10 and saw that coordinates for data points appeared small and uniform (they would have to be small, given that amount of white in the ICA axes). Because training on such small, similar values, the gradients will be much weaker and less informative in general. This will of course slow down the training process and make it appear that the network has not learned anything (in fact, there is a slight upward trend in the training accuracy for ICA).