Assignment 3: Unsupervised learning

Rohan D. Kekatpure

References and links

1. MNIST handwritten digit database.
   http://yann.lecun.com/exdb/mnist/
2. Ionosphere data at UCI.
   https://archive.ics.uci.edu/ml/machine-learning-databases/ionosphere/ionosphere.names

---

1. Description of the datasets

This assignment asks us to apply five supervised learning algorithms on two datasets. The main goal of this exercise to understand the trade-offs in using these algorithms for real-world data mining and predictive modeling tasks. To accomplish this, one needs to identify non-trivial datasets rich in features and size to allow such comparative study. I have chosen the MNIST [1] and the Ionosphere observations [2] datasets for my study.

1.1. What makes these datasets interesting?

The MNIST is a classic image recognition benchmark dataset consisting of 60000 labeled images of handwritten digits in the training set and 10000 in the testing set. Ionosphere observation dataset is 350 observations of 34 attributes with an objective to classify them into ‘good’ ‘bad’ observations — classic binary classification problem. I believe the following factors qualify these datasets as ‘interesting’:

MNIST

1. It has been used to benchmark a number of learning algorithms in image recognition, allowing comparisions to published studies.
2. Each training example is a 28×28 pixels image giving a dimensionality of 784. At this scale, dimensionality is expected to affect model performance making this dataset especially relevant for Assignment 3.
3. By using this dataset I’m taking the risk of not being able to perform training on the full dataset. Using dimensionality reduction helps us train on the full dataset, albeit with a reduced feature set. This once again highlights the utility of and need for dimensionality reduction.

Ionosphere

For our studies, we include 150 observations in our training set and use the remaining 201 observations for testing. We don’t know the train/test split use by the published studies, which might affect how closely we’re able to reproduce these results.

1. Ionosphere is a binary classification problem with 34 attributes. Its binary nature allows an easier comparative analysis of various binary classification algorithms.
2. A feature-set of size 34 should highlight some interesting aspects of dimensionality reduction. It is not as large as MNIST but not trivially small either.

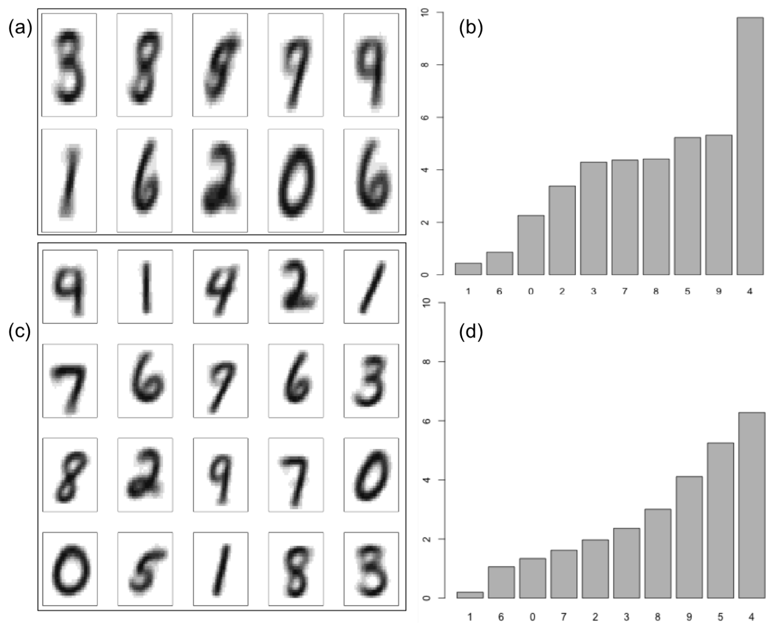

Fig. 1. Cluster centers and per-digit training error for MNIST data clustered using Kmeans clustering with $k = 10$ (a, c) and $k = 20$ (b, d)

2. Kmeans on MNIST dataset

Using the `stats` package in R, we performed K-means clustering on the MNIST dataset with $k = 10$ and $k = 20$. In a sense, the number of classes here is pre-decided by our domain knowledge. That is, we already know that we're classifying 10 handwritten digits. The question is how closely the cluster centers line up with identifiable digits. However, if our metric is training or test error, then we need to relax the $k = 10$ constraint and study the error evolution as a function of k. Instead of the full evolution, here we perform clustering for only two values of k.

For $k = 10$ and $k = 20$, the image representation of cluster centers is shown in Figure 1. For $k = 10$, as shown in Fig 1(a), the centers largely seem to align with identifiable digits. However, because of the fixed number of classes, some digits seem to have merged (such as '4', '7' and '9'). Increasing k to 20 seems to separate out the digits as shown in Fig 1(c).

2.1. *How effective is Kmeans for classification*

This is fundamentally an image classification problem. So we also know the labels already. To study to effectiveness of Kmeans for classification task (a *supervised* learning problem) we computed training error by hand-assigning the classes to the returned clusters (`c = [3,8,5,7,9,1,6,2,0,6]` for $k = 10$ and `c = [9,1,4,2,1,7,6,7,6,3,8,2,9,7,0,0,5,1,8,3]` for $k = 20$). Notice the inherent subjectivity in the label assignment. The error are computed per digit for both $k = 10, 20$. The results are shown in Fig 1(b) and (d). For $k = 10$, the maximum error is in identification of the digit '4' since there is no identifiable cluster corresponding to it. For $k = 20$ the maximum error is for '8' and '3' since their corresponding clusters are intermixed. Overall, Kmeans is an effective technique to visualize the clusters in the MNIST dataset, but is not sufficient by itself for the classification task.

Training set size	# of clusters	Avg classification error
10000	10	40.36%
10000	20	27.20%

Table 1. Average training error in detection of digits in the MNIST database using KMeans algorithm.

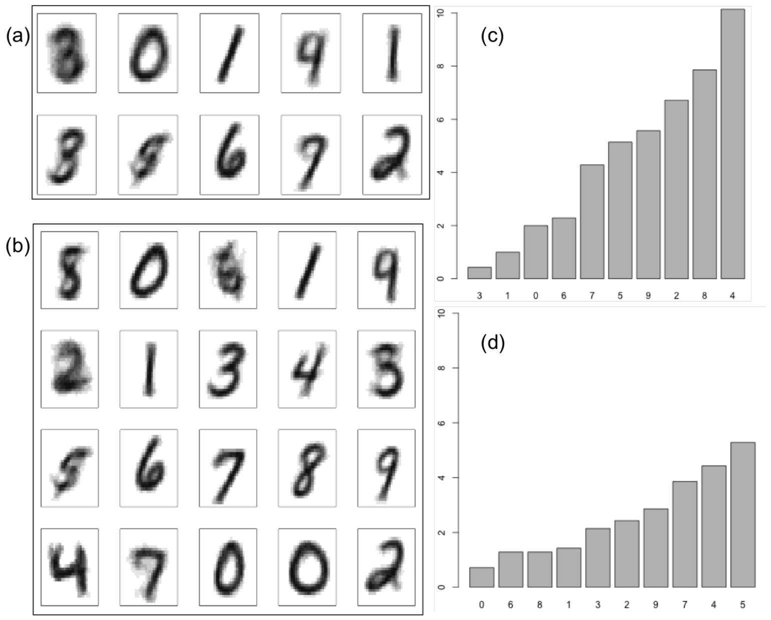

Fig. 2. Cluster centers and per-digit training error for MNIST data clustered using EM clustering with $k = 10$ (a, c) and $k = 20$ (b, d)

Training set size	# of clusters	Avg classification error
1000	10	41.29%
1000	20	29.12%

Table 2. Average training error in detection of digits in the MNIST database using the EM algorithm.

3. Expectation maximization on MNIST

We repeated the same experiment as above, but with Expectation Maximization (EM) as our clustering algorithm. The results are shown in Fig 2. The general observations for EM algorithm as well as its treatment for classification are similar to Kmeans. One difference is that EM algorithms requires more training time due to more involved Bayesian inversion as opposed to simply calculating means in Kmeans. As a result, the EM algorithm was run on a sample size that was 1/10th the size for KMeans. Table

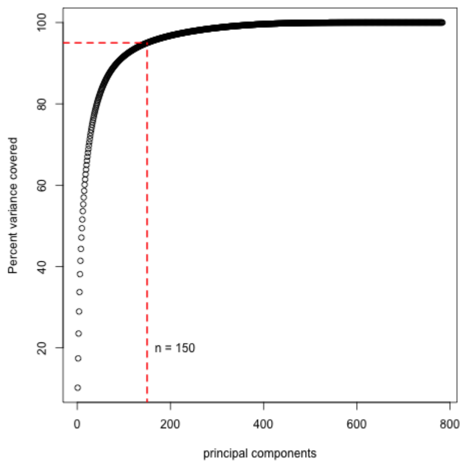

Fig. 3. Cumulative variance covered by the successive PCA components for the MNIST dataset.

4. PCA on MNIST

Next we applied Principal Components Analysis (PCA) on the MNIST training set. The dimension of the original feature space for the MNIST dataset is 784. PCA essentially rotates this input space and ranks the new co-ordinate directions by the amount of variance present along each. We can then choose to retain the number of principal components required to achieve a certain test error.

We ran the PCA on 10000 rows of the MNIST training set. Figure 3 plots the cumulative variance captured by addition of each individual principal component. As shown by the red line, 150 components are required to cover 95% variance. The number 95% is arbitrary and a rough guideline only; it does not mandate that our new training set must contain 150 components. The ultimate test is the training and the test error we get by applying a learning algorithm on the reduced dimensionality set. For a further insight into PCA, we plotted 20 Singluar vectors (1–15, 20, 30, 40, 50 and 100) of the MNIST database in Fig 4(a) and a reconstruction of the original data using the first 25 principal components in Fig 4b. As expected from the PCA theory the successive principal components become more pixelated, signifying more positive-to-negative crossings and thereby more variance. For sufficiently high orders, the principals components are hard to distinguish from noise and justifying their exclusion from analysis.

Fig 4(b) shows that as few as 25 components are sufficient for reconstructing identifiable digits. Later we'll be presenting neural network test error as a function of number of Principal components.

5. ICA on MNIST

We ran ICA on the MNIST dataset using the `ICA` package in R. The total kurtosis for the first 100 components is plotted in Fig 5. As expected the algorithm has found highly kurtotic components under the assumption of independence.

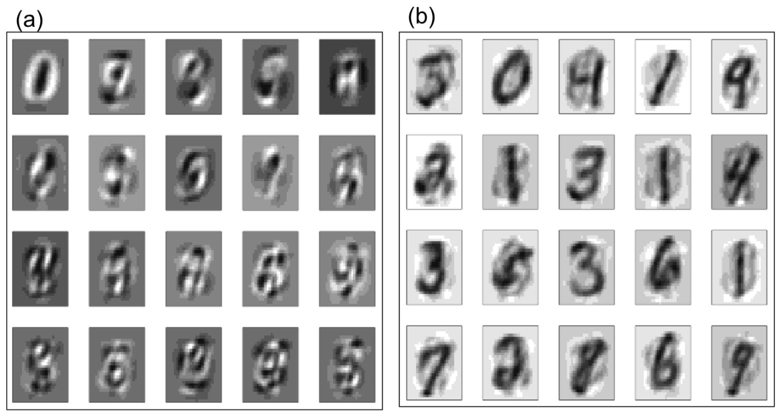

Fig. 4. (a) Selected singular vectors of the MNIST database. Reading from left to right the plotted principal components are 1–15, 20, 30, 40, 50 and 100. (b) Reconstruction of the original data using the first 25 principal components.

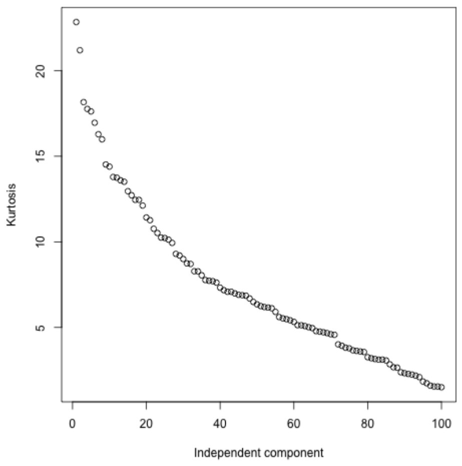

Fig. 5. Kurtosis of the first 100 ICA components for the MNIST dataset

cluster before DR	cluster after DR	similarity
1	3	99.28
2	7	98.43
3	6	81.29
4	2	86.21
5	10	98.05
6	4	92.19
7	9	89.06
8	1	98.77
9	8	87.97
10	5	82.06

Table 3. Cluster similarity with PCA + KMeans with 10 PCA components.

cluster before DR	cluster after DR	similarity
1	6	92.10
2	2	100.00
3	3	100.00
4	9	100.00
5	5	97.32
6	8	95.00
7	7	100.00
8	1	100.00
9	4	92.29
10	10	100.00

Table 4. Cluster similarity with PCA + EM with 10 PCA components.

6. PCA + KMeans

We ran KMeans clustering on the data after retaining the first 10 PCA components. Since the clusters cant be visualized as digits after dimensionality reduction, the question of how similar the clusters are before and after dimensionality reduction needs to be considered carefully. To measure similarity, we use the Jaccard index. That is, we ask, ”how many observations have remained in the same cluster even after dimensionality reduction?” We compute this per digit. Our results are summarized in the table below. Most training examples stayed with their original clusters after dimensionality reduction. Some examples (for example original cluster # 3) shifted more. If we look at the original cluster # 3 in Fig 1, the membership of that cluster is hard to ascertain.

7. PCA + Expectation maximization

Dimensionality reduction does not appear to affect cluster membership calculated by the EM algorithm as much as it does KMeans. The maximum-likelihood-based approach potentially makes the algorithm more robust compared to KMeans (this is just an hypothesis that’ll need to be more carefully investigated with more experiments).

cluster before DR	cluster after DR	similarity
1	6	85.238095
2	8	93.457944
3	4	97.826087
4	3	87.500000
5	9	96.491228
6	5	97.979798
7	6	91.794872
8	3	93.085106
9	5	82.905983
10	2	98.924731

Table 5. Cluster similarity with ICA + KMeans with 10 ICA components.

cluster before DR	cluster after DR	similarity
1	9	79.234973
2	2	99.047619
3	4	97.727273
4	5	96.000000
5	7	100.000000
6	9	70.454545
7	10	76.923077
8	8	93.333333
9	9	97.222222
10	6	98.437500

Table 6. Cluster similarity with ICA + EM with 10 ICA components.

8. ICA + KMeans

Repeating the same experiment, and the same similarity definition as before for ICA + KMeans we obtain the following table of before/after cluster memberships. Note that original clusters 1 and 7 in this case are mapped to cluster 6 in the post DR set. But the overall cluster membership has been retained at a fairly high percentage.

9. ICA + EM

EM with ICA does not display the same robustness in before/after cluster membership as it did with PCA. One possible cause of this may be that the reduced dimensionality representation of the MNIST dataset using PCA is more accurate than ICA.

10. PCA + Neuralnet

We ran the NN from assignment 1 (1 hidden layer, 50 hidden units) by feeding, respectively, 95, 43, 19, and 9 principal components. Also, for this and all the following neural network experiments, the training set size was 500 and the test set size was 10000 samples.

For the PCA + neural net, the variation of the training time, training error and the test error is shown in Table 7 The data in Table 7 suggest that running a model with only 20 components gives the same performance on the test set in a substantially lesser time. Moreover,

# Components	training error %	test error%	run time (sec)
95	0.0	11.34	100
43	0.0	10.53	22
19	0.0	11.88	8
9	0.0	15.18	3

Table 7. Performance of neural net + PCA

# Components	training error %	test error%	run time (sec)
10	0.200000	33.390000	2.772000
20	0.000000	23.740000	6.389000
30	0.000000	24.400000	11.262000
40	0.000000	20.180000	17.538000
50	0.000000	23.790000	25.594000
60	0.000000	25.310000	34.456000
70	0.000000	22.510000	47.097000
80	0.000000	22.170000	61.531000

Table 8. Performance of neural net + ICA

pre-smoothening the dataset with PCA also reduces overfitting.

11. ICA + Neuralnet

As with PCA, we looped through the number of independent components for ICA and computed the training and the test set error of the neural network. Our results are reported in Table 8.

The performance of Neural net with ICA is considerably worse compared to PCA (10% vs 20% test error). ICA's central assumption is that the total signal is a linear sum of statistically independent sources. However, for MNIST data, the statistical independence of consecutive (or even far removed) pixels can be questioned. It can be argued that in text images, there are likely to be continuous streaks of similarly valued pixels.

Overall, the worse performance of Neuralnet + ICA may be stemming from ICA not being the right dimensionality reduction technique for digitized handwritten text data.

12. KMeans + Neuralnet

After obtaining the clusters using KMeans, one needs to define a way to extract the reduced feature set from the clusters. There are multiple options. We can calculate distance to the cluster centers, or the dot product with the unit vectors in the directions of the clusters. However, ignoring the magnitude of the sample and depending only on the direction seems less optimal. So we calculate our reduced feature set from KMeans using 'distance to cluster centers' method. We varied the number of clusters and computed the training and the test set error. The results are reported in Table 9 The results are reasonable considering the simplicity and computational cost of the KMeans relative to PCA. However, there seems to be no straightforward way to improve the performance of KMeans as a dimensionality reduction technique. It is preferable to have a trade-off between computational cost (run time) and accuracy and KMeans does not seem to offer that.

# Clusters	training error %	test error%	run time (sec)
10	1.400000	25.140000	3.103000
30	0.000000	18.140000	12.000000
50	0.000000	15.940000	26.353000
70	0.000000	16.370000	47.841000
90	0.000000	15.470000	79.926000

Table 9. Performance of neural net + KMeans