# Handwritten Digit Recognition via Unsupervised Learning

Xinyi Jiang, Huy Pham, Qingyang Xu

December 11, 2015

## Abstract

We present the results of several unsupervised algorithms tested on the MNIST database as well as techniques we used to improve the classification accuracy. We find that spiking neural network outperforms k-means clustering and reaches the same level as the supervised SVM. We then discuss several inherent issues of unsupervised methods for the handwritten digit classfication problem and propose several methods to further improve the accuracy.

# I. Introduction

Handwritten digit recognition is a benchmark test for computer vision algorithms and has a wide range of applications from bank check processing to postcode recognition. Currently the supervised learning methods yield much higher classification accuracy than most unsupervised ones.

Despite its high accuracy, supervised learning methods tend to be very expensive to develop since they require large training sets to be manually labeled. On the other hand, unsupervised approaches have the potential of being much more efficient to train and can be generalized to more complicated classification problems such as handwritten alphabet or Chinese characters with low additional cost.

In this project, we aim to implement unsupervised algorithms and apply novel techniques to improve the accuracy. We use the MNIST database[1], which consists of 60,000 images in training set and 10,000 in test set. Each image is a $28 \times 28$ array of pixels in gray scale ranging from 0 to 255 and labeled with the digit it represents.

Our first approach is the k-means clustering algorithm, divided into three stages.

We first preprocess the data in order to extract the small subset of most relevant features from the high dimensional input so the program runs more effciently. We use feature selection techniques, such as principal component analysis and sparse autoencoder, as well as better heuristics such as input discretization and image centralization.

Then we improve the initialization process of k-means, either by the probabilistic k-means++ initialization or by choosing the centers which minimize certain cost function we construct.

Finally, we run the clustering process with certain modifications such as k-medians and k-medoids algorithms and gauge their effects.

With the improved feature set and initialization, our k-means algorithm achieves 60% classification accuracy on average, 10% higher than the original.

Spiking neural network is the best performing unsupervised method for handwritten digit recognition up-to-date. We reproduced the 89% accuracy of Diehl and Cook (2015) with the Brian simulator of SNN.

For reference of accuracy, we also run supervised algorithms such as SVM and neural network. SVM yields 90% high accuracy, and it also reveals some inherent issues of handwritten digit recognition which might explain the relatively low accuracy of unsupervised approaches. Neural network has a surprisingly low accuracy around 20%, and we find that the fitting parameters tend to oscillate without converging.

# II. Preprocessing Data

## 2.1 General Methods

We first reshape each image from a $28 \times 28$ array of pixels into a vector of the same size. The default metric we used to cluster and classify the images is the Euclidean distance between them. Since k-means only assigns the inputs to clusters to minimize the total distance from each training example to its corresponding cluster centroid, there is no way to directly calculate the accuracy. Instead, in any given cluster $S$, we choose the digit $d$ which appears the most frequently in $S$ and calculate the ratio of all $d$ present

in $S$ and take this as the classification accuracy of $S$. The total accuracy is obtained by averaging over accuracies of the ten clusters.

## 2.2 Feature Selection

The high input dimensions, if weighted equally, not only causes the program to run extremely slowly but also obscures the essential underlying structure of each digit, which is the most crucial in the classification process and exactly what our program tries to learn. Hence we first need to reduce the dimension of the input and replace it with a much smaller subset of most relevant features.

Towards this end, we implement two algorithms: principal component analysis and sparse autoencoder. To compare their accuracies, we train both models under the same initial condition of 2000 inputs and with the same number of reduced dimensions.

### 2.2.1 Principal Component Analysis

Principal component analysis selects, via the covariance matrix, the $k-$dimensional linear subspace spanned by the $28 \times 28$ dimensional inputs with the largest correlation to each other. In practice, for 2000 inputs, the accuracy does not vary much when we lower the input dimension to 100, and we find that the optimal number of reduced dimensions is 80 (Fig.1).

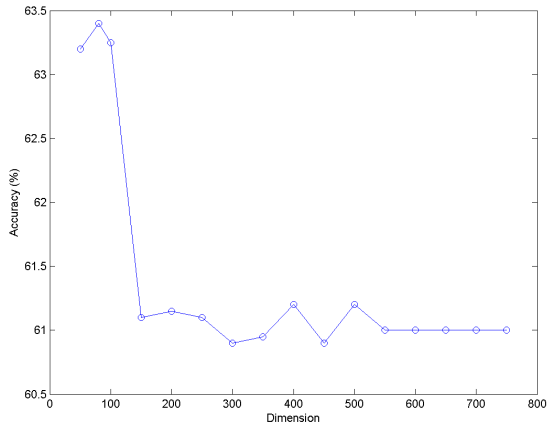


Figure 1: Reducing dimension with PCA increases the classification accuracy

### 2.2.2 Sparse Autoencoder

The sparse autoencoder is a variation of neural network that tries to reconstruct the input through a small number of hidden neurons, which encode the most relevant structure of each digit. Each neuron in the input and output layers corresponds to a pixel in the training images, and we test to find the optimal number of hidden neurons that leads to lowest training errors.

We use the sigmoid function as the activation function for the autoencoder. Hence we need to normalize the input coordinates to between 0 and 1 so the the autoencoder can effectively reconstruct the input. Once the training is complete, we step through the training set again and feed the outputs of the hidden neuron on each training example as the new input to the k-means clustering algorithm.

We find that the optimal hidden dimension for training with 2000 examples is 600, with accuracy as high as PCA. However, the accuracy has a large fluctuation with the hidden dimension (Fig. 2). Also, we find that sparse autoencoder cannot reduce dimension as effectively as PCA since the accuracy drops sharply when the hidden dimension gets below 100.

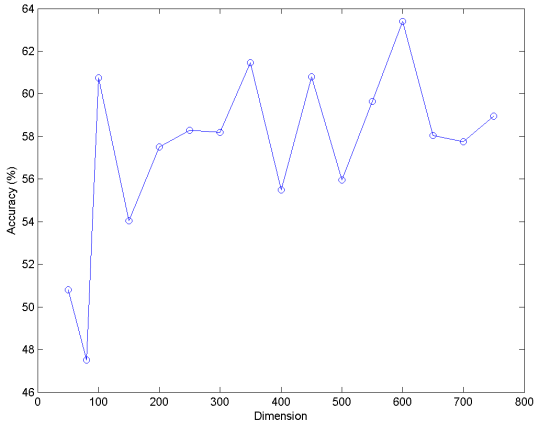


Figure 2: Accuracy fluctuates with varying hidden dimensions for sparse autoencoder

## 2.3 Heuristics

### 2.3.1 Input Discretization

The intensity of the pixels, which ranges from 0 to 255, may be redundant and create extra noise in classification since the structure of the digits should be largely independent of the intensity. Hence, we need to empirically determine the optimal cutoff in pixel intensity, discretize the pixel values to 0 or 1 and use these binary values as the new input for k-means.

The optimal cutoff value we find is 130 for 2000 training examples. With this threshold, the effect of input discretization varies among differently initial-

ized clustering algorithms (Fig. 3 c.f. Section 3.1). The accuracy of the probablisitc k-mean++ slightly improves and the effect is consistent with larger training sets. However, for deterministically initialized k-means with PCA, the accuracy is lower with discretization since we have eliminated too much information from the input so that the new covariance matrix is less effective in revealing the essential correlation between the input components.

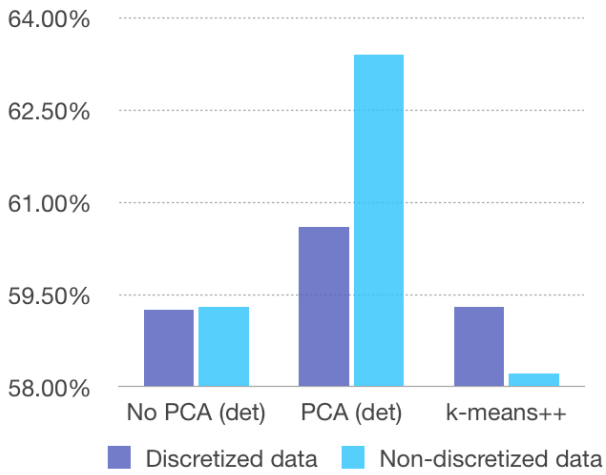


Figure 3: The effect of input discretization varies with determinisitic (left two) and probablistic (right) initialization.

### 2.3.2 Image Centralization

The digit is independent of whether it appears slightly to the left or right of the image, we use the translational invariance to fix the center of all images at the same point and reduce uncorrelated variance.

For each image $I$, we compute its center by the weighted average of the pixel values

$$\left(\left[\frac{\sum_{0\leq m,n<28} I_{mn}\times m}{\sum_{0\leq m,n<28} I_{mn}}\right],\left[\frac{\sum_{0\leq m,n<28} I_{mn}\times n}{\sum_{0\leq m,n<28} I_{mn}}\right]\right)$$

where $[x]$ denotes the integer closest to $x$. We round to the closest integer instead of rounding down since the latter can cause the off-by-1 shift which is detrimental to the classification.

We find that the images in the MNIST dataset are already very well-centered. Computing the weighted average with respect to the original pixel values and the discretized binary values both give the same center $(14, 14)$ for every image in the data set.

# III. K-means Clustering Algorithm

## 3.1 Improved Initialization

The k-means clustering algorithm is sensitive to the intialization of the centroids which, if randomly chosen, often converges to local optimum that does not necessarily yield the best classification.

To improve centroid initialization, we implement the k-means++ heuristic[7], which first randomly chooses a training sample as the center, computes the distance $D(x)$ of every other inputs to its closest center, and then randomly chooses another center from the other inputs with probability weighted by $D(x)^2$. It repeats this process to generate all ten centroids.

We also implement a deterministic heuristic which computes the cost function for each image $I$

$$C(I)=\sum_{i=1}^{n}\frac{d(I,I_i)}{\sum_{k=1}^{n} d(I_i,I_k)}$$

where $d(I, I_i)$ represents the Euclidean distance between our image and the ith training example. We then take the ten images with lowest cost function as the initial centroids.

Both initialization methods show overall improvement on accuracy (Fig. 4), with k-means++ slightly outperforming minimization of the cost function. For comparison, we also try to initialize the centroids with the first ten images of the training set.

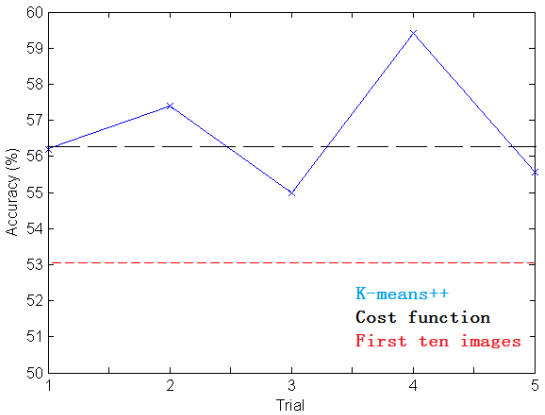


Figure 4: K-means++ overall outperforms cost function minimization with 2000 trianing samples

## 3.2 Execution

In order to get a good estimate of average-case accuarcy and minimize the effect of anomalies in random initialization, we need to run the k-means clustering for multiple trials on the same training set to gauge the overall effect.

Each round of clustering terminates when the cost function, defined as the sum of Euclidean distances squared of all images to their assigned centroids, converges within $10^{-2}$. After each convergence, we choose at random a training example from each cluster as the new centroids and repeat the clustering to see if it further minimizes the cost function.

The program terminates when choosing new centroids either has not improved the cost function for $k_{fix}$ consecutive trials or when it has improved for $k_{imp}$ trials in total. For the results below (Fig. 5), we set $k_{imp} = k_{fix} = 3$ and report the averaged accuracy.

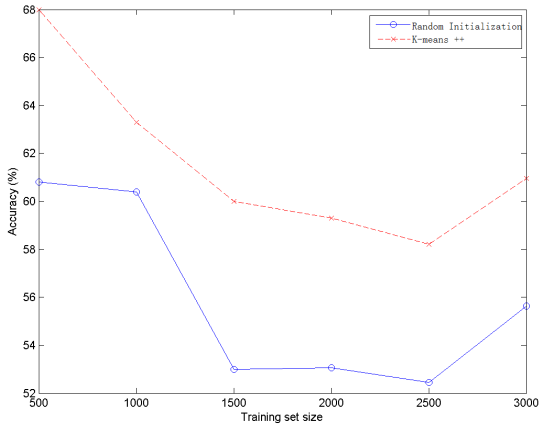


Figure 5: Accuracy vs. training set size for regular k-means and k-means++

Rerunning the clustering in the above process in general yields lower values of the cost function. However, in practice this does not guarantee a higher classification accuracy. Therefore, even with the reduced input from feature selection, the default metric of Euclidean distance still has limited effect in capturing the structural similarity between digits.

# IV. Spiking Neural Network

The spiking neural network (SNN) is a biologically inspired model in which neurons evolve in time and communicate with each other by sending spikes through synapses. We train the model by updating the weight associated with each synapse depending on how frequently spikes travel on it. So far SNN has the best performance on handwritten digit recognition and, aided by the Brian simulation package and their code available online, we are able to reproduce the 89% accuracy by Diehl and Cook[3].

One issue with the current implementation of SNN is that after training, we need to label each neuron with the digit it represents by showing the SNN images of all digits and see how each neuron responds. In this sense, SNN still requires labels of the images and therefore is not completely unsupervised.

# V. Supervised Learning Algorithms

## 5.1 Support Vector Machine

We use the liblinear library[8] to run SVM and find that it yields high accuracy which increases with training set size. But certain digits such as 3 and 8 have consistently higher error rate, which means that their structures are inherently harder to classify.

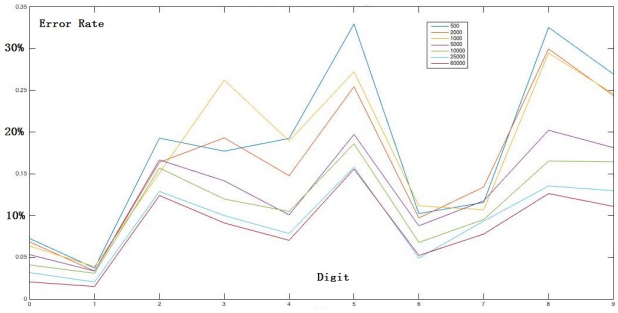


Figure 6: Error rate of each digit showing certain digits consistently worse than others

## 5.2 Neural Network

Our neural network has input and output layers of the same dimension as the training images. We use sigmoid function as activation, train the model by varying the number of hidden neurons and find that it has overall low accuracy around 20%. Close investigation shows that the fitting parameters often oscillate without converging during each backward propagation update. In a few cases when they do converge,

however, these parameters yield very high error. The problem persists when we instead use softmax function to compute activation.

# VI. Discussion and Future Work

The overall performance of each algorithm we implement is summarized in Fig. 7 below:

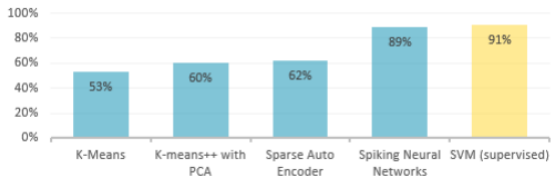


Figure 7: Overall accuracy of each algorithm

We improve k-means clustering with PCA and sparse autoencoder to reduce the dimension of the input, as well as k-mean++ and cost function to optimize initialization. Overall, the improved versions of k-means achieve 60% accuracy when we train with 2000 inputs. K-means++ and PCA run fairly quickly while sparse autoencoder takes longer to train.

One inherent issue with handwritten digit recognition is that when we look at the distribution of digits inside each cluster after running k-means, we usually find that digits $0, 1, 2, 6$ tend to be much better classified into their respective clusters than $3, 5$, and $8$. The error rate of each digit in SVM (Fig. 6) confirms that the structures of $3, 5$, and $8$ are inherently harder to classify even in supervised setting.

We believe the underlying cause of this problem is that the various optimizations we implement to k-means mainly improve the clustering efficiency. However, as discussed in Section 3.2, higher clustering efficiency does not guarantee higher classification accuracy if the algorithm cannot correctly tell one digit from another. Therefore, future work should focus on improving the effectiveness of our algorithm to recognize the structural distinction of each digit

One possible solution, inspired by the generative learning models, is to first construct prior models to encode the most essential structure of each digit and then perform clustering with these prior models as either centroids or reference to how we select centroids.

Another remedy to the same problem is to improve the cost function which is minimized by each iteration of clusteirng. The cost function should essentially quantify the structural similarity between images which may correctly suggest whether they represent the same digit. The current Euclidean metric does not take into account the position of the pixels, and we tried another metric which scales the pixel values according to the distance from the center, but it yields even lower accuracy. Still, we should try to construct better cost functions whose output effectively quantifies similarity.

Finally, as mentioned in Part IV, the current implementation of SNN is not strictly unsupervised. So it still has the disadvantage of requiring large, manually labeled traing sets. Based on the work by Diehl and Cook, we would like to correctly classify the excitatory neurons in SNN without labels. This would make SNN much less expensive and much easier to generalize into other situations.

## VII. Reference


1. MNIST database of handwritten digits: http://yann.lecun.com/exdb/mnist/

2. Andrew Ng. Lecture notes on Neural Network and Sparse Autoencoder for CS294A.

3. Peter Diehl, Matthew Cook. Unsupervised Learning of Digit Recognition Using Spike-Timing-Dependent Plasticity. IEEE TRANSACTIONS IN NEURAL NETWORKS AND LEARNING SYSTEMS.

4. Hae-Sang Park, Chi-Hyuck Jun. A simple and fast algorithm for K-medoids clustering. Expert Systems with Applications 36 (2009).

5. Andrew Ng. CS229 lecture notes.

6. Stanford Unsupervised Feature Learning and Deep Learning Tutorial.

7. David Arthur and Sergei Vassilvitskii. k-means++: The Advantages of Careful Seeding. SODA '07 Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms Pages 1027-1035.

8. Chih-Chung Chang and Chih-Jen Lin. LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/˜cjlin/libsvm.