Application of Linear Algebra Methods in MNIST Handwritten Digit Classification

Yuchen Luo

Introduction

The linear algebra method has been used widely in different areas, such as engineering analysis, high performance computing, chemical/genetic pathways study and computational animation. Matrix manipulation technique is one of the most used methods in different applications and the one we are going to study in this report. MNIST [1], a classic dataset in machine learning especially classification, is employed in this work. It provides us opportunities to try the real-world data without spending too much time on data cleansing and not be limited by the computational power. In this work, Singular Value Decomposition (SVD) and Nonnegative Matrix Factorization (NMF) are used in the classic MNIST classification problem. nMNIST dataset [2] is employed in this study to test the stability of different methods, the definition of stability will be introduced later in report. In addition, two certain architectures Convolutional Neural Network (CNN) is also used in the same problem to compare with SVD and NMF's performance (accuracy and stability) as a case study.

Related Work

This work is triggered by the work of Mazack's [3,4]. Mazack (2009) has used SVD to classify the U.S. Postal Service handwritten digit. He uses a database with 7291 greyscale handwritten digits with resolution of 16 by 16. His approach is followed in this work to do the classification problem in this work, which will be described in detail in the next section. The accuracy reported in [2] is 93.572%. But there is no discussion about the error induced by the classifier, which might be interesting and valuable to peruse. In the same year, [4] was carried out as a continue of the work in [3] on the handwritten digit recognition. NMF is employed to classify the same dataset. In [4], four different algorithms of NMF are proposed using Matlab. The published Matlab code are directly used in my application, including NMF-MU, NMF-ALS without projected gradient and SNMF/L. But the projected gradient with ALS algorithms is not included in the paper. The reported prediction accuracy of using SNMF/L is 92.676% in the paper and a parametric study was done to control the sparsity of the factorized matrices. But no comparison between different algorithms, runtime wise nor accuracy wise, has been done.

In this work, the method to approach the MNIST classification problem is the same as Mazack's work [3], but several extensions are made. The contributions of this work are:

1) Add an error analysis, to explain the mislabel rate of the classifier;
2) Add a different algorithms performance study for NMF method;
3) Introduced the stability study of both SVD and NMF methods;
4) Discussed the accuracy and stability of the popular CNN method.

Dataset and features

Two datasets are used in this study, one is from [1] a classic MNIST database, the center of handwritten digits is moved to the center of the image, the other one is taken from [2] a nMNIST database with gaussian noise with signal-to-noise ratio of 9.5. Both MNST and nMNIST contain image of size 28 pixels by 28 pixels. Parts of the MNIST and nMNIST are shown in Figure 1 below:

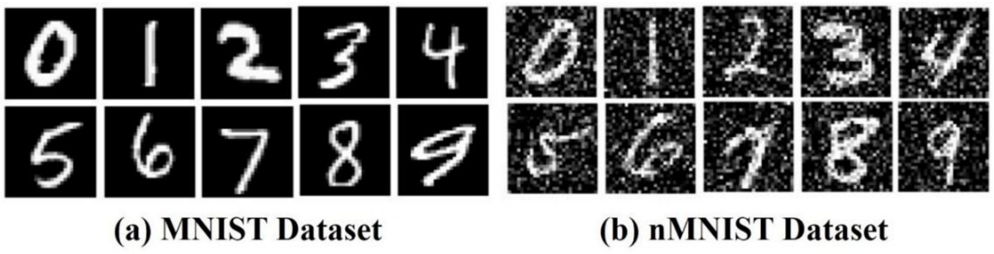


Figure 1 (a) MNIST and (b) nMNIST Dataset

Methods

1) Problem formation

Before SVD and NMF are discussed, the classification problem of the MINIST is introduced first. In the MNIST dataset, there are handwritten digit image from zero to nine with labels indicating which digit the image refers to. The goal of the study is to create a classifier which can recognize the handwritten digits accurately. As general procedures of classification problem in the machine learning area, the classifier is first fed with training dataset and then tested with the test dataset.

The more detailed procedures we carry out are as following: As each raw image represented by a 28 by 28 matrix in which the entities are from 0 to 255 (0 represent the black and 255 represent the white. The value in between is the transition from the black color to the white color.), We reshape this 28 by 28 matrix into a size 784 vector. Thus, each handwritten digit image matrix in the database is converted to a 784-dimension handwritten digit image vector. Then we can assembly handwritten digit image vector to form an assembly matrix by category. Assembly matrix is a matrix containing handwritten digit image with the same label. Each column of the assembly matrix is a sample of the handwritten digit. An example of the assembly matrix for digit '5' is shown in Figure 2, we use D to represent assembly matrix in later context. Similarly, we can obtain the assembly matrices D for all 10 digits based on the data we have in the training set. It is clear that the assembly matrix D for each digit is not necessary a square matrix. It has 784 rows which is the same as the image vector's dimension, and the number of columns is equal to the number of sample with certain types of label in the training set. In Figure 2's case, the number of columns of D_5 equals to the number of digit '5' in the training set.

For the sake of convenience, we use m to denote the number of rows and n to denote the number of column of a rectangular matrix. m << n is the case we have right now, considering in a big dataset like MNIST, the number of samples of each digit is more than 5,000. Let us denote the vector of a test image as z, the process of classification is to find a best linear combination of columns of D_i to represent z, where i is from 0 to 9. The residual is denoted with ρ. If the vector z lives in the column space of D_i, then it is perfectly fine to use the linear combination of D_i's columns to represent that vector, which yields a zero residual, $\rho = 0$. But in the reality, z is not usually in the space spanned by D_i. We can never use the linear combination of D_i's columns to represent z without any residual. As we can see from Figure 3, when z is not within the column space of D_i (in this case, space is represented as a surface), the minimum distance from z to the space is z – D_ix when z – D_ix is orthogonal to the D_i space. Mathematically, we can write the orthogonality as the norm form, which is shown in Eq. 2. In general, to classify a test image, we need to compute the its minimal distance to the column space of all 10 digits' assembly matrix, and classify the test image to the digit which has the smallest distance. The closer of the test image to the digit's assembly matrix, the higher possibilities that the image has the label of that digit. But note that we only pick up the smallest distance to classify the test image, it may be prone to some errors. If the distances to two digits assembly matrix are very close to each other, there might be some uncertainties in the classification. But we will not address that issue in this report, and just employ the rigid cutoff criterial for the classification problem. In general, as shown in Eq.3, we need to find the digit i for which the distance from z to column space of Di is the smallest among all ten digits.

$$\rho_i = min_x \|D_i x - z\| \qquad \text{(Eq. 1)}$$

$$D_i^T(D_i x^* - z) = 0 \qquad \text{(Eq.2)}$$

$$min_i \rho_i,\ \text{i} = 0, 1, 2, \dots, 9 \qquad \text{(Eq. 3)}$$

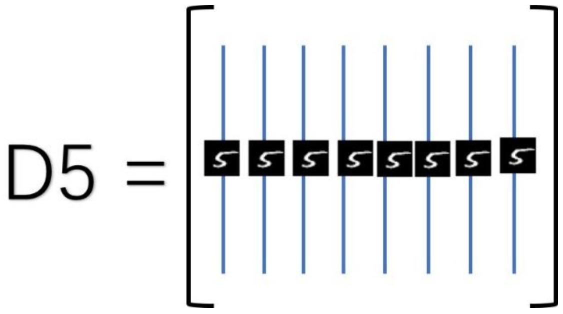


Figure 2 Assembly matrix of digit '5'

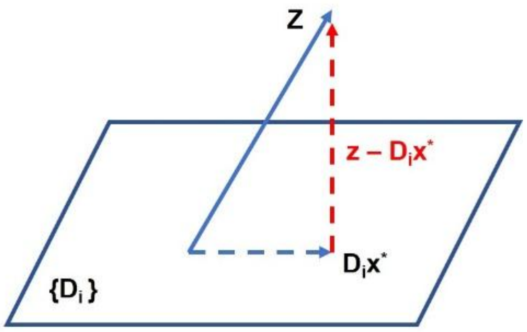


Figure 3. Schematic of the Least Square Problem

The classification problem is equivalent to solve a least square problem: We need to solve a least square problem for each digit and take the smallest value among them. Let us start with Eq. 1. By observation, the assembly matrix Di is not a full rank matrix, we have potential to reduce the number of column vectors in the matrix. Among all dimension reduction algorithms, SVD is the most widely used method and enjoys good accuracy. SVD will be used firstly in this work.

2) SVD (Singular Value Decomposition)

Theorem: For any matrix $A \in R^{m\times n}$, there exist unitary matrix $U \in R^{m\times m}$ and $V \in R^{n\times n}$ such that $A = U\Sigma V^T$ where Σ is a diagonal matrix with entries $\sigma_{ii} \geq 0$. [5] Where σ is the singular value of matrix A.

The SVD can be used in the classification problem by factorizing the digit matrix D_i in Eq. 1. As we know that $Ax = U\Sigma V^T x = Uy$, if we rewrite $\Sigma V^T x = y$. Then the least square problem in Eq. 1 is equivalent in solving a least square problem in Eq. 4

$$\rho_i = min_y \|U_i y - z\|, \text{ i} = 1, 2, 3, \ldots, 9 \qquad \text{(Eq. 4)}$$

Because a unitary matrix preserves the length, $\rho_i = min_y \|U_i y - z\| = min_y \left\|U_i^T (U_i y - z)\right\| = min\left\|y - U_i^T z\right\|$. There is no constrain on y, we can pick any y as needed. It is convenient just to choose y = $U_i^T z$ to make the minimal value of ρ be reached, which is zero. And we can substitute the relation of y and z back to Eq. 4, an explicit equation for calculating the least square distance to digit's assembly matrix is obtained as Eq. 5

$$\rho_i = \left\|U_i U_i^T z - z\right\|, \text{ i} = 1, 2, 3, \ldots, 9. \qquad \text{(Eq. 5)}$$

Note that for a unitary matrix, $U_i U_i^T$ does not necessarily yield an identity matrix if the matrix U_i is not a square matrix, which is the case in our classification study. But $U_i^T U_i = I$ holds for all cases even for a rectangular matrix.

In order to reduce the computational load and not lose too much accuracy, a dimension reduction is used in the SVD classifier. Only singular vectors corresponding to 10 leading singular values are kept to form unitary matrix U_i. 10 is chosen as a hyperparameter which is from [3]. Since we do not intend to focus on the study of choosing hyperparameter, we just take the value reported in the literature. The SVD classifier is implemented in ipython notebook (python 3.6) and the SVD algorithm is obtained by using packages in numpy.linalg.svd.

3) NMF (Nonnegative Matrix Factorization)

Theorem: Let $A \in R^{m\times n}$ be a matrix such at all elements in the matrix are nonnegative (either position number or zero). Then for $k \leq \min\{m, n\}$, there exist $W \in R^{m\times k}$ and $H \in R^{k\times n}$ such that $A \approx WH$, and all elements in the matrix W and H are nonnegative as well [4].

The NMF is attractive for its nonnegative property, which makes the factorization easy to be interpreted. For some applications, take MNIST classification for example, there is no physical meaning for negative values as pixel value should always be nonnegative. NMF can be used in the classification problem in a similar way as the SVD method does as discussed in the previous section. After factorizing the matrix A, $Ax = WHx = Wy$ is achieved. Let $y = Hx$. Eq. 1 now can be rewrite as:

$$\rho_i = min_y||W_i y - z||, \text{ i} = 1, 2, 3, \ldots, 9 \qquad \text{(Eq. 6)}$$

Finding least square distance from z to digit assembly matrix D_i is equivalent to find the least square distance from z to factorized matrix W_i, which has a smaller dimension compared with the original matrix A. The classification goal is to find the smallest least square distance among all 10 digit's reduced matrix W_i, the dimension W_i is a hyperparameter as well as in SVD case and set before the factorization starts. In our study the column dimension of Wi is chosen to be 10, following literature [4].

In the process of performing NMF, there are lots of different algorithms to achieve such goal. There are Multiplicative Updates (MU), Alternating Least Square (ALS) and Sparse Nonnegative Matrix Factorization Right Version (SNMF/R) and Sparse Nonnegative Matrix Factorization Left Version (SNMF/L).The algorithms are introduced in brief here, for more details please refer to [4]. MU method is the most straightforward method. By providing random nonnegative matrices as W^0 and H^0, H and W are updating alternatively for a certain number of iterations, the iteration number can be set as a hyperparameter. It is written as an elementwise calculation format: First $H_{ij} = H_{ij}\frac{(W^TA)_{ij}}{(W^TWH)_{ij}}$,then $W_{ij} = W_{ij}\frac{(AH^T)_{ij}}{(WHH^T)_{ij}}$. Since all the matrices have the nonnegative elements, the updating is nonnegative guaranteed. There is no need to apply any constrain during the updating process. But it faces an issue, the element in the denominator may be zero. In practice, usually a very small value is added to the denominator such as 1e-9 or even smaller. And this method can not guarantee convergence even after it runs out the iteration steps we set before. For the ALS algorithm, it is similar to MU but some modifications have been done in the iteration steps. A least square problem with nonnegative constrain is solved at each iteration. First, solve for $min_H||WH - A||_F^2$, st. $H \geq 0$; after obtaining H, use it to solve $min_W||H^TW^T - A^T||_F^2$, St. $W \geq 0$. This method can avoid the divide by zeros issue, but the computational cost is larger. Projected gradient is used to accelerate the algorithm [6]. Matlab has built-in NMF function which adapted the projected gradient version of ALS, which will be used as the main algorithm to factorize a NMF factorization in this work. Different methods and their computational time and prediction accuracy will be discussed in the result section. Apart from the normal way of solving the NMF factorization problem, we can also obtain a highly sparse matrix by careful designed algorithms. The objective function in the NMF needs to be modified for sparsity generation algorithm by adding the 1-norm of factorized matrix to the objective function. The sparsity of factorized matrices can be adjusted by changing the weight coefficient before the newly added 1-norm term. If we add 1-norm of H to the objective function, it is called the SNMF/R algorithm. If we add 1-norm of W to the objective function, it is called the SNMF/L algorithm. The NMF classifier is implemented in Matlab R2017a.

4) Solve the least square problem

After applying SVD and NMF to reformat the classification problem, we have converted the original classification problem into a least square problem without constrain. There are several classic methods can be used to deal with it, for example Norm Equation, QR decomposition [5]. In this study, the pseudo inverse method is employed to tackle the problem. For SVD case, the least square problem has been solved as in Eq. 5. For NMF, W_i is not necessary an invertible matrix, so we are going to use pseudo inverse. If the case we are studying here, the row dimension is larger than the column dimension, the pseudo inverse can be written as: $W^{+} = (W^{T}W)^{-1}W^{T}$. The solution for Eq.6 is: $y = W^{+}z$. Substitute back y to the equation, an expression of least square distance for the digit assembly matrix D_i after NMF can be obtained.

$$\rho_i = min_y \|W_i W_i^{+} z - z\|, \text{ i} = 1, 2, 3, \ldots, 9 \qquad \text{(Eq. 7)}$$

5) CNN (Convolutional Neural Network)

CNN is used in this study mainly for the comparison purpose. Comparison is not in the general context as CNN's performance is highly related to the architecture and parameters it has. It is not a symmetrical comparison between SVD/NMF and CNN, instead it is more like a case study of CNN in the context of this work: accuracy and stability in MINIST classification. A single-layer and two-layer CNN are employed in the MNIST classification study. CNN is implemented by Keras with Tensorflow running backend. The details setting of the CNN is from [7]. In the result section, the architecture will be show.

6) Accuracy and stability study

Two metrics are introduced to assess the result of the classification problem. The first metrics is the accuracy metric. The accuracy is defined as the ratio of the number of correctly classified test images among all the test images. It is a commonly used metric to assess the quality of the classification problem. Another metric I used is defined as the stability. The definition of the stability is how well a trained classifier performs when the test set is perturbed by the noise or other perturbations, but the training set is noise-free or has not been exposed to the same noise. A common test procedure is carried out to assess the accuracy metrics, by feeding classifier the test set it has never seen and checking the successful rate of prediction. The stability test is performed as following: by feeding the nMNIST dataset as a test set to the classifier which is trained by MNIST dataset and checking the successful rate of the classifier in the test set image label prediction.

Results and Discussion

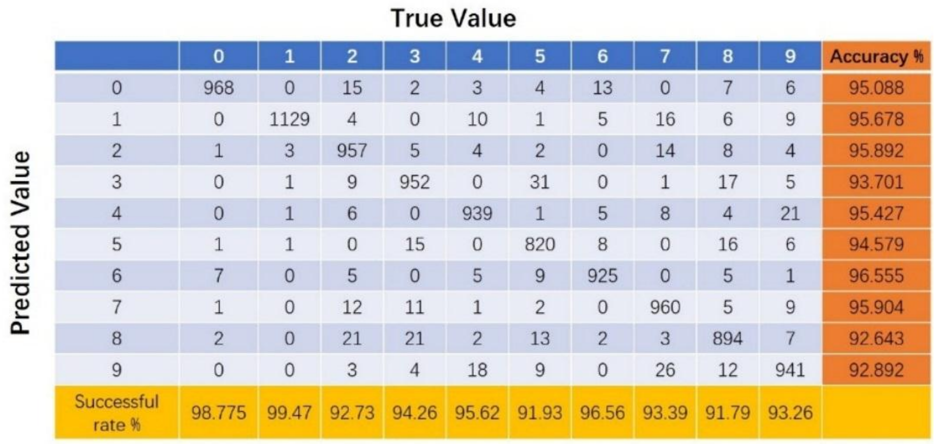
The result section is arranged as following: firstly, the accuracy metrics are used to assess two methods, SVD and NMF. Some efforts are spent to understand the accuracy result. Then the stability result from these two methods are shown, and some comparisons are done. Finally, the accuracy and stability discussion are expanded to CNN as a case study.

1. Accuracy Metrics Study

i) SVD

As shown in Table 1, a predicted result validation matrix (the transpose of usually used confusion matrix) is formed based on the result of SVD. The row is showing the true value composition of each predicted value. Let us take the first row as an example. The predicted digit '0' is made of true value '0' (total 968), true value '2' (total 15), true value '3' (total 2), true value '4' (total 3), true value '5' (total 4), true value '6' (total 13), true value '8' (total 7) and true value '9' (total 6). 95%

of the predicted digit ‘0’ are the real zeros, the others are the false zeros. Similarly, we can see the accuracy of different predicted digits, the accuracies are all above 92%, and predicted digit ‘6’ has 96.55% probability to hit the true digit ‘6’. The column is showing the predicted value composition of each true value. Let us take the first column as an example. The first column is made of predicted ‘0’ (total 968), predicted ‘2’ (total 1), predicted ‘5’ (total 1), predicted ‘6’ (total 7), predicted ‘7’ (total 1) and predicted ‘8’ (total 2). It shows that for the true value ‘0’, 98.775% chance the classifier recognizes it as zero and labels it as predicted ‘0’. The chances of mislabel the digit 0 is quite small. Similarly, we can get the information of other true digits. Among all the digits, digits 1 is has the highest chance to be successfully classified, while true digit 8 is the most frequently mislabel digit by the classifier. The overall accuracy of SVD prediction is 94.85%.



		True Value										
		0	1	2	3	4	5	6	7	8	9	Accuracy %
Predicted Value	0	968	0	15	2	3	4	13	0	7	6	95.088
	1	0	1129	4	0	10	1	5	16	6	9	95.678
	2	1	3	957	5	4	2	0	14	8	4	95.892
	3	0	1	9	952	0	31	0	1	17	5	93.701
	4	0	1	6	0	939	1	5	8	4	21	95.427
	5	1	1	0	15	0	820	8	0	16	6	94.579
	6	7	0	5	0	5	9	925	0	5	1	96.555
	7	1	0	12	11	1	2	0	960	5	9	95.904
	8	2	0	21	21	2	13	2	3	894	7	92.643
	9	0	0	3	4	18	9	0	26	12	941	92.892
	Successful rate %	98.775	99.47	92.73	94.26	95.62	91.93	96.56	93.39	91.79	93.26	

Table 1. Accuracy result of SVD methods

Another interesting observation we obtain from the Table 1 is that there are some symmetrical features: there are 7 samples of true value ‘0’ are classified as digit ‘6’, true value ‘0’ is most frequently and easily to be classified as ‘6’; there are 13 samples of true value ‘6’ are mislabeled as ‘0’, ‘6’ is also most easily be classified as ‘0’. Therefore, we can say digit ‘0’ and ‘6’ are two most easily misclassified pairs. We can find similar pairs, such as (‘3’, ‘5’), (‘2’,’7’), (‘4’,’9’). It implies that there are some underlying reasons for this symmetrical feature. We show the top 10 of the singular vectors of each digit in the training set in Figure 4 to get some insight about the underlying reason for such feature. The most easily mislabeled pairs are also shown in the Figure 4. For all these mislabeled pairs, we can find they are very similar especially for the higher order singular vectors. To achieve a higher prediction accuracy, we can try to do the method of training in pairs. Among each pair, a ‘tie-breaker’ algorithm is needed to separate the pair, but the algorithm needs to be very sensitive and capable to separate these easily mislabeled pairs. Tangent distance method [3] can be a candidate for this problem.

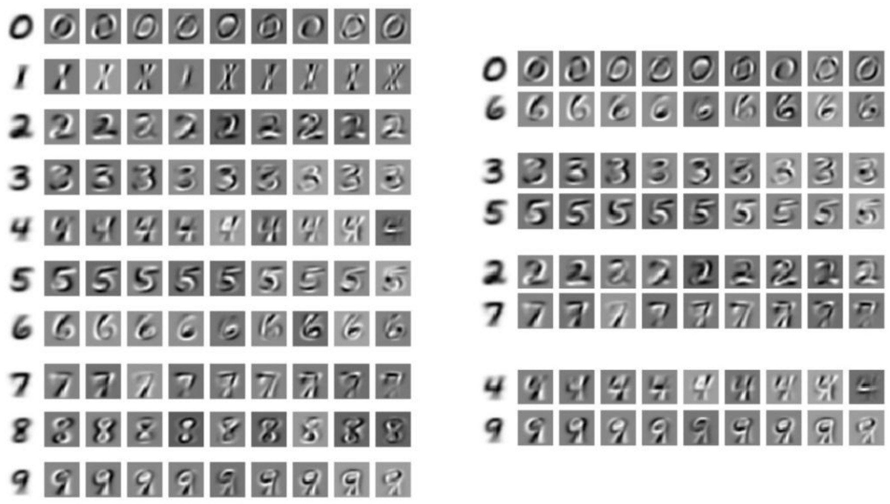
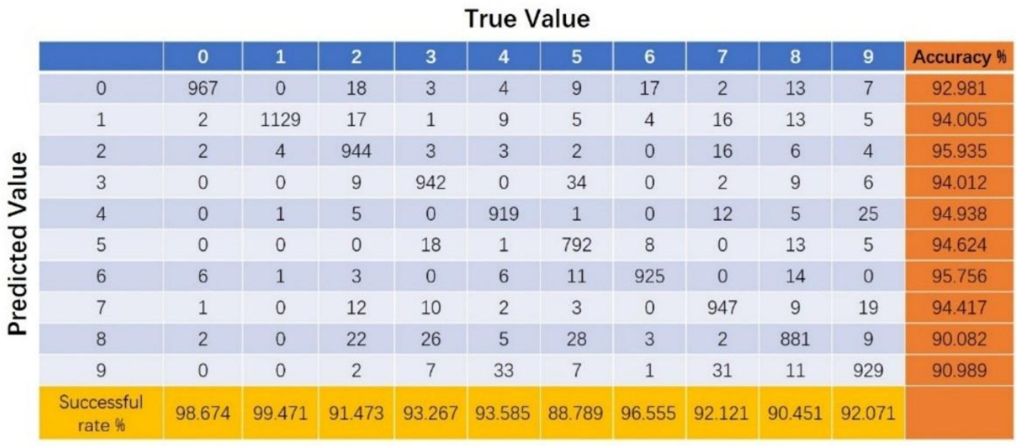


Figure 4. Leading 10 singular vectors for ten digits and frequently mislabeled pairs

ii) NMF

We can present the result of the NMF in the similar structure as shown in the previous SVD section. The structure of the NMF result matrix is the same as the SVD result matrix. The symmetrical structure is also observed in the Table 2. Similar result and conclusion can be draw as well, so I will just skip this part of analysis for the sake of concision and focus on new features of NMF. The overall classification accuracy for NMF-ALS is 93.47%, a little bit lower compared with the classification accuracy for SVD. In the Table 3, we can see that the variation of the prediction of the accuracy is very small between different algorithms, but the there is a considerable different in the running time. The run-time of MU is approximately the same as ALS in 100-time-iteration case. But SNMF/L requires one order of magnitude higher run time because the objective function has been modified. Since the accuracy does not depend on the algorithm used, we will use NMF-ALS as the algorithm in the stability study. Note that in the result, only one run is performed to generate the result. As author has checked, performing multiple runs does not significantly change the value. The conclusion draw from the data still holds.



Predicted Value	True Value										
	0	1	2	3	4	5	6	7	8	9	Accuracy %
0	967	0	18	3	4	9	17	2	13	7	92.981
1	2	1129	17	1	9	5	4	16	13	5	94.005
2	2	4	944	3	3	2	0	16	6	4	95.935
3	0	0	9	942	0	34	0	2	9	6	94.012
4	0	1	5	0	919	1	0	12	5	25	94.938
5	0	0	0	18	1	792	8	0	13	5	94.624
6	6	1	3	0	6	11	925	0	14	0	95.756
7	1	0	12	10	2	3	0	947	9	19	94.417
8	2	0	22	26	5	28	3	2	881	9	90.082
9	0	0	2	7	33	7	1	31	11	929	90.989
Successful rate %	98.674	99.471	91.473	93.267	93.585	88.789	96.555	92.121	90.451	92.071	

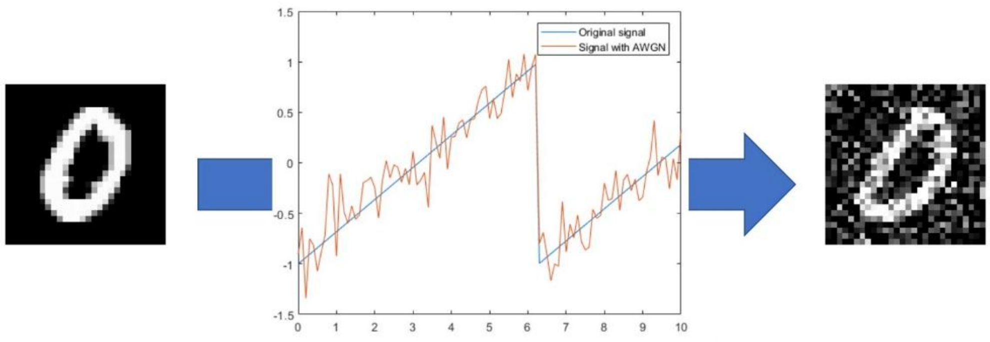
Table 2. Accuracy result of NMF methods

	Run Time (s)	Accuracy (%)
Multiplicative Update	115	93.71
ALS + Projected Gradient	144	93.47
SNMF/L	5400	93.17

Table 3 Run time and accuracy of different NMF algorithms (Iteration 100 times)

2. Stability Metrics Study

The gaussian noise is added to perturb the image from MNIST to produce nMINIST as shown in Figure 5. The training set are obtained from the noise-free MNIST dataset, but the test set are fetched from the nosy nMNIST dataset. In the ideal case, a noise-free test set will be used to test the classifier. But in the reality, the application can be in an environment with noise. For example, when there is an electromagnetic interference disturbing the camera or when some pixel in the camera just died. The camera condition while taking the testing image can be different than the camera condition while taking the training image, we just use the noise interfered image to mimic this realistic situation. In Table 4, we can see that after adding the noise to the test set, the accuracy of the SVD classifier and NMF-ALS classifier is not largely disturbed. For SVD the prediction accuracy degraded by 1.5%, and for NMF-ALS, the accuracy degraded by 0.5%. Because both SVD and NMF are dimension reduction methods, which can filter the noise out without being disturbed too much.



https://www.mathworks.com/help/comm/ref/awgn.html

Figure 5. Gaussian noise added to perturb the testing set.

	Noise Free Test Set	Noise Added Test Set
SVD	**94.85%**	**93.45%**
NMF-ALS	**93.47%**	**92.94%**

Table 4 Stability study of SVD and NMF-ALS algorithms

3. Case Study of CNN

Based on our study in the previous section, we know that SVD and NMF can achieve relatively good prediction accuracy and stay stable under the disturbance of noise. What about the most popular method CNN nowadays? What is the prediction accuracy of CNN and how will CNN react to this noisy data? These two are the question we want to address in this part of the report. A simple 1-layer and a little bit more complex 2-layer CNN are used in the MNIST classification study and the analysis of the data is done from accuracy and stability viewpoints.

First, a 1-layer CNN is used in the MNIST classification problem, the architecture of CNN is shown in Figure 6. Dropout rate is chosen as 0.2. The CNN network can be denoted as 32C5-2P2-FC128-FC10. The parameters are taken directly from [7], since we are not focusing on parameter tuning here. The overall accuracy of the 1-layer CNN is as high as 99% tested using MNIST dataset, which is much higher compared with the SVD/NMF-ALS method. From the reported value in literature [8], the CNN based classifier can be as high as 99.77%. If the nMINSIT dataset is used as the test set for the CNN, the accuracy drops to 90.15%, which is lower than the SVD and NMF's testing accuracy using nMINIST dataset.

If we make the CNN more complex by adding another layer, the architecture is shown in Figure 7. Dropout rate is chosen as 0.2. The CNN network can be denoted as 30C5-2P2-15C3-2P2-FC128-FC50-FC10. The overall accuracy of the 2-layer CNN is as high as 99.31% tested using MNIST dataset, which is higher compared with the 1-layer CNN. And the nMNIST dataset is fed to this 2-layer CNN as test set, the accuracy drops to 92.03%. The summarized result is shown in Table 5. We can see that for CNN, although the accuracy is much better than the linear algebra methods, such as SVD and NMF, when it perturbed by some noise, the high accuracy of CNN cannot hold. The level of accuracy drop depends on the types of perturbation and the architecture of the CNN. Based on the two tests I have done here, the accuracy of the CNN classifier is jeopardized by the noise introduced in the testing set.

	Noise Free Test Set	Noise Added Test Set
1-layer CNN	99%	90.15%
2-layer CNN	99.31%	92.03%

Table 5 Stability study of 1-layer and 2-layer CNN

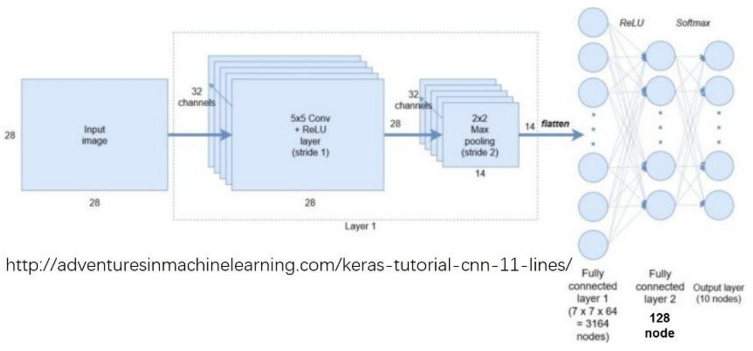

28
Input image
28
32 channels
5x5 Conv + ReLU layer (stride 1)
28
28
32 channels
2x2 Max pooling (stride 2)
14
14
flatten
Layer 1
ReLU
Softmax
Fully connected layer 1 (7 x 7 x 64 = 3164 nodes)
Fully connected layer 2
128 node
Output layer (10 nodes)
http://adventuresinmachinelearning.com/keras-tutorial-cnn-11-lines/

Figure 6 1-layer CNN in the MNIST application

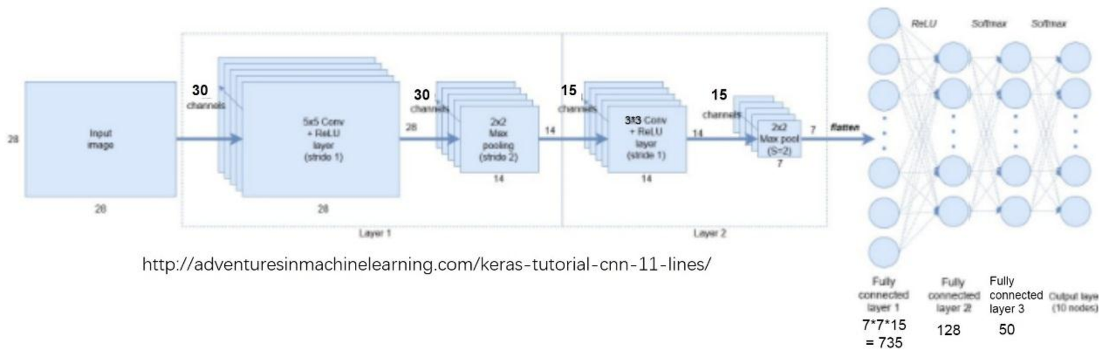


Figure 7 2-layer CNN in the MNIST application

Conclusion

In this report, SVD and NMF are implemented and used in the MNIST classification problem. The prediction accuracies of both methods are higher than 92%: 95% for SVD and 93% for NMF. Both SVD and NMF can resist perturbations to the system, such as gaussian noise. On the other hand, both 1-layer and 2-layer CNN have equal to or higher than 99% classification accuracy, but neither of them can survive the gaussian noise perturbation test. 1-layer CNN prediction accuracy drops to 90% and 2-layer CNN prediction accuracy drops to 92%. It seems that CNN can achieve higher prediction accuracy under the assumption that the testing sample are collected in the same controlled condition as the training set. In the reality and real-world applications, it cannot always be the case. If a well-trained CNN classifier is exposed to new types of data, the prediction accuracy is most likely to be jeopardized. From the robust point of view, SVD and NMF are better classifier for providing acceptable accuracy and can resist some level of perturbation (noise).

Reference

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE*, 86(11):2278-2324, November 1998.

[2] Saikat Basu, Manohar Karki, Sangram Ganguly, Robert DiBiano, Supratik Mukhopadhyay, Ramakrishna Nemani, Learning Sparse Feature Representations using Probabilistic Quadtrees and Deep Belief Nets, European Symposium on Artificial Neural Networks, ESANN 2015.

[3] M. Mazack. Algorithms for Handwritten Digit Recognition." Master’s colloquium, Mathematics Department, Western Washington University, 2009

[4] M. Mazack. “Non-negative Matrix Factorization with Applications to Handwritten Digit Recognition.” Department of Scientific Computation, University of Minnesota, 2009

[5] CSIC 5304 Lecture Note 10-2017 Fall-by Y.Saad

[6] M.W. Berry et al. (2007), "Algorithms and Applications for Approximate Nonnegative Matrix Factorization," Computational Statistics and Data Analysis, vol. 52, no. 1, pp. 155-173.

[7] https://machinelearningmastery.com/handwritten-digit-recognition-using-convolutional-neural-networks-python-keras/

[8] Ciregan, D., Meier, U., and Schmidhuber, J. (2012). Multicolumn deep neural networks for image classification. In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pages 3642–3649. IEEE