Machine Learning - CS 7641 Assignment 3

Anthony Menninger
Georgia Tech OMSCS Program
amenninger3
tmenninger@gatech.edu

Abstract

This paper explores several unsupervised learning algorithms. It uses two datasets: Census Data labeled to income above or below $50,000 and MNIST handwritten digit image data for digits 0-9. Clustering is performed on the original datasets, then several dimensionality techniques are used to reduced the feature set and then clustering is preformed again. Finally, the reduced datasets are tested using learning algorithms.

Introduction

The first data set is from 1994 and 1995 current population surveys conducted by the U.S. Census Bureau [2] from the UC Irvine Machine learning site. The classification task is to determine if the user type for each instance has an income above $50,000 or below ($94,000 in today's dollars). There are 40 usable features, with both continuous features, such as age, and discrete, such as married status or sex. The training set has 199,523 instances and the test set has 99,762 instances. Because many of the features in the file are string based, I created transformed instances using the sklearn preprocessing LabelEncoder to translate discrete string values into consistent numeric values. This was necessary for algorithms that only used numeric features. I chose this data set because it the training dataset seemed relatively large and I expect somewhat noisy with a smallish number of features.

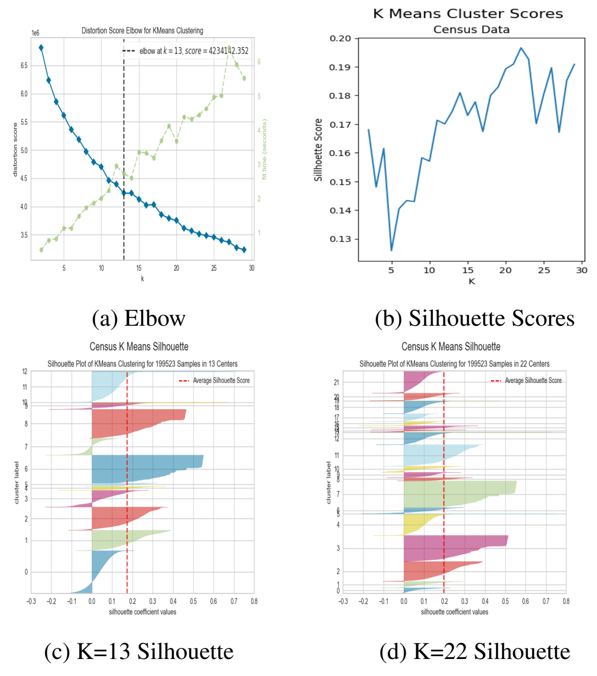
The second data set is The MNIST Database of Handwritten Digits [2]. This consists of instances of 28 X 28 greyscale images of the digits 0-9. One transformation performed was that the values where scaled to 0 to 1, from 0 to 255. The images were flattened, creating 784 features, one for each pixel. There are 60,000 training instances and 10,000 test instances. I chose this because it is a good comparison to the first data set, with a very different type of data (images), which leads to significantly more features (784). In addition, each of the features can be thought of as related to each other, as they are all positional in a two dimensional grid, while the first data set features do not have any necessary relation between themselves ie: age is not related to sex.

For this assignment, the core python library used was sklearn [3], which has many unsupervised learning algorithms. In addition, an enhancement Yellowbrick [4] was also used for some visualizations.

Clustering

K Means

K Means creates K clusters based on the distance of instances to cluster centers. A key question is what is the optimal number of clusters. An Elbow Plot looks at the intra distance within clusters at different K, and then picks the K where intra-distance change slows down, or the "elbow" of the plot. A silhouette plot charts the intra cluster distance and references it vs the next nearest cluster intra cluster distance which can help show where instances may want to be in a different cluster. For a given K, a silhouette chart can provide visual insight for qualitative review. Both methods were used in choosing an optimal K. sklearn was used for KMeans clustering and yellowbrick was used for visualization.

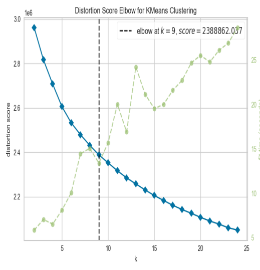


(a) Elbow (b) Silhouette Scores

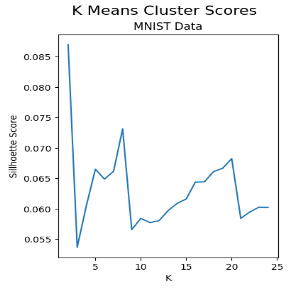
(c) K=13 Silhouette (d) K=22 Silhouette

Figure 1: Census K Means Review

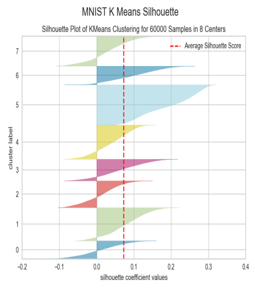
Figure 1 shows the Kmeans clustering review for the Census data. Figure 1a shows the elbow plot, which indicates that 13 is the optimal selection. However, the chart is bumpy, which makes it a bit trickier to use. Figure 1b shows the silhouette scores, where K=22 shows the highest score. For silhouette scores, reference to the individual charts is also useful, as we look to have balanced clusters that meet the score average. Figures 1c and 1d show K=13 and 22 respectively. Neither is a particularly clean silhouette, which is perhaps related to the bumpiness of the elbow chart. I would select K=13 for the optimal cluster size.



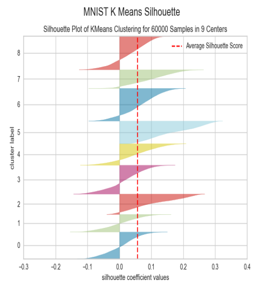
(a) Elbow



(b) Silhouette Scores



(c) K=8 Silhouette



(d) K=9 Silhouette

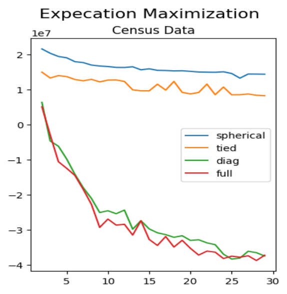
Figure 2: MNIST K Means Review

Figure 2 shows the Kmeans clustering review for the MNIST data. Figure 2a shows the elbow plot, which indicates that 9 is the optimal selection. This chart is much smoother than the Census elbow chart, making it more straight forward to interpret. Figure 2b shows the silhouette scores, where K=8 shows the highest score. Figures 2c and 2c show the individual silhouette charts for K=8 and 9 respectively. These are much cleaner silhouette charts than the Census data and more interpretable because of the lower cluster sizes. The elbow chart at K=8 is pretty close to K=9, but is much better for the silhouette score, so I would choose K= 8 as the optimal cluster size.

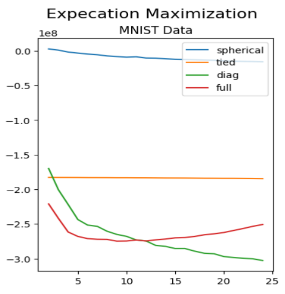
Overall, the MNIST data is more amendable to clustering as it is equally scaled to begin with with the range of 0-255 and its features are also naturally laid out in 2d space.

Expectation Maximization

Expectation Maximization (EM) creates a probabilistic model for each point belonging to a cluster. As in K Means, a key decision is what is the optimal number of clusters. I chose to use Bayesian Information Criterion (BIC). This criterion balances finding the maximum likelihood model given the data with the desire to find the smallest number of clusters. The sklearn EM model supports different covariance models (spherical, diagonal, tied or full covariance), all of which were tried at different cluster counts.



(a) Census



(b) MNIST

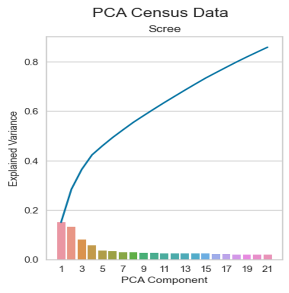
Figure 3: Expectation Maximization Review

Figure 3 shows the EM review. Figure 3a shows the EM review for the Census data. Based on this, Full Covariance with a cluster count of 28 would be chosen. Figure 3b shows the MNIST data, where diagonal covariance at a cluster count of 24 would be chosen. Both of these optimal cluster counts are much higher than the optimal K found for K Means.

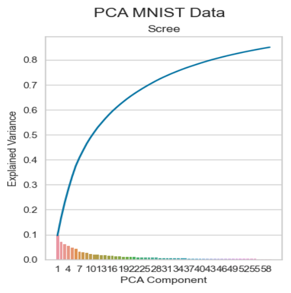
Dimensionality Reduction

Principal Component Analysis (PCA)

Principal Component Analysis maximizes variance along constructed dimensions through successive iterations, with each iteration explaining smaller amounts of variance. Using this model, new constructed features can capture variance to an arbitrary threshold, potentially representing the original dataset with fewer features.



(a) Census



(b) MNIST

Figure 4: PCA Review: PCA analysis was run to meet a threshold of 85% of variance explained. The Census data only reduced the original feature count by about half, while the MNIST data was able to reduce the features count by over 90%.

Figure 4 shows the PCA review, with the goal of explaining 85% of the original variance. Figure 4a shows the PCA review for the Census data. This needed 21 new features, which reduces the number of features by about half. Figure 4b shows the MNIST PCA review, which needed 58 new features. This is a very significant reduction from 784, reducing the feature count by almost 94%. I think this much greater reduction is a function of the necessary 2d relationships in digit images. Each digit will have strong likelihood to have pixels next to each other on or off, meaning they are ripe for a more compact representation. In the case of the Census data, the relationship between features is much less structured, therefore requiring proportionally more information from the original features. Even the 50% reduction of features is very significant, given the exponential nature of the state space and date requirements with increasing feature count.

Scaling the MNIST data seems to make PCA perform much worse than if I just divide the raw data by 255 to set values between 0 and 1. In looking into this, I see many examples of using PCA to perform image compression, where the data is not scaled or normalized. This makes sense to me because the columns features are already scaled to each other for image data and performing the column wise rescaling throws them out of alignment.

Independent Component Analysis (ICA)

sklearn FastICA used defaults

References

1 The MNIST Database of Handwritten Digits. url: http://yann.lecun.com/exdb/mnist/.

2 Census Income (KDD). url: https://archive-beta.ics.uci.edu/ml/datasets/census+income+kdd.

3 Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

4 Bengfort et al., (2019). Yellowbrick: Visualizing the Scikit-Learn Model Selection Process. Journal of Open Source Software, 4(35), 1075, https://doi.org/10.21105/joss.01075.

4 Menninger, A. (2022) Code created for this experiment https://github.com/BigTMiami/ML_Assign_2 Created: 2022.