Non-Negative Matrix Factorization*

Charles Stoksik, Olivia Heiner, Teresa Rexin, Christopher Clegg[1]

***Abstract*— In this project we analyze the Unsupervised Learning technique of Non-Negative Matrix Factorization on both a mathematical and applied level. In our mathematical analysis, we will be summarizing different initialization methods, different convergence methods, and how the algorithm differs from some other unsupervised learning techniques. In our applied section, we will look at how NMF can be applied to the MNIST data-set. We also use NMF to visualize and analyze the ideological positions of U.S. Congress members, based on their voting behavior. In the reduced dimension space, it becomes much more feasible to compare ideological stances of congress members and political parties.**

I. INTRODUCTION

The field of Machine Learning is often broken up into three separate fields of interest: Supervised Learning, Unsupervised Learning, and Reinforcement Learning. These three classes of Machine Learning often have different goals and use different types of models and data-sets. The goal of this project is to explore the Unsupervised Learning method of Non-Negative Matrix Factorization and its applications. We chose to focus on an Unsupervised method because of its real world impacts. The majority of the data available in the world is unlabeled, meaning it is ideal for unsupervised learning models. While not always as predictive, or useful, as a supervised method, our exploration of Unsupervised Learning illustrates a technique optimal for storing less data and learning the main features of the data.

Non-Negative Matrix Factorization is designed for factoring non-negative matrices into two similarly non-negative matrices that when multiplied result in a positive low rank approximation of the original matrix. If we want to factor positive $\mathbf{X} \in \mathbb{R}^{m \times n}$ we can do so by solving the following optimization problem:

$$\arg\min_{W,H} ||\mathbf{X} - \mathbf{WH}||_F \tag{1}$$

Where a hyperparameter k is chosen such that the dimensions of $\mathbf{W} \in \mathbb{R}^{m \times k}$ and $\mathbf{H} \in \mathbb{R}^{k \times n}$.

A. Notation

We will be referring to $\mathbf{W}$ as the Feature Matrix, where each column is a feature of the initial $\mathbf{X}$ data-set. We will be referring to the $\mathbf{H}$ matrix as the Coefficient Matrix, as each columns contains the coefficients needed to multiply by $\mathbf{W}$ to achieve what is closest to the initial data entry in $\mathbf{X}$.

*This work was not supported by any organization

[1]Department of Mathematics, University of California, Los Angeles, 520 Portola Plaza, Los Angeles, CA 90095, USA

B. Organization

We will be organizing this paper into three following parts: Mathematical Foundations, Different Methods of Convergence, Experiments, and Conclusions.

II. MATHEMATICAL FOUNDATIONS

A. Mathematical Formulation

Clustering is a method of organizing data to solve classification problems, where each cluster is a subset of the data that are close in distance. Intuitively, the data points within each cluster are more related to one another than with data points in other clusters. Non-Negative Matrix Factorization is a clustering technique in which the means of each cluster are represented by basis vectors, and the coordinates of data points are represented with respect to their basis vector. Finding a low-rank approximation using Singular Value Decomposition (SVD) can often lead to factors with negative elements. To ease interpretation of such approximations in the context of their applications, algorithms for finding a Non-Negative Matrix Factorization were developed.

Given a matrix $\mathbf{X}^{m \times n}$ and an integer $k \leq m, n$, the NMF problem is finding a factorization of $\mathbf{X}$ into $\mathbf{W}^{m \times k}$ and $\mathbf{H}^{k \times n}$ such that $\mathbf{X} \approx \mathbf{WH}$. The basis vectors make up the columns of $\mathbf{W}$, and the columns of $\mathbf{H}$ represent the coordinates, or coefficients, of a data point with respect to the basis vectors. Having non-negative entries for $\mathbf{W}$ and $\mathbf{H}$ allows us to more clearly interpret the importance of the basis elements in representing each data point.

NMF is considered to be a nonlinear optimization problem for both $\mathbf{W}$ and $\mathbf{H}$. If $\mathbf{W}$ or $\mathbf{H}$ is known, this problem would become the standard least squares problem. Thus, a popular method of solving Equation 1 is by using alternating non-negative least squares (ALS), where one first solves for one of the matrices and then solves for the other. This is described in Algorithm 1.

A common method of initializing $\mathbf{W}$ is a variation on the SVD of $\mathbf{X}$. If $\mathbf{X} = \mathbf{W\Sigma V}^T$ is the SVD of $\mathbf{X}$, the first singular vector u_1 is the first column in $\mathbf{W}^{(1)}$. Conversely, v_1^T can be the first row in $\mathbf{H}^{(1)}$. The second singular vector u_2 will likely have negative elements due to orthogonality. Replacing all of the negative components in u_2 with zero, we can compute $\mathbf{C}^{(2)} = u_2 v_2^T$. Then, replacing the the negative elements of $\mathbf{C}^{(2)}$ with zero to obtain $\mathbf{C}_+^{(2)}$, the first singular vector of $\mathbf{C}_+^{(2)}$ will be non-negative and a good approximation of u_2. Thus, we can use the first singular vector of $\mathbf{C}_+^{(2)}$ as the second column of $\mathbf{W}^{(1)}$. This process can be repeated to compute the remaining columns of $\mathbf{W}^{(1)}$ [6].

Algorithm 1 Non-Negative Matrix Factorization

Require: $\mathbf{X} \in \mathbb{R}^{m \times n}$, $\mathbf{W} \in \mathbb{R}^{m \times k}$, $\mathbf{H} \in \mathbb{R}^{k \times n}$

1: Set an initial value for $\mathbf{W}^{(1)} \geq 0$.
2: For k = 1, 2, ... until convergence, repeat steps 3 and 4:
3: Solve $\mathbf{H}^{(k)} = \min_{\mathbf{H} \geq 0} ||\mathbf{X} - \mathbf{W}^{(k)}\mathbf{H}||_F$.
4: Solve $\mathbf{W}^{(k+1)} = \min_{\mathbf{W}^{(k)} \geq 0} ||\mathbf{X} - \mathbf{W}\mathbf{H}^{(k+1)}||_F$.

Furthermore, rewriting Equation 1 in terms of the columns of $\mathbf{X}$, x_j, and $\mathbf{H}$, h_j, it is shown that the matrix least squares problem in Equation 1 is equivalent to the vector least squares problem,

$$\min_{\mathbf{h_j} \geq 0} ||\mathbf{x_j} - \mathbf{W}^{(k)}\mathbf{h_j}||_F$$

for j = 1, 2,..., n. After transposing the matrices, Equation 1 is equivalent to m vector least squares problems which is useful when implementing Algorithm 1 by hand. ALS and Algorithm 1 are further explored in section III.

NMF does not produce a unique factorization for $\mathbf{WH}$. One can introduce a diagonal matrix D to normalize the factors, such as by scaling the columns of W so that the largest element in each column is 1:

$$\mathbf{WH} = (\mathbf{WD})(\mathbf{D}^{-1}\mathbf{H})$$

Additionally, since ALS is is computationally expensive and time consuming, one can utilize the thin QR decomposition $\mathbf{W} = \mathbf{QR}$ as an alternative, yielding,

$$\mathbf{H} = \mathbf{R}^{-1}\mathbf{Q}^T\mathbf{X}$$

This is a much cheaper solution to solve for $\mathbf{W}$ and $\mathbf{H}$ [6].

B. Comparison to Other Factorizations

K-means is similar to NMF in that both are clustering techniques that calculate representative values of each of the clusters. Instead of basis vectors, k-means calculates the mean value of each cluster called centroids. After an initial random partitioning into k, in each iteration the centroids of each cluster are calculated and the Euclidian distance between each data point and centroid is measured. Data points are reassigned to the cluster with the centroid they are closest in each iteration. Given data in matrix $\mathbf{X}$, the centroids can be thought of as the basis vectors in $\mathbf{W}$ from NMF, and the coordinates of $\mathbf{X}$ in terms of the centroids or basis are the columns of $\mathbf{H}$.

Principal Component Analysis (PCA) is also related to NMF. Projecting the data onto a space having dimensionality $k < m$ and finding the eigenvalues of its covariance matrix is similar to finding the basis matrix $\mathbf{W}$. Instead of representing the correlation of each data point with the basis vectors as in NMF, with PCA each data point can be represented as a linear combination of these basis vectors. The corresponding coefficients make up the matrix $\mathbf{H}$. However, the matrices in PCA are often a mix of positive and negative components, while the matrices produced by NMF must be non-negative. This contributes to the more difficult interpretation of PCA compared to NMF–the negative coefficients in the linear combination can be hard to understand in the context of the application. [2]

C. Classical Applications

One common application of NMF is text mining. One can represent a set of documents using the bag-of-words matrix representation to generate a matrix $\mathbf{X}$, where each row corresponds to a word and each column represents a document. Then, the NMF of $\mathbf{X}$ will produce a matrix $\mathbf{W}$ whose columns are the basis documents, or topics since a certain topic generally uses words in the same domain. $\mathbf{H}$ would indicate how to sum aspects from different topics to reconstruct the words of a given document. Thus, NMF can identify topics and classify documents according to these topics [6].

Another application of NMF is image processing. Given an image of $m \times n$ pixels, NMF can deconstruct the image into two matrices: one with the features in the image making up the columns of $\mathbf{W}$, and one with of the importance of each feature making up the $\mathbf{H}$. Using the example of an image of a face, the facial features such as the eyes, ears, nose, and mouth, would each represent the columns of $\mathbf{W}$ and $\mathbf{H}$ would indicate which features are present and where they are located on the face to reconstruct the image as a weighted sum of the basis images [12].

III. METHODS

A. Multiplicative Update Rules

Lee and Seung (2000) proposed two multiplicative update rules based around two different choices of cost function[13]. The first cost function considered is the squared Euclidean distance

$$||V - WH||^2 = \sum_{ij}(V_{ij} - WH_{ij})^2$$

For this choice of cost function they propose the following update rule:

$$H_{\alpha\mu} \longleftarrow H_{\alpha\mu}\frac{(W^TV)_{\alpha\mu}}{(W^TWH)_{\alpha\mu}}$$
$$W_{i\alpha} \longleftarrow W_{i\alpha}\frac{(VH^T)_{i\alpha}}{(WHH^T)_{i\alpha}}$$

Lee and Seung point out that this may be considered a gradient descent method of the form

$$H_{\alpha\mu} \longleftarrow H_{\alpha\mu} + \eta_{\alpha\mu}\left[(W^TV)_{\alpha\mu} - (W^TWH)_{\alpha\mu}\right]$$

with

$$\eta_{\alpha\mu} = \frac{H_{\alpha\mu}}{(W^TWH)_{\alpha\mu}}$$

The next cost function they consider is given by

$$D(V||WH) = \sum_{ij}\left(V_{ij}\log\frac{V_{ij}}{(WH)_{ij}} - V_{ij} + (WH)_{ij}\right)$$

which reduces to the Kullback-Leibler divergence when $\sum_{ij} V_{ij} = \sum_{ij}(WH)_{ij} = 1$. For this choice of cost function they propose the following update rule:

$$H_{\alpha\mu} \longleftarrow H_{\alpha\mu}\frac{\sum_i W_{i\alpha}V_{i\mu}/(WH)_{i\mu}}{\sum_k W_{k\alpha}}$$
$$W_{i\alpha} \longleftarrow W_{i\alpha}\frac{\sum_\mu H_{\alpha\mu}V_{i\mu}/(WH)_{i\mu}}{\sum_\nu H_{\alpha\nu}}$$

Furthermore, Lee and Seung prove that both cost functions are nonincreasing under their respective update rules. Advantages of these multiplicative update rules are that they are easy to implement, they ensure that the approximations are non-negative, so long as the intial values are non-negative, and the fact that the cost function is nonincreasing means that they offer adequate performance in a number of cases. However, it has been pointed out that this property does not guarantee convergence to a local minimum, and these multiplicative methods can in some cases be quite slow[8].

B. Projected Gradient Descent

Another approach to NMF is to use a gradient descent algorithm. The basic form of such an algorithm for a cost function f is as follows:

$$W \longleftarrow W - \epsilon_W \frac{\partial f}{\partial W}$$
$$H \longleftarrow H - \epsilon_H \frac{\partial f}{\partial H}$$

In the least-squares case, we would have

$$\frac{\partial f}{\partial W} = (WH - V)H^T$$
$$\frac{\partial f}{\partial H} = W^T(WH - V)$$

This step is then followed by projecting each of the approximations onto a constraint space. One reason for this is to ensure non-negativity. Hoyer (2004) proposes a gradient descent based method whereby the approximations for W and H are also subject to sparseness constraints[9]. By controlling the degree of sparseness of one or both components of the factorization, one is able to control the locality of the model's representation of the data. This, in turn, is useful because, as Hoyer explains, this locality is one of the primary appeals of NMF over other matrix factorization methods such as principal component analysis and vector quantization. However, it has been pointed out that without careful choice of ϵ_W and ϵ_H, the convergence of these methods can be difficult to analyze, a difficulty which is further compounded by the projection step necessary to produce non-negative results[1].

C. Alternating Non-Negative Least Squares

Looking once again at the problem of minimizing the squared Euclidean distance cost function

$$||V - WH||^2$$

we observe that, while this problem is not convex in W and H simultaneously, it is convex in each one separately. The idea behind the alternating least squares technique is to fix W and optimize with respect to H, then fix H and optimize with respect to W, or vice versa. This may be formulated as

$$W^{(k+1)} = \arg\min_{W \geq 0} f(W, H^{(k)})$$
$$H^{(k+1)} = \arg\min_{H \geq 0} f(W^{(k+1)}, H)$$

where $H^{(1)}$ and $W^{(1)}$ are initialized as some non-negative matrices. Lin (2007) offers an efficient means of accomplishing this which utilizes projected gradient methods to solve each optimization sub-problem[15]. As Lin points out, these projected gradient methods possess advantageous convergence properties in the case of the subproblems, thus this algorithm tends to yield good results. Kim and Park (2007) offer an alternating least squares algorithm for sparse NMF by instead using the following objective function:

$$\min_{W,H} \frac{1}{2}\left\{||V - WH||_F^2 + \eta||W||_F^2 + \beta\sum_{j=1}^{n}\left(\sum_{q=1}^{k} H(q,j)\right)^2\right\},$$
$$s.t. \quad W, H \geq 0$$

Additionally, they establish that the two-block minimization process for this problem is convergent [10].

D. Hierarchical Alternating Least Squares

Chichoki and Phan (2009) propose an algorithm for NMF which works by minimizing a set of local cost functions with the same global minima as those for the alternating least squares approach[5]. In this fashion, they produce a method which updates each column of W and row of H with an optimal closed-form solution which can be computed quite easily[7]. This amounts to performing coordinate descent with the columns of W and rows of H. Chichoki and Phan begin by defining the following:

$$V^{(j)} = V - WH + w_j h_j^T$$

where the i^{th} column of W is denoted by w_i and the i^{th} row of H is denoted by h_i. They then propose the following update rule:

$$h_j \longleftarrow \frac{1}{w_j^T w_j}\left[V^{(j)T} w_j\right]$$
$$w_j \longleftarrow \frac{1}{h_j^T h_j}\left[V^{(j)} h_j\right]$$

Alternatively, Gillis and Glineur (2012) describe the following update rule, in which they limit their attention to W:

$$W_{:p}^{(k+1)} =$$
$$\max\left(0, \frac{A_{:p} - \sum_{l=1}^{p-1} W_{:l}^{(k+1)} B_{lp} - \sum_{l=p+1}^{r} W_{:l}^{(k)} B_{lp}}{B_{pp}}\right)$$

for $p = 1, 2, ..., r$, where $A = VH^{(k)^T}$ and $B = H^{(k)}H^{(k)^T}$. These methods result in an algorithm that is highly efficient in practice. The primary update method utilized in this paper has been a hierarchical alternating least squares algorithm, optimizing an objective function that includes L1 and L2 norm based penalties to impose sparseness.

IV. EXPERIMENTS

A. MNIST Image Data

MNIST is a popular data-set often used as a baseline for many algorithms. This data-set is made up of grayscale, labeled 28 by 28 images of handwritten digits [11]. Given that when made into a vector images are 784 pixels in length, if we taken 10,000 images and perform a NMF with the hyperparameter of 50 features, our matrices will take the following dimensions:

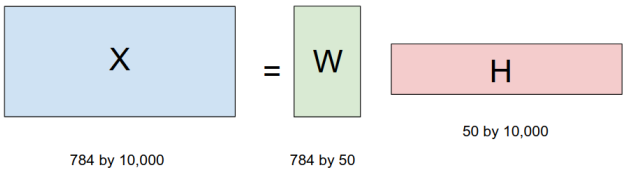

Fig. 1. Dimensions for our NMF with 50 features on 10,000 entries of the MNIST data-set

The $\mathbf{W}$ matrix in 1 is made up of 50 vectors of 784 features. These columns correspond to each possible "feature" of the original image. We essentially take the 50 most common parts of an image and store them in this Matrix column wise. Now if we look at $\mathbf{H}$, we see our 10,000 images, but as vectors of 50 features. Each of these columns correspond to coefficients that dictates how much of the corresponding feature is in the reconstructed image.

Now with a data-set of only 9s we will create a 10 feature model that translates the input 784 data points to just 10 data points. For our model we will use the SVD technique for initialization, the Coordinate Descent algorithm to converge, and solve the optimization problem in (1). Here are the columns of $\mathbf{W}$, which represent our features:

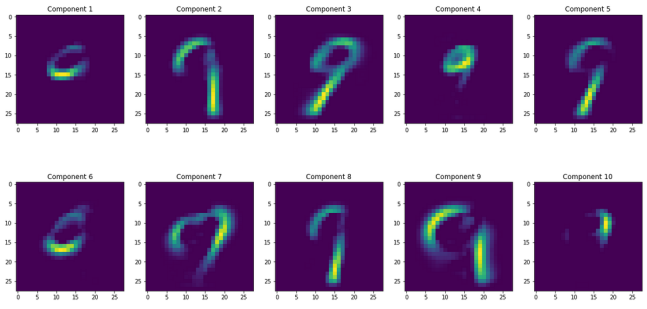

Fig. 2. The 10 features of the 9s NMF model. Also the columns of $\mathbf{W}$.

Having built the matrix $\mathbf{W}$, which has columns in 2, we will 5 new vectors v_i where $v_i \in \mathbb{R}^{784}$ represents an image. Let us denote u_i as a reconstructed vector. Here we can visualize 5 v_i and u_i in 3.

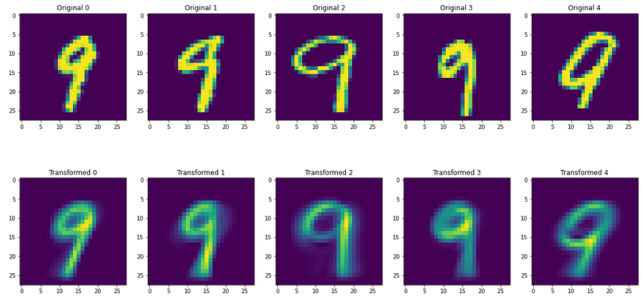

Fig. 3. After creating an NMF model for MNIST data 9s, these are the translations of the data.

Knowing the target values allows us the create an NMF model with just 10 features that still keeps the appearance of a '9' when transformed back into the original feature space.

Next we will attempted to compare different feature sizes. One of the main benefits of NMF is reducing dimension size, and thus reducing memory usage with data. Therefore figure 4 shows the difference in original feature space between models of different feature sizes. Our goal here is to both balance readability of the image with saving memory and computational efficiency in the data.

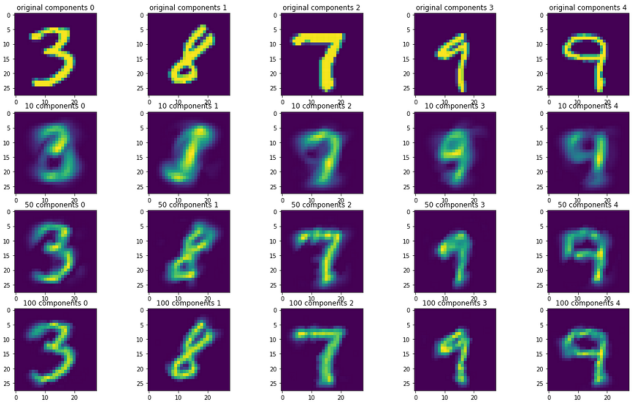

Fig. 4. NMF inverse transform for five images not used in compression models.

As we can see in figure 4, as the number of features increases, so does the quality of the reconstruction. In the 10 feature images, the average person would be hard-pressed to figure out what number is represented in each image. In the 50 feature case it is easier to tell, but the fourth image could give someone a hard time. In the 100 feature case, it is pretty easy to tell. This means that we can now store length 100 vectors, instead of length 784 vectors, and still preserve the interpretation of the images.

We can vary some of the hyperparameter's mentioned in the Mathematical Foundations section. For instance if we change the solver from 'cd', standing for Coordinate Descent, to 'mu', standing for Multiplicative Update. We can also change the initialization from 'nndsvd', which is Non-Negative Singular Value Decomposition to 'random' which means a Random initialization of $\mathbf{W}$ and $\mathbf{H}$. Here are some simple results of these changes:

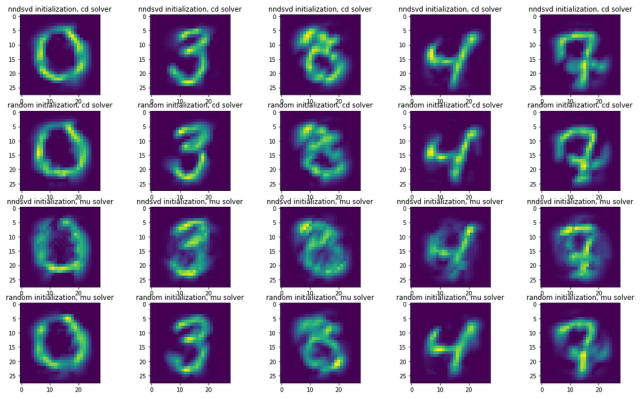

Fig. 5. Changing the solver and the initialization on the MNIST data-set.

Here we used a 50 feature NMF model and each of the results is comparable to each other. It should be noted that the first row where Coordinate Descent and Non-Negative SVD are used would likely be considered the clearest by the average viewer. Although this is a small sample size, it seems to point that the default hyperparamter choices for the Sci-kit Learn package [17] are the most effective.

In order to quantify some results of the previous compression comparisons, we can apply a Random Forest Classifier to our labeled MNIST data. A description of a Random Forest Classifier can be found at [3]. For all our models we will use 50 decision trees. We will split the data into a training set of 10,000 digits, and a testing set of 1,000 digits. In order to quantify our comparisons, we will use an accuracy score.

$$A = \frac{\sum_i \delta(y_i, t_i)}{N} \tag{2}$$

In equation (2), t_i is the label of test instance i, and y_i is the label chosen by the Random Forest Classifier. The Kronecker Delta δ is 1 if $y_i = t_i$, else it is 0 [16]. N is the test set size. Applying a Random Forest Classifier with 50 trees on the MNIST data-set without using an NMF model yields a .931 accuracy score. We will compare this to NMF models with 1 to 100 feature sizes in the graph below:

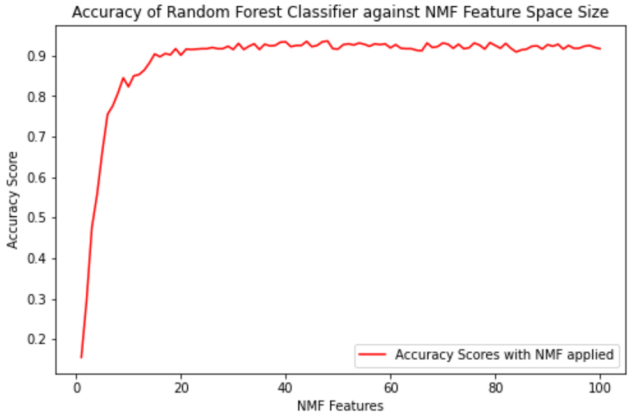

Fig. 6. Accuracy Score of Random Forest Classifier when trained on data with NMF feature space

As we can see in 6, the Random Forest Classifier starts to match the original accuracy score by around the 40 feature mark. We can further analyze this score by comparing to roll outs of four different models with the specified hyperparamters from before. We will do 10 repetitions of a classification procedure with 50 features and visualize the accuracy scores in the figure below.

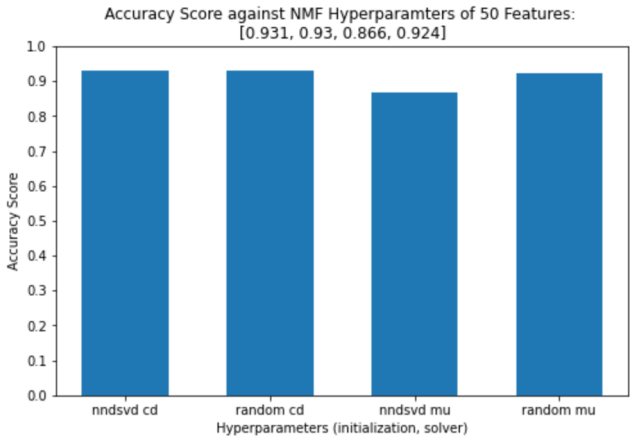

Fig. 7. Average random forest accuracy scores over 10 trials with specified hyperparameters for NMF

In 7, the SVD initialization with the Coordinate Descent Solver scores the best over 10 trials, and has the same accuracy score as the one shot initial Random Forest that did not utilize NMF. Therefore, with basic Random Forest Models, we can remove 784 - 50 = 734 features from memory for all the entries. Overall we achieve the same level of classification accuracy whilst using 6.878% of the memory.

B. United States Congressional Voting Data

We next apply NMF to United States (U.S.) congressional voting data. The data set we use[14] contains Senate and House roll call votes, as well as other data (name, party, state, etc.) about members of congress.

First we apply NMF to the 116$^{\text{th}}$ Senate. After removing two members for which there were missing values, we obtain a data matrix $\mathbf{X} \in \mathbb{R}^{667\times 99}$, where the 667 rows represent the roll call numbers (measures to be voted on) and the 99 columns each represent a member of the Senate. The matrix takes on non-negative values representing yea, nay, present, or abstention votes.

There are two main political parties in the U.S. Congress: Democrat and Republican. Because of this, we choose the column dimension of $\mathbf{W}$ to be 2, resulting in an NMF factorization $\mathbf{X} \approx \mathbf{WH}$ where $\mathbf{W} \in \mathbb{R}^{667\times 2}$ and $\mathbf{H} \in \mathbb{R}^{2\times 99}$. Each column in the coefficient matrix $\mathbf{H}$ represents a senator with 2 values. This divides the data into two clusters: senator j belongs to cluster k where $k = argmax_{i\in\{1,2\}}\mathbf{H}_{ij}$.

In Figure 8, We plot the 2D representation of each senator for the 116th Senate given by the points $(\mathbf{H}_{1j}, \mathbf{H}_{2j})$. In this plot, all points with $\mathbf{H}_{1j} > \mathbf{H}_{2j}$ (right of the boundary line) are in cluster 1. All points with $\mathbf{H}_{1j} < \mathbf{H}_{2j}$ (left of the boundary line) are in cluster 2. One can see from the plot that all Republican senators belong to one cluster whereas all Democrat senators belong to another cluster. Furthermore,

the two Independent senators belong to the same cluster as the Democrat senators.

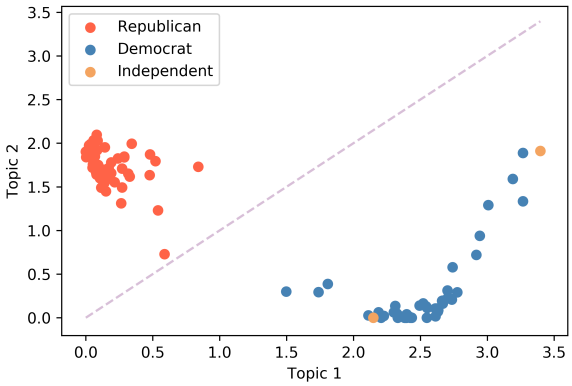

Fig. 8. 116th Senate represented in two-dimensional space, where NMF is used for the dimensionality reduction.

Looking at the plot of Senate data reduced to two dimensions, it is easier to visualize the "ideological" distance between senators, both within their own parties and between parties. The senators at the top-right-most part of cluster 1 are Bernie Sanders, Kamala Harris, Amy Klobuchar, all of whom are known to be more progressive/liberal senators [18]. The bottom-left-most part of the same cluster consists of Kyrsten Sinema, Doug Jones, and Joe Manchin, all of whom are known for being moderate Democrats in support of bipartisan politics [18]. In cluster 2, the bottom-right-most senators are Lisa Murkowski and Susan Collins, who are known for being the two most moderate Republicans [18]. These consistencies between known senator ideologies and our NMF model gives us confidence in the model's ability to generate meaningful "topics". We repeat this experiment for each Senate between 1970-present. Some representative experiments are visualized in Figure 9.

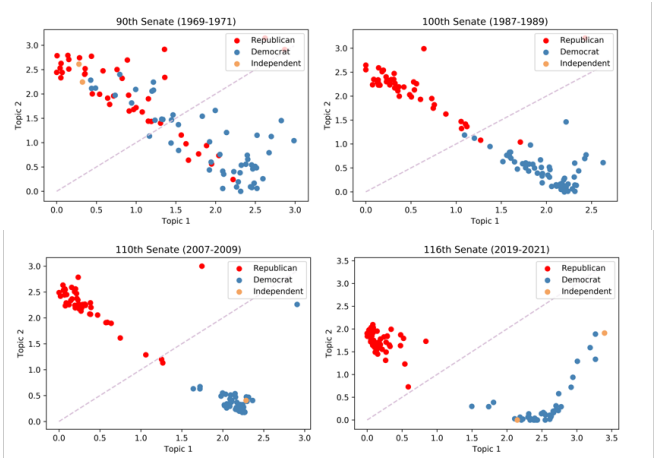

Fig. 9. 90th, 100th, 110th, and 116th Senates visualized in two-dimensional space. Parties become increasingly separable over time.

To mathematically quantify the ideological within-party distance (WPD), we took the mean of the Euclidean distances between the 2D centroid $\mathbf{g}^{\{k\}}$ of the party and each member of the party:

$$\mathrm{WPD}^{\{k\}} = \frac{1}{n_k} \sum_{j \in J_k} \|\mathbf{g}^{\{k\}} - \mathbf{h}_j\|_2$$

To quantify the between-party distance (BPD) we calculate the Euclidean distance between the 2D centroids $\mathbf{g}^{\{k\}}$ of the two parties:

$$\mathrm{BPD}^{\{k\}} = \|\mathbf{g}^{\{1\}} - \mathbf{g}^{\{2\}}\|_2$$

Both these measures are derived from cluster evaluation methods in [4], but are modified so that the WPD takes a mean of Euclidean distances, rather than just a sum. This is so we can compare ideological "distances" across different years where the features and sample size differ. The results for WPD and BPD in the U.S. Senate from 1970 to 2020 are shown in Figure 10.

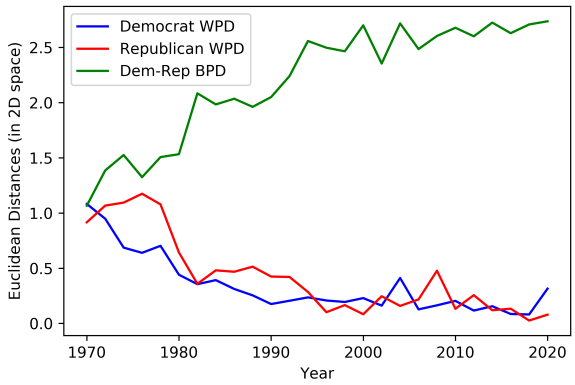

Fig. 10. Ideological Euclidean distances within political parties (WPD) and between political parties (BPD) in the U.S. Senate.

These analyses show increasing polarization between political parties over time, while within-party cohesion is increasing over time. This is an indication that party politics may be becoming more prevalent in determining how politicians vote.

V. CONCLUSIONS

In this paper, we have explored the mathematical principles underlying NMF and the methods used to perform the factorization. We have also compared it to other dimensionality reduction and clustering methods, such as PCA and k-means. We then applied NMF to the MNIST data set, where we showed that MNIST data can be compressed via NMF and still maintain important features that distinguish numbers from one another. Lastly, we used NMF to visualize and quantify the polarization of political parties in the U.S. Senate over time.

REFERENCES

[1] Michael W. Berry, Murray Browne, Amy N. Langville, V. Paul Pauca, and Robert J. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics Data Analysis*, 52(1):155 – 173, 2007.

[2] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

[3] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[4] Igor Brigadir, Derek Greene, James Cross, and Padraig Cunningham. Dimensionality reduction and visualisation tools for voting records. *Insight Centre for Data Analytics*.

[5] Phan A.H. Chichoki, A. Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, E92-A(3):708–721, 2009.

[6] Lars Elden. *Matrix Methods in Data Mining and Pattern Recognition*. Society for Industrial and Applied Mathematics, 2019.

[7] Nicolas Gillis and François Glineur. Accelerated multiplicative updates and hierarchical als algorithms for nonnegative matrix factorization. *Neural Computation*, 24(4):1085–1105, Apr 2012.

[8] Edward F. Gonzalez and Yin Zhang. Accelerating the lee-seung algorithm for nonnegative matrix factorization. 2005.

[9] P.O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

[10] Hyunsoo Kim and Haesun Park. Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics*, 23(12):1495–1502, 05 2007.

[11] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[12] D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[13] D. Lee and H. S. Seung. Algorithms for Non-negative Matrix Factorization. *Proceedings of Neural Information Processing Systems*, pages 556–562, 2000.

[14] Keith Poole Howard Rosenthal Adam Boche Aaron Rudkin Lewis, Jeffrey B. and Luke Sonnet. Voteview: Congressional roll-call votes database. https://voteview.com/. Accessed: 12-2020.

[15] C. Lin. Projected gradient methods for nonnegative matrix factorization. *Neural Computation*, 19(10):2756–2779, 2007.

[16] M. E. J. Newman. *Networks: an introduction*. Oxford University Press, Oxford; New York, 2010.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[18] J. Tauberer. Govtrack dataset. https://www.govtrack.us/congress/votes. Accessed: 2020-10-25.