HMM for MNIST Handwritten Digit Data Classification and Reconstruction

Ping-Ko Chiu, Wei Zuo, Jong Bin Lim
University of Illinois at Urbana-Champaign

***Abstract*—Hidden Markov Model is widely applicable in many area such as sequence prediction, speech recognition, bioinformatics, or any other fields where data is treated as time-series sequence data. In this paper, Mixed National Institute of Standards and Technology (MNIST) handwritten digits dataset is treated as sequential time-series data, and ten HMM's of each digit was correspondingly trained. These trained ten HMM's can not only be used to predict and classify the random input digit, but also be used to reconstruct the image when the certain parts of the input image are missing. When the input image is complete, the ten trained HMM's successfully classified the label at the accuracy of 70%. We assumed three major cases of which the input data could be incomplete – Bottom part missing, middle part missing, and randomly spotty. For those three types of incomplete data, reconstruction was done assuming the correct classification. Our reconstruction method showed better results than other methods such as sample-mean fill, SVD-soft fill, three-nearest-neighbor fill and nuclear-norm-minimization fill.**

***Keywords*—Hidden Markov Model, MNIST, Recognition, Reconstruction**

I. Introduction

Many signals are sequential in nature. Information transmitted through the network are transmitted in a sequential nature, regardless of what type of information that the network is transmitting. For large sized data, the data needs to be broken into small packets or be streamed. In network communication, we often run into problems of missing data packets, complete communication outage, or slow network transfer speeds. Data reconstruction methods allow us to fill the missing data in a way such that the data is still usable. Hidden Markov Model (HMM) provides a robust method to extract the structure of time-varying vector sequences. It is also popular in the fields of speech recognition in that time-series data can well be trained and adopted for prediction of words and sequences. In this paper, Mixed National Institute of Standards and Technology (MNIST) handwritten digit data set were treated as time-series data. The 2-dimensional image data were divided into certain number of chunks, and each chunks of data represents a data at a time. Since human's handwritting of digit or character usually starts from the top and ends at the bottom, we have treated the 2-dimensional image data as 1-dimensional time-varying vector sequences as if an HMM would give a model for human's handwritting. Also, we chose HMM as an appropriate model for this work because the HMM can not only emit likelihood of certain sequence, but also, once the most likely sequence or HMM is identified, can be used to reconstruct the original data when there are some missing parts. Our HMM's can be treated as a data-generating model where new data can be generated by using the model alone. As shown in the figure 1, we assumed three typically different types of missing data. Since pixel data or packet loss could always occur on any imperfect communication systems, the three possible cases, but not limited to, are: 1) randomly spotty data, 2) middle part missing, and 3) bottom part missing. For the first case, this would be a scenario of when noise is added during data transfer on a well constructed communication channel. Also,

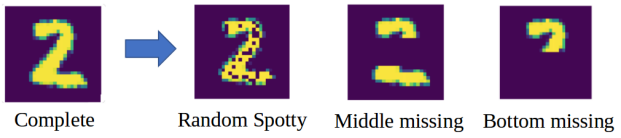

Figure 1. An Example of Complete MNIST Handwritten Data Image and Corresponding Three Types of Incomplete Image

some parts of the data communication are randomly lost during the transfer. The second case would be an instantaneous disconnection during transfer, and resumption of transfer. Third case would be an instantaneous disconnection, and no resumption of transfer occurs. Of course, there are different well-defined communication protocols with hand-shaking protocol to prevent any of this miscommunication scenarios. However, on a cheap communication channel, with a well-trained HMM, our work shows that it is still possible to recognize, retrieve, and reconstruct the original sender's data.

With the MNIST data set, we used about 70% of the data set to train ten HMM's for all ten digits. Then, we used the remaining 30% to test the accuracy and evaluate the quality of reconstruction. Once the missing data is recognized, the quality of the reconstruction is evaluated both qualitatively and quantitatively. Qualitatively, human can quickly observe the image visually, and determine if the reconstructed image data reasonably looks original. Quantitatively, we use Mean Squared Error (MSE) to measure the error between the reconstructed image and the original image, and numerically evaluate the similarity of the reconstructed digit image. There are other data reconstruction techniques with decent quality, but many of those do not consider the time-varying nature of the data, and may only achieve limited error rates for reconstruction [1], [2]. For an example, models like CNN's are suited for the task of MNIST digits classifications [3]. However, for the task of data reconstruction, HMM is natural at performing next-sequence generation, data completion, and reconstruction. This makes HMM's a naturally suitable model for us to explore when dealing with these kinds of reconstruction tasks.

II. Background

Hidden Markov Model (HMM) was first introduced and studied in the late 1960's and early 1970's. HMM was widely studied in the fields of signal processing where a good signal model can realize important practical systems – e.g., prediction systems, recognition systems, identification system, etc. HMM is referred to as Markov sources or probabilistic functions of Markov chains. To understand HMM better, basic theory of Markov chains first needs to be studied, and the knowledge can be extended to hidden Markov Models. Hidden Markov Model contains three fundamental problems: 1) to calculate the likelihood (or probability) of an observed sequence given an HMM, 2) to determine the best sequence of model states, 3) to train

the HMM by adjusting HMM parameters to best explain the observed signals[4].

A. Hidden Markov Process

In discrete Hidden Markov models, each state corresponds to an observable physical event. However, the difference of hidden Markov Model comes from the case where the observation itself is a probabilistic function of the state. Thus, the resulting model is an embedded stochastic process with an underlying stochastic process that is *not* observable or 'hidden'. However, this underlying stochastic process can only produce the sequence of observations. There are several elements of an HMM that explain how an HMM generates observation sequences.

1) N, the number of states in the model. We denote the individual states as $S = \{S_1, S_2, ..., S_N\}$, and the state at time t as q_t.
2) M, the number of distinct observation symbols per state. It is the observation of physical output of the system being modeled. We denote the individual symbols as $V = \{v_1, v_2, ..., v_M\}$.
3) The state transition probability $A = \{a_{ij}\}$ where
$$a_{ij} = P[q_{t+1} = S_j | q_t = S_i], 1 \leq i, j \leq N. \quad (1)$$
For most HMM's, we would have $a_{ij} = 0$ for one or more (i, j) pairs. But, for some cases where any state can transition to any other state in a single step, $a_{ij} > 0$ for all i and j.
4) The observation symbol probability distribution in state j, $B = b_j(k)$, where
$$b_j(k) = P[v_k \text{ at } t | q_t = S_j], \ 1 \leq j \leq N \\ 1 \leq k \leq M \quad (2)$$
5) The initial state distribution $\pi = \{\pi_i\}$ where
$$\pi_i = P[q_1 = S_i], \ 1 \leq i \leq N. \quad (3)$$

B. Three basic problems for HMM[4]

1) Given an observation sequence and an HMM model, what is the probability of observing this sequence under the model?
2) Given an observation sequence and an HMM model, what is the optimal hidden state sequence that represents the observation sequence?
3) Given a set of observation sequences, how do we build an HMM to maximize problem 1?

These three basic problems detailed by Rabiner are applied to our problem of data reconstruction. We go in reverse order. Solution to Problem 3) will allow us to train an HMM that best represents out data. Given images of the same digit, we find a model that maximizes the probability of observing these images. Details of training will be discussed in the training section of this report. Solution to Problem 2) will aid us in the reconstruction of missing data. Given a piece of data that has missing parts, we can identify the most likely state sequence of the HMM that corresponds to this data and generate a new image that fills the missing parts. Solution to Problem 1) will make the classification of digits possible. Given 10 different HMM's trained on the different digits and a new image to be classified, we can calculate the probability of this new image under the 10 different HMM's and take the $argmax$ to be the classification of this image. Combining this with the solution of Problem 2, makes the reconstruction of any randomly chosen digit data possible.

III. Training Hidden Markov Model

In our study, MNIST handwritten digit image data were modeled as Hidden Markov Model. Number of states were defined, and each state has an emission distribution that represents a chunk of data. A sequence of states then represent a sequence of chunks of data that, when stitched together, forms a complete image of handwritten digit. Given the transition matrix, emission distribution, and starting probabilities of an HMM, there are certain sequences of states that have higher probabilities to be emitted than that of other sequences of states. For each of the ten individual handwritten digit image data, ten HMM's were trained. Then, these trained models could achieve the following:

1) A random walk of the Markov network that represents the digit 1, for example, will likely emit a sequence that is similar to a 1.
2) The probability of observing a digit in its corresponding HMM should be higher than its probability in HMM's of other digits. This may involve some kind of normalization of probabilities among many HMM's.
3) Given an image, we can generate the most likely HMM state sequence that represents the image and use it to generate a reconstruction of the image.
4) Given an image that is incomplete, we can generate the most likely partial HMM state sequence. Using this partial state sequence, we can generate most likely sequences to complete the image.

The HMM's are trained with a few assumptions. We assume that the data chunks are Gaussian. This means that we fit a multivariate Gaussian mean and variance to the data set. For the MNIST data set, this needs not to be true since the distribution of the pixel intensities are somewhat bi-modal with pixel values either near 0 or 1. Modeling a multinomial distribution might also be reasonable. However, we decided not to model the multinomial as our first attempt. Instead, we assumed our data chunks to be Gaussian.

We tried different parameters and HMM structures which is described as follows.

HMM state sequencing structure: We tried two types of HMM state sequencing for modeling a given image: Row-based state walk and grid-based state work, which are shown in figure 2.

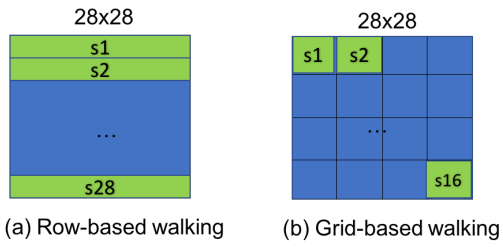

Figure 2. Two types of state sequencing methods

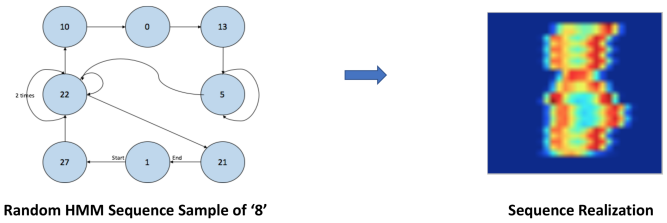

Figure 4. Random HMM Sequence Sample of digit 8 and its Sequence Realization

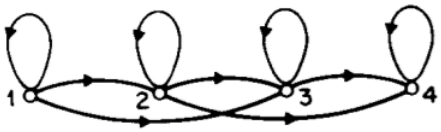

Figure 3. 4-state Left-Right Model

For row-based state walking, we divided the handwritten digit image data horizontally. Each state has a dimension of 28 pixels, which is a row. We expect the state sequence can capture the continuous writing strokes across the row. For grid-based state walking, we separate the image into 2-D grids, where each gird is a state. The reason we tried this is because besides capturing the sequence relation horizontally (between the rows), it also can capture the sequence vertically.

The choice of the number states: We experimented with different number of states, which would dictate the flexibility of the model. The more number of states we define, the more subtle the structure within the digits could be modeled. However, it also means that our data can overfit with a too complicated model. After our experiments, we found that since MNIST contains fairly simple pattern, there is no need to have many states.

The choice of covariance types: We tried different covariance types. *Full covariance* where all entries in the covariance matrix is learnable; *Diagonal covariance* where only the diagnoal of the covariance matrix has non-zero values, namely different dimensions are independent. *Spherical covariance* which is a diagonal covariance matrix where all the non-zero entries have the same value. *Tied covariance* which is a full covariance matrix where all entries have the same value. *Fixed diagonal covariance matrix* where it a diagonal covariance matrix with fixed small values. The small variances guarantee that our output will not be noisy across different samples. After our experiments we found that diagonal covariance or fixed convariance gives the best results. We believe for a simple dataset such as MNIST, where each image contains very sparse information (little variation) the simple convariance matrix is enough to capture the variation.

The choice of transaction matrix and starting state: The initial transition matrix and the starting probabilities act as priors to the HMM training process. Using a custom transition matrix, we can impose different network structures. To help HMM further constrain the state walking, we tried to impose a Left-Right model where states are only allowed to transition in one direction as shown in 3.

In terms of technical implementation and training of the model, we used the "HMMlearn" package [5]. The package allows to specify the emission probability distribution, prior starting probability, transition matrix, and other parameters. Using the Baum-Welch algorithm [6], we iteratively update the model parameters (state emission probabilities, starting probability, and transition matrix) to improve our model. By iteratively increasing the probability of observing the data under the model, we have a model that represents the training data. Using the Viterbi algorithm [7], we can find the most likely hidden state sequence given an observation. This allows us to reconstruct images and predict partial sequences. We have trained the ten HMM's, and generated a random HMM sequence sample of digit '8' and its sequence realization as shown in figure 4. The left hand-side figure shows the random HMM sequence sample of digit '8'. The sequence was realized and generated based on the most likely sequence of the trained HMM. Also, with the trained HMM of digit '0', we provided the actual image data, and predicted state sequence. Then, with this predicted state sequence, we reconstructed the image. Qualitatively and visually, the two images look not very different because the predicted state sequence of digit '0' is the most likely predicted sequence, and pixel data of its corresponding state was reconstructed as shown in figure 5.

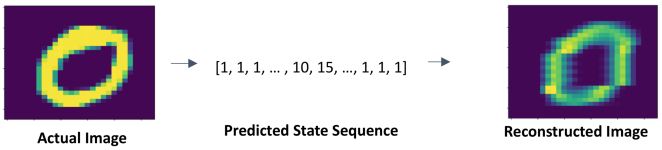

Figure 5. Predicted HMM State Sequence of digit '0', and Reconstructed Image with Pixel Data of Corresponding State Sequence

IV. Classification

A. Classification of Complete Input

With our ten trained Hidden Markov Models of ten digits from '0' to '9', we evaluated the classification (or recognition) accuracy. The original MNIST data is provided for qualitative evaluation as shown in figure 7. We have tried several different methods to train the best HMM as possible by changing certain parameters such as number of states, covariance type, and covariance update over iterations of training. Figures 8, 10 and 11 show the emission result of all ten HMM's. The average accuracy was 70% with number of states being 14, diagonal covariance matrix, and covariance matrix being not updated after each iteration of training as shown in figure 11.

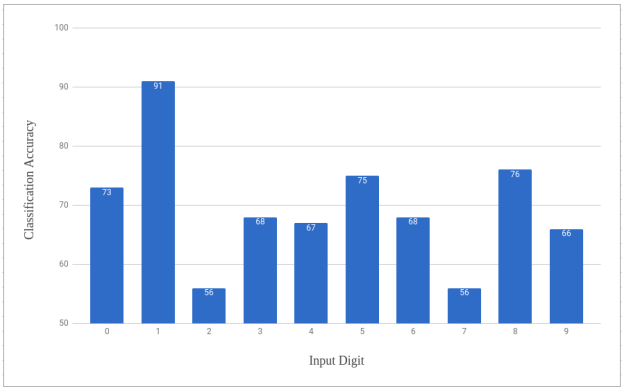

Figure 6. Classification Accuracy of Random Complete Input Digit

Figure 7. Randomly Selected Original MNIST Data

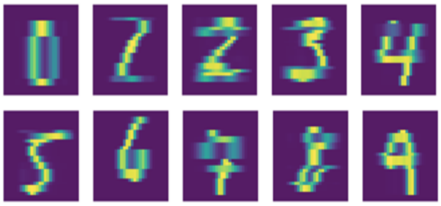

Figure 8. HMM Emission with 14 States, Spherical Covariance Type and Classification Accuracy of 56%

The overall result is shown in the following figure 6. The overall accuracy of the ten HMM's showed an average accuracy of 70%. However, it is noticeable that certain digits had higher classification accuracy. For example, the accuracy of the digit '1' was the highest 91% whereas that of the digit '2' and '7' are both the lowest 56%. This trend comes from the geometric shape of the digit itself. Since the HMM's were trained by horizontally divided chunks of pixel data, digits with vertically symmetric shapes such as '0', '1', '5', and '8' had accuracy higher than the overall average. However, digits with vertically non-symmetric shapes such as '2' and '7' showed low classification accuracy. In the case of '5', although the shape was not exactly symmetrical vertically, the classification accuracy was higher than average 70%.

B. Classification of Incomplete Input

Given an incomplete piece of data, we can still attempt to classify the data. The three kinds of missing data was discussed in the earlier sections of the report. They are: spotty missing, bottom missing, and middle missing.

For spotty missing, we can still pass the data with spots of missing data to our HMM. Using the forward algorithm, we can compute the most likely sequence the the probability of the most likely sequence. Although the data is incomplete, if we assume the spotty missing data to be some form of Gaussian noise, then the data with noise should still be relatively closer to the mean of a particular state than the data that is missing a very significant feature rich slice.

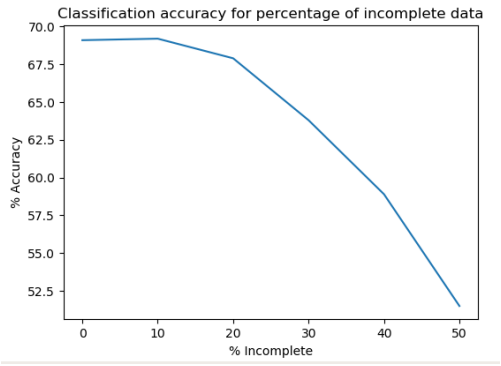

Figure 9. Classification Accuracy with bottom missing data

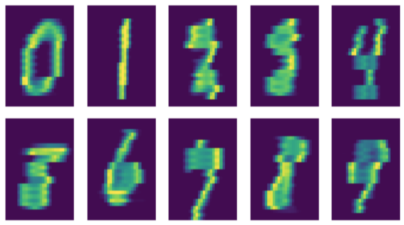

Figure 10. HMM Emission with 14 States, Full Covariance Type and Classification Accuracy of 53%

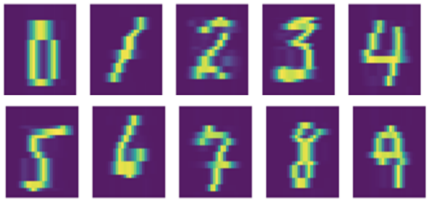

Figure 11. HMM Emission with 14 States, Spherical Covariance Type and Classification Accuracy of 70%

For bottom missing, we can compute the probability of the existing data. Since we are assuming the first $x\%$ of the data to be available, we can use the existing data to predict the most likely state sequence and the probability of observing this image under the HMM model. In Fig. 9 we can see the classification accuracy at various missing %. With 100% of the data, our classification accuracy is 70%. When 50% of the data is missing, the classification accuracy drops to 50%. This implies that given half of the data, we can still predict the 10 digits with 50% accuracy.

For the case of the middle missing data, we complete the similar procedure but instead of passing just the first continuous data, we pass in the concatenation of all the available data in the order it was received. In the view of an HMM, each state has a certain probability to transition ahead of itself to many states beyond its most likely next state. This observation allows us to pass in the incomplete data into the HMM and get a measure of probability among all the 10 different HMMs.

The classification accuracy of partial data is dependent on the classification accuracy of the full data. Therefore if the classification accuracy of the full data can be improved, we believe that the classification accuracy of the partial data can also be improved.

C. Attempt to Improve Classification Accuracy with Regression Layer

We tried several methods to increase the classification accuracy. One thing we observed is that the scoring for each trained HMM is not normalized. For example, it is possible that one HMM gives higher scores to all 10 digits and another HMM gives lower scores to all the digits, while separately each HMM gives the highest score to the correct digit. Therefore, to select the correct HMM for a given digit, if we only compare the digit of each HMM and

select the highest score, it may not be accurate. To factor in this situation, we trained a regression layer connecting to the output of the 10 HMMs, which is shown in figure 12. Three types of regressions are experimented. The logistic regression (we used the scikit-learn.logisticregression package), 1-layer fully-connected layer with softmax function and 2-layer fully-connected layer with softmax function (we used Tensorflow).

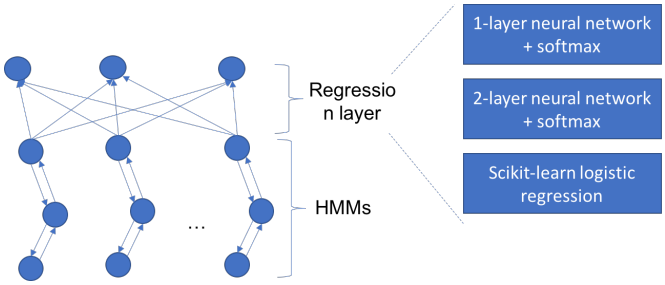

Figure 12. Regression after HMMs. Three types of regressions are used: logistic regression, softmax 1-layer neural network regression, and softmax 2-layer neural network regression

Unfortunately, we could not get better results after the regression. We observed that the accuracy after logtistic regression, softmax 1-layer neural network regression and 2-layer neural network regression are 64.5%, 65.7%, 64.3%, respectively, which is worse than the original accuracy. We could not find a satisfactory explanation for this.

V. Reconstruction

A. Reconstruction of Incomplete Input

The reconstruction process requires that we first classify the incomplete input into one of the 10 classes. Give the classification, we can use the corresponding HMM to reconstruct the missing data of the incomplete input. Therefore a good classification should improve the quality of the reconstruction. In the following sections we will discuss the different methods to fill or reconstruct the missing parts of the input.

1) Random Spotty: Given a randomly spotty data. We compute the most likely state sequence and construct a realization of the data like Fig. 5. Similar to the idea of calculating a probability of observing the data from the HMM, we compute the most likely sequence corresponding to this probability and reconstruct the image. The reconstructed image should be similar to the original image with the missing part reconstructed/filled. We can then impose the reconstructed image onto the original incomplete input to reconstruct the image.

2) Bottom Missing: If the bottom of the image is missing, we compute the most likely sequence with the data that we have and continue to generate the most likely states after the given data. This consists of two parts. The first part is the sequence prediction where the most likely sequence for the available data is calculated. This step gives us an indication of the last state of the HMM that is observed before the next piece of data is missing. Given the state that the currently available data ends with, we can continue to traverse the HMM states using the transition probability matrix. We compute the remaining sequence with the highest probability and concatenate this sequence with the initial sequence generated from the available data. Given the full sequence, we can reconstruct the image similar to Fig. 5. We can then impose our reconstructed data to the incomplete data to reconstruct the full image.

3) Middle Missing: If a middle chunk of the data is missing, we will compute the most likely sequence for the missing slices. This is the same method as the bottom missing case. With the available data, we compute the most likely sequence and use the sequence to complete the missing portions. this will be done multiple times until all of the middle missing slices are filled.

B. Compare out method with different imputing methods

We expect with the help of HMMs, we could reconstruct the data even when there is a large and continuous chunk missing. To test the effectiveness of our method, we compared our method with the following beseline methods, which are implemented in python package `fancyimpute` [8].

- SimpleFill: Replaces missing entries with the mean or median of each column.
- KNN: Nearest K neighbors imputations which weights samples using the mean squared difference on features for which two rows both have observed data.
- SoftImpute: Matrix completion by iterative soft thresholding of SVD decompositions [1].
- NuclearNormMinimization (NNM): Simple implementation of Exact Matrix Completion via Convex Optimization.

We focused on the incomplete input with `bottom missing` since this is the most difficult case to reconstruct the data. To test the result, we computed the mean square error (MSE) for different imputing method. Here we focus on 40% of data is missing. The results are shown in figure 13.

From the result we can see that our method does not always give lowest MSE. We believe this is not a good measurement for this case because MSE is very vulnerable to shape changing: when the shape does not match the original input, even it is a meaningful reconstruction, it contributes to error. However, from Figure 13 we can clearly observe in many cases, our HMM can successfully impute the data while other fails. Such as digit “0”, “1”, “3”, “5”, “7” and “8”. In some cases, the HMM could not impute correctly, such as digit “9”. The miss reconstruction is because of two factors. First the classification is incorrect. We can only achieve 70% accuracy, so in some cases with the incomplete digit, it classifies incorrectly. For example, “9” is classified as “8”, which is a mistake. Second, since the fill-in blocks are generated based on random sampling, according to the trains HMM distributions, with some probability, it cannot give exactly the correct output. However, it is very clear that with the help of HMM, our method can reconstruct the digits much better than other methods.

VI. Conclusion

In this project we studied HMMs with MNIST handwritten digits dataset. Each digit is treated as sequential time-series data, and ten HMM’s of each digit was correspondingly trained. First We use the trained HMMs to classify the random input digit, including complete and incomplete digits. Second we used the HMMs to reconstruct the incomplete input digits. Three cases of data missing are considered, namely bottom part missing, middle part missing, and

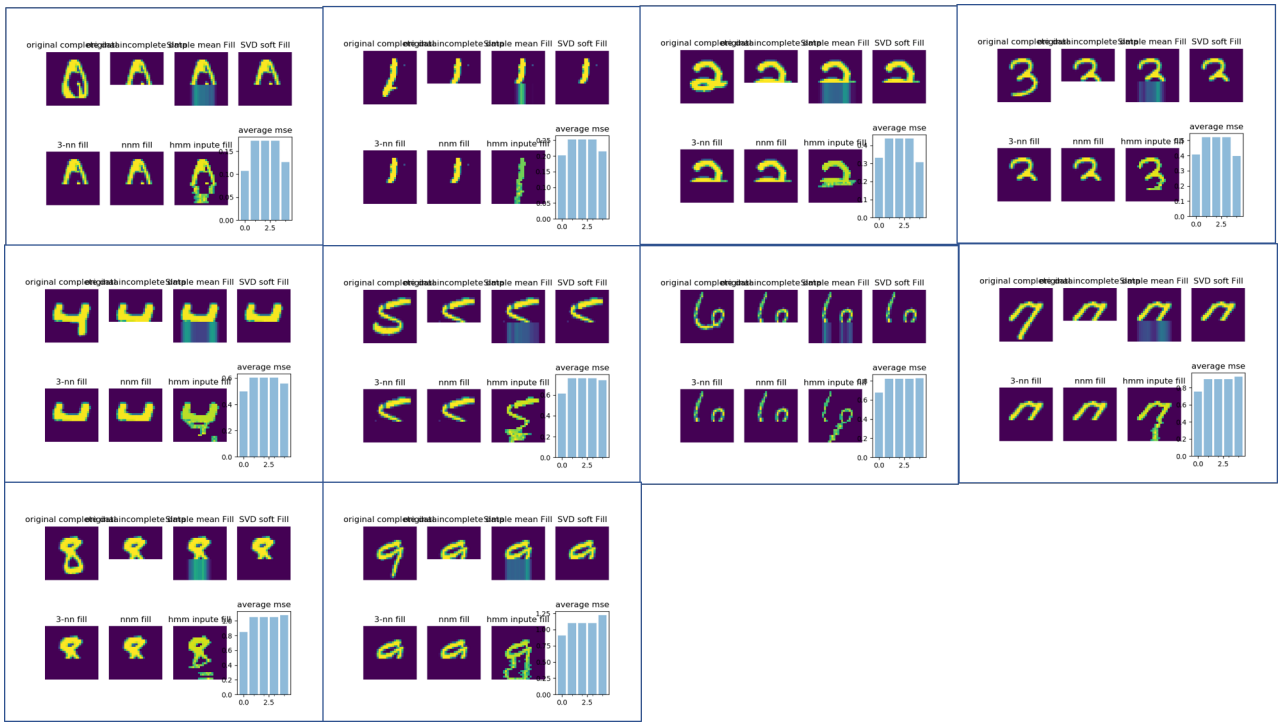

Figure 13. The reconstruction of 10 digits, comparing with baseline. For each digit, the order of the pictures are: original digit, incomplete digit, simpleFill result, softImpute result, 3NN result, NNM results, Our result. The bar chart is the MSE of these five impute algorithms correspondingly

randomly spotty. Our reconstruction method showed better results than other methods such as sample-mean fill, SVD-soft fill, three-nearest-neighbor fill and nuclear-norm-minimization fill.

REFERENCES

[1] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *J. Mach. Learn. Res.*, 11, August 2010.

[2] Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Found. Comput. Math.*

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

[4] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, Feb 1989.

[5] Sergei Lebedev. hmmlearn, 2015. http://hmmlearn.readthedocs.io/.

[6] P. M. Baggenstoss. A modified baum-welch algorithm for hidden markov models with multiple observation spaces. *IEEE Transactions on Speech and Audio Processing*, 9(4):411–416, May 2001.

[7] G. D. Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, March 1973.

[8] fancyimpute. https://github.com/hammerlab/fancyimpute.