Noname manuscript No.
(will be inserted by the editor)

A Deep Non-Negative Matrix Factorization

Jennifer Flenner · Blake Hunter

1 Abstract

Recently, deep neural network algorithms have emerged as one of the most successful machine learning strategies, obtaining state of the art results for speech recognition, computer vision, and classification of large data sets. Their success is due to advancement in computing power, availability of massive amounts of data and the development of new computational techniques. Some of the drawbacks to these deep neural networks are that they often only perform well on data sets with large amounts of observed data, they are not well mathematically understood when they will work, and why. They can randomly fail to converge and the learned representations are not well understood what they represent. Other strategies for data representation and feature extraction, such as topic modeling based strategies have also recently progressed. Previously previously criticized for their computational complexity, it is now possible to quickly perform topic modeling on large streaming data sets. Topic models combine data modeling with optimization to learn interpretable and consistent feature structures in data. We introduce a deep non-negative matrix factorization framework capable of producing reliable, interpretable, predictable hierarchical classification of many types of data. Our proposed framework illustrates that it is possible to combine the interpretability and predictability of topic modeling learned representations with some of the power and accuracy of deep neural networks. Furthermore, we uncover a new connection between sparse matrix representations and deep representations by empirically showing that connecting multiple layers with a non-linear function, followed by backpropagation promotes sparsity.

2 Introduction

Deep neural network learns an input-output network composed of multiple layers of representations [Krizhevsky et al., 2012]. In particular, deep convolutional neural networks are the current leaders in image classification, speech recognition, and classification of large data sets; they have obtained state of the art results for classification, even surpassing human level performance [He et al., 2015b], [Amodei et al., 2015], [Le Roux et al., 2015], [Boureau et al., 2010], [LeCun et al., 2015] [Gan et al., 2015],

Address(es) of author(s) should be given

[Flenner et al., 2015]. One of the major drawbacks of the deep learning approach is that the models are not well understood mathematically. For example, there is no known convergence criteria, the true value of hyper-parameters are unknown, the accuracy is unpredictable, it is not known a priori where or why they fail. When neural networks do fail, the failure can be significant, at times misclassifying data with high confidence [Nguyen et al., 2015].However, other strategies for data representation and feature extraction, such as topic modeling based strategies [Blei, 2012], [Lee and Seung, 1999], are well understood [Cichoki et al., 2009], [Rajabi and Ghassemian, 2015]. Topic models combine data modeling with optimization to learn interpretable and consistent features in data [Blei and Lafferty, 2009], [Hoyer, 2004].

We combine a deep architecture, containing multiple layers, nonlinearities, and backpropagation, with the interpretability of topic modeling. This proposed deep non-negative matrix factorization (deep NMF) is capable of producing reliable, interpretable, and predictable hierarchical classification of text, audio and image data. Both the non-negative matrix factorization model and deep models are learned through optimizing an energy function. We demonstrate that it is natural to leverage pooling and backpropagation from deep neural networks and combine them with NMF based representations. First we propose a multilayered NMF that provides a hierarchical topic model. Secondly, we use this multilayered NMF as a model for a deep neural network. This allows us to create a single efficient numerical algorithm that optimizes a deep multilayered NMF model that maintains the generative interpretable nature of NMF at the top layers while simultaneously obtaining state of the art classification accuracy of deep neural networks.

2.1 Basics of Non-negative Matrix Factorization

One linear algebra based topic modeling technique is non-negative matrix factorization (NMF). This method was popularized by Lee and Seung through a series of algorithms [Lee and Seung, 1999], [Leen et al., 2001], [Lee et al., 2010] that can be easily implemented. Given a data matrix X such that $X_{ij} \geq 0$, non-negative matrix factorization finds a data representation by solving the optimization problem

$$\min_{A,S} ||X - AS||_F, \quad \text{such that } A_{ij} \geq 0,\ S_{ij} \geq 0, \tag{1}$$

where $|| \cdot ||_F$ is the Frobenius norm. This optimization provides a generative model of the data through linear non-negative constraints. The data matrix X is factorized into a basis matrix A and corresponding coefficient matrix S.

Minimization in each variable A, S separately is a convex problem, but the joint minimization of both variables is non-convex [Cichoki et al., 2009]. Many NMF algorithms can get stuck in local minima, therefore, the algorithm's success depends on initialization. This problem can often be overcome by providing several random initializations and keeping the factorization that maximizes some performance criteria.

Due to its speed and simplicity, we use the multiplicative update equations, originally derived by Lee and Seung [Lee and Seung, 1999], to optimize Equation 1 . Note that $A \odot B$ represents component wise multiplication, i.e. $(A \odot B)_{ij} = A_{ij}B_{ij}$ and that division is component wise for non-zero entries; the update equations are given by

$$A \leftarrow A \odot \frac{XS^T}{ASS^T}, S \leftarrow S \odot \frac{A^TX}{A^TAS}. \tag{2}$$

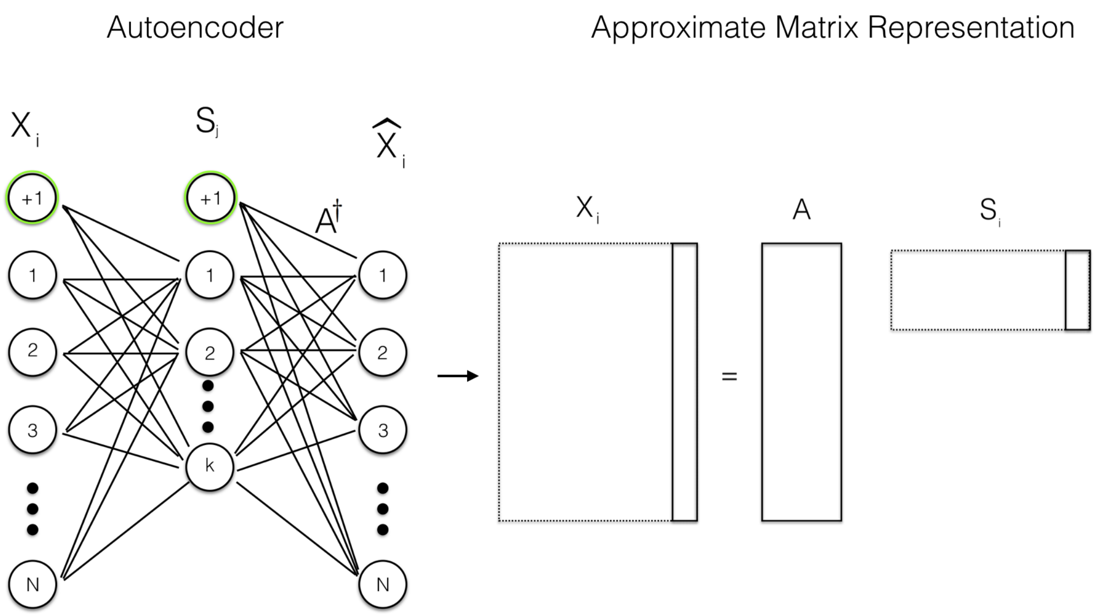


Fig. 1: An analogy can be made between a one layer autoencoder and NMF. However, in the diagram above it should be clear that A in the autoencoder scheme is not mathematically equivalent to the operator A used in NMF. The properties of A in the autoencoder schematic do not have a fixed mathematical interpretation, there is no known regularization to make sense of the output.

2.2 Overview of Deep Learning and Deep Neural Networks

Deep learning and deep neural networks are a rebirth of artificial networks from the 80's inspired by biological nervous systems [LeCun et al., 2015]. Advancements in computing power, the availability of massive amounts of data, and the development of modeling and computational techniques have allowed deep techniques to outperform state of the art results for voice recognition, machine translation, and image classification problems using supervised training techniques [Deng et al., 2013], [Bengio, 2013], [Deng, 2014], [Krizhevsky et al., 2012] on extremely large data sets. The two most powerful changes to modeling and computational techniques were the introduction of multiple hidden layers and backpropagation.

Previously there have been attempts to combine supervised deep neural networks with unsupervised representations. One common way of combining these is using autoencoders. Autoencoders are a class of self-supervised neural networks that learn a data representation which approximates it's own input. Initially, autoencoders were seen as a promising way to initialize deep neural networks. However, autoencoders have failed to produce as high of an accuracy as random initializations [He et al., 2015a].

In this section, we formulate an autoencoder as a matrix factorization to highlight the similarities and differences between these two generative models. Given a set of samples x_n, the autoencoder is a generative algorithm that solves the optimization problem

$$\min_W \sum_{n=1}^{N} ||x_n - f(x_n, W)||,$$

for some norm $||\cdot||$ and some neural network $f(x, W)$ with parameters W. Let $A^\dagger$ be a pseudo-inverse, such as the More-Penrose pseudo-inverse [Moore, 1920], to the equation $X = AS$ then $A^\dagger X = S$. Let s_n be the column of S corresponding to the n^{th} column of X. and the autoencoder optimization problem's norm can be written as

$$||x_n - f(x_n, W)|| = ||x_n - As_n|| = ||x_n - AA^\dagger x_n||.$$

The goal of an autoencoder is to learn the operators A and $A^\dagger$. From the above equation, it is clear that a lossless autoencoder using the above model will only learn the identity. A lossy autoencoder with the above model will learn an approximate identity.

In order to learn more interesting features than the linear model given above, it is typical for an autoencoder to include a nonlinear activation function $g(x)$. Some common choices of are the softplus function [Glorot et al., 2011], the sigmoid function [Cybenko, 1989], [Hornik et al., 1989], and the rectifier [Nair and Hinton, 2010], [LeCun et al., 2015]. The rectifier activation function given by

$$g(z) = max(0, z),$$

is the most interesting for non-negative matrix factorization since it is equivalent to requiring all dictionary weights to be non-negative. More precisely, if we change the autoencoder optimization to

$$||x_n - Ag(A^\dagger x_n)|| \leftarrow ||x_n - AA^\dagger x_n||,$$

then we have included a non-negative constraint on the weights $s_n = A^\dagger x_n$. However, this is still not equivalent to NMF since A is not constrained to have non-negative elements. This is a very important distinction because the columns of A are the basis vectors that give physical meaning to the NMF generative model.

Self-supervised algorithms, such as the autoencoders, often do not learn interesting features. For example, there is nothing preventing an autoencoder from learning the identity function. Along these lines, a stacked autoencoder, capable of learning several layers of a function, was developed. However, the set of stacked autoencoders tended to learn a lossy reconstruction which was either PCA or a close approximation [Baldi and Hornik, 1989], [Chicco et al., 2014] . It was found that a self-supervised algorithm that reconstructs the initial data with high fidelity is unlikely to have learned a representation that is useful for applications involving learned representations such as denoising, localization, or classification. Therefore, it became common to restrict the autoencoder in some way, forcing it to learn a lossy reconstruction of the data rather than the exact input data.The lossy reconstructed data learned by autoencoders was used as a pre-training technique for deep networks. However, these techniques were abandoned when it was discovered that better results, in less time, could be obtained by initializing deep networks through random weight initialization schemes [He et al., 2015a].

2.2.1 Imagenet

Currently, one of the most succesful methods for image classification is Imagenet [Russakovsky et al., 2015]. Deep convolutional neural networks learn a data representation at each layer. Each of the deep convolutional layers has a set of parameters, weights and biases, that are learned through labeling. Alternately, the pooling layer is a fixed function and does not need to learn any parameters. The goal is to determine the parameters that provide good classification performance for new, unseen, data samples. The purpose is not only to find a repeatable representative patterns in the data, but also to perform directed tasks on a large data set, such as classification, object recognition, and denoising [Wang, 2016], [LeCun et al., 2015]. The patterns discovered by deep convolutional neural networks are designed to produce classification labels, but these filters are often not interpretable or physically meaningful, and the learned coefficients are not able to accurately reconstruct the data. This is a direct result of the fact that these deep models are not required to be generative models; they are focused on learnability, not representability.

2.2.2 Limitations of Neural Networks

Deep learning algorithms constructed of hierarchical nested layers, specifically, deep convolutional neural netoworks [He et al., 2015b], [Russakovsky et al., 2015], have recently lead the field in producing state of the art classification results, at times matching or exceeding human classification, for problems related to extremely large data sets. However, despite recent success, there are four main limitations to these methods.

- First, regardless of all the efforts to understand why deep models produce excellent classification results, these approaches are still not well understood [Giryes et al., 2015]. Currently there is no coherent framework for understanding the functionality of each of the layers of deep algorithms [Bruna and Mallat, 2013], which renders construction of an optimal deep neural network architecture for a specific mathematical problem to be more art than science [Szegedy et al., 2015]. If we consider the examples of Figures 1 and 2 which map autoencoder deep generative models to the NMF topic model the mathematical problems become visible.
- Secondly, if the generative model requirement is removed, then the output is not physically interpretable [LeCun et al., 2015].
- Many common problems that can occur during training such as: overfitting, generalization error, and computation time [Bengio, 2013].
- Lastly, deep neural networks work well for large, or very large data sets, but do not perform well on problems with limited observed training data or small data sets.

Contributions of this paper are

- Our deep NMF addresses this issue because the operators in each layer are linear algebra operators. The function and behavior of these operators is well understood.
- Our deep NMF retains the generative model requirement, but does so by exploiting known attributes of the data. The observed data is a mix of multiple source signals. Deconstruction of this data by NMF is a method of blind source separation which unmixes source signals from the observed data. This means we won't learn the identity function, furthermore, although reconstruction will be lossy, we are likely to retain important features in the signal. The results of this deconstruction will be physically interpretable.

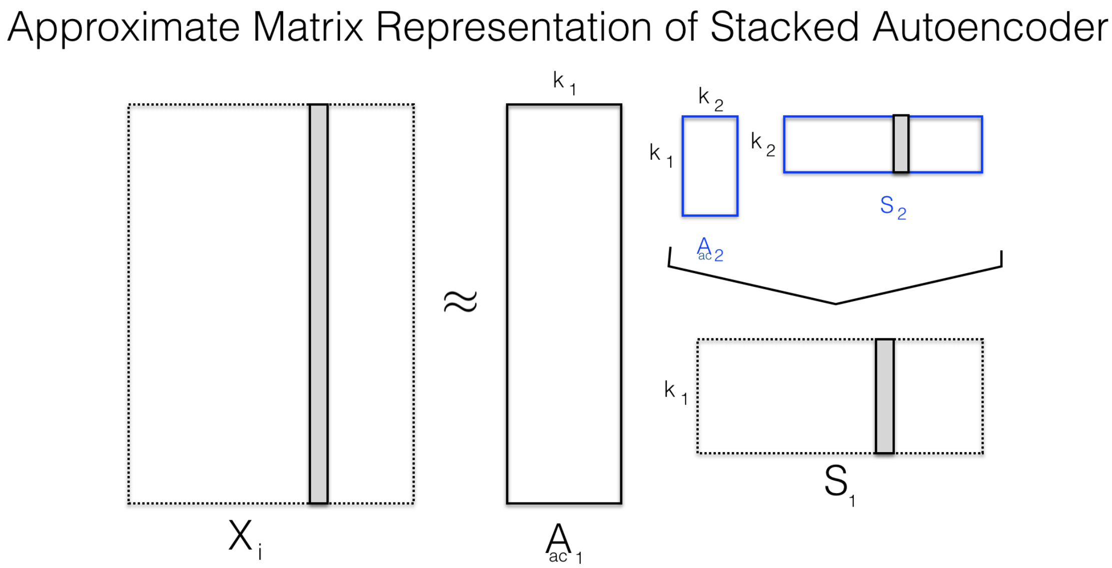


Fig. 2: An analogy can be made between a the stacked autoencoder and multilayered NMF. Again, however, The properties of A in the autoencoder schematic do not have a fixed mathematical interpretation, there is no known regularization to make sense of the output.

- Although the deep NMF learned basis can shrink or grow, we have control over the rank restriction in each layer. We can set common sense rank restrictions given domain information. Control over rank restriction reduces computation time, prevents overfitting and underfitting of the data.
- It is well known that NMF works well on small data sets, in fact, has only recently been scaled to perform well on extremely large data sets. The generative, feature preserving aspects of this model, enable us to perform well on any size data set.

3 Methods

3.1 Deep Networks

Deep networks are compositions of different functions, or network layers, commonly referred to as hidden layers. Deep networks can be used to approximate any L_2 function and separate linear or non-linear data [Hecht-Nielsen, 1989], [Demartines and Hérault, 1997], [Hassoun, 1995]. The most successful deep network, Imagenet, is a convolutional network which consists primarily of two different types of network layers that we reproduce in our optimization framework [Krizhevsky et al., 2012]. Imagenet uses multiple pairs of a dictionary learning layer and a pooling layer. The dictionary learning layer in Imagenet learns a hierarchy of linear operators on the data. The linear operators are typically convolutional operators, which are translation invariant [Long et al., 2015]. In order to be more consistent with non-negative matrix factorization, we only consider neural networks with a set of linear functionals.

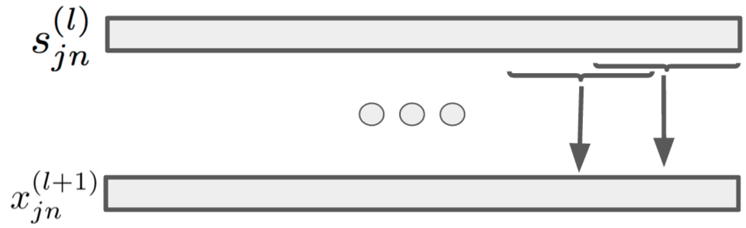


Fig. 3: Illustration of the pooling operator.

Let $\mathbf{x} \in \mathbb{R}^d$ be an input data point. The dictionary layer is a set of functionals $l_k : \mathbb{R}^d \to \mathbb{R}$ such that

$$l_k(\mathbf{x}) = \mathbf{s}_k.$$

By the Riesz representation theorem, we can write the kth dictionary element $l_k(\mathbf{x}) = \langle \mathbf{a}_k^\dagger, \mathbf{x} \rangle$ for weights $\mathbf{a}_k^\dagger \in \mathbb{R}^d$. This means that each layer of a neural network can be written as a matrix multiplication, as seen in Figure 2. The output of all the linear operators is a vector $\mathbf{s} = (s_1, s_2, \ldots, s_K)^T = (\langle \mathbf{a}_1^\dagger, \mathbf{x} \rangle, \ldots, \langle \mathbf{a}_K^\dagger, \mathbf{x} \rangle)^T$. The $\mathbf{a}_k^\dagger$ vectors are the neural network analog to the non-negative dictionary $A = (\mathbf{a}_1^\dagger, \ldots, \mathbf{a}_K^\dagger)$ in NMF.

Neural networks often then apply an activation function $g(z)$. Some common choices are the softplus function, the sigmoid function and the rectifier. Applying the rectifier activation function, given by

$$g(z) = max(0, z),$$

can be minimiced by requiring all dictionary weights to be non-negative.

For $l < D$ the second layer of the network, the max pooling layer, is defined by the set of functions

$$\begin{aligned} p_k(\mathbf{x}) = \max\{x_k \,|\, j - l \leq k \leq j + l,\, k > 0, \\ k < D, 0 \leq j \leq D\}. \end{aligned}$$

We define $p(\mathbf{x}) = (p_1(\mathbf{x}), p_2(\mathbf{x}), \ldots, p_K(\mathbf{x}))$ as the pooling function, see Figure 3. For each row of the matrix S, the pooling operator non-linearly maps a subset of the row to one output value. The Imagenet pooling layer can be constructed with an average pooling function [Oyallon et al., 2013] or the max pooling function. Analysis and experiments on sparse data by [Boureau et al., 2010] indicate that max pooling outperforms average pooling, so we implement max pooling as well.

A final layer is the classification function $g : \mathbb{R}^{d \times N} \to \mathbb{R}^N$. This function maps the final layer to an integer as seen in Figure 7.

A deep neural network is a composition of the network functions with a different parameter set, $A^{(l)}$, when applicable [Jia et al., 2014]. Define $f_l(\mathbf{x}) = f(\mathbf{x}, A^{(l)})$ as the convolution function with parameter set $A^{(l)}$. The output of our neural network with $\mathcal{L}$ layers, denoted as $h(\mathbf{x})$, is the composition of functions.

Note that the output of a pooling layer, p, is the input to a dictionary learning layer f_l.

In order to optimize these parameters on the training data, [Williams and Hinton, 1986] suggests the backpropagation method. Backpropagation depends on calculating derivatives of the pooling function, but the derivative of the max function does not always exist. Where it exists, the derivative of the max function can be written as

$$\frac{\partial \max(\mathbf{x})}{\partial x_k} = \begin{cases} 1 & \text{if } \max(\mathbf{x}) = x_k, \\ 0 & \text{otherwise} \end{cases} .$$

This derivative can be used to define the derivative of $p(\mathbf{x})$ with respect to each of the components of the vector $\mathbf{x}$. A derivation of the derivative of the pooling operator will be presented after the deep NMF section.

3.1.1 Deep Network Backpropagation

The backpropagation algorithm has become the standard algorithm to train the neural networks [LeCun et al., 2012], [Jia et al., 2014]. Given a set of examples $\mathbf{x}_n$ with a corresponding class label $y_n \in \mathbb{Z}$ for each sample, the backpropagation algorithm defines an energy function

$$E = \frac{1}{2}\sum_{n=1}^{N}(y_n - f(\mathbf{x}_n))^2.$$

Let W_l denote the parameters for the function f_l at the l^{th} layer of the network and ∇_{W_l} represents the gradient with respect to these parameters. Using a gradient descent to minimize the energy E updates the parameters according to the rule

$$W_l^{(n+1)} = W_l^{(n)} - \eta \nabla_{W_l} E.$$

The variable η is often called the *learning rate*. An energy is defined based on the output of the network and the parameters for the l^{th} layer. This energy is then updated through the l gradients that back-propagate from the output to the l^{th} layer, hence the name backpropagation.

3.1.2 Semi-Supervised NMF Multiplicative Update Equations

The semi-supervised NMF algorithm requires a label matrix Y. Let $Y \in \mathbb{R}^{N\times K}$ be a class matrix where for each sample n then $Y_{nk} = 1$ if $\mathbf{x}_n$ is in class k and $Y_{nk} = 0$ otherwise. Next, approximate the known label matrix Y by finding a separating hyper plane defined by a new operator B such that $||Y - BS||^2$. Finally we can introduce binary indicator matrices W_{nk} and L_{nk} to model missing data and known data labels respectively as

$$W_{ij} = \begin{cases} 1, & \text{if } \mathbf{x}_{ij} \text{ is observed} \\ 0, & \text{if } \mathbf{x}_{ij} \text{ is unobserved,} \end{cases} \quad \text{and}$$

$$[L]_{:,j} = \begin{cases} 1_k, & \text{if label } \mathbf{x}_j \text{ is known} \\ 0, & \text{otherwise.} \end{cases}$$

The NMF energy function can be rewritten as

$$E(A,B,S) = \frac{1}{2}||W \odot (X - AS)||_F^2 + \frac{\lambda}{2}||L \odot (Y - BS)||_F^2,$$

and we include the constraints $A_{ij} \geq 0$, $B_{ij} \geq 0$ and $S_{ij} \geq 0$. The term λ is used to weight the importance of the labeling. If $\lambda = 0$ then the energy functional is equivalent to the unsupervised equation, a small λ can be useful if some of the data is mislabeled, and a large λ emphasizes the labels, see [Lee et al., 2010] for more details.

3.2 Deep NMF

Our deep NMF model consists of multilayered NMF combined with a pooling layer followed by backpropagation. The multilayered NMF consists of nested NMF decompositions into $\mathcal{L}$ layers. After the primary data observations are deconstructed, each subsequent data layer is acted upon by a pooling function prior to each subsequent NMF decomposition. The last, or $\mathcal{L}$, layer of multilayered NMF is decomposed by semi-supervised NMF, instead of NMF. The semi-supervised step is used to create a label learning energy functional which softens the invertibility requirement; backpropagation acts on the energy functional to learn a better set of basis coefficients. Backpropagation on the energy functional is what is known as the learning step of the algorithm. This model will be analyzed in three parts: multilayered NMF , supervised multilayered NMF with pooling, and deep NMF with backpropagation.

3.2.1 Deep NMF Backpropagation

Backpropagation is a way to optimize learning step of a deep algorithm. Backpropagation algorithms require an energy functional based on the output of a neural network and optimize that energy functional. The neural networks we are considering can be written as composition of (convolutional) representation layers and pooling layers with the final layer a classification layer. The neural network can be written in the form

$$h(\mathbf{x}) = g(f(p(f(\dots f(\mathbf{x}, A^{(0)})\dots, A^{(\mathcal{L}-1)})), A^{(\mathcal{L})}).$$

It is useful to define the output of the neural network up to layer l after the dictionary learning and pooling steps respectively as

$$d_l(\mathbf{x}) = f(p(f(\dots f(\mathbf{x}, A^0)\dots, A^{(l-1)})), A^{(l)}),$$
$$q_l(\mathbf{x}) = p(f(\dots f(\mathbf{x}, A^0)\dots, A^{(l-1)})), A^{(l)}).$$

Let $z(n)$ correspond to the class of the n^{th} column. Consider the energy functional, E_{NN}, the neural network energy

$$E_{NN}(\{\mathbf{x}_n\}_{n=1}^N, z) = \frac{1}{2}\sum_{n=1}^{N}(z(n) - h(\mathbf{x}_n))^2.$$

The backpropagation algorithm learns the parameters one layer at a time and is based on gradient descent of this energy. There are no parameters to learn for the pooling operator.

3.2.2 Multilayered NMF

Consider the independent nested set of NMF decompositions. Let $X^{(0)}$ be the original data observations. Each column in the spectrogram is a document, the sum of all the documents is called the corpus. Let the corpus, $X^{(0)}$, be the first input. The first NMF decomposition obtains

$$X^{(0)} \approx A^{(0)} S^{(0)},$$

where $A^{(0)}$ are a set of topics, or basis vectors, and $S^{(0)}$ are the topic weights, or basis coefficients. Next the basis coefficients $S^{(0)}$ become the new input. The second layer deconstructs $S^{(0)}$ to obtain a subtopics and subtopic weights,

$$S^{(0)} \approx A^{(1)} S^{(1)}.$$

The two layer nested decomposition can be rewritten as,

$$X^{(0)} \approx A^{(0)}(A^{(1)} S^{(1)}),$$

shown graphically in Figure 4.

This process can continue in order to learn as many layers as is desired. The nested decomposition for $\mathcal{L}$ layers can be found by minimizing

$$||X^{(0)} - A^{(0)}(A^{(1)}(...(A^{(\mathcal{L})} S^{(\mathcal{L})})))||.$$

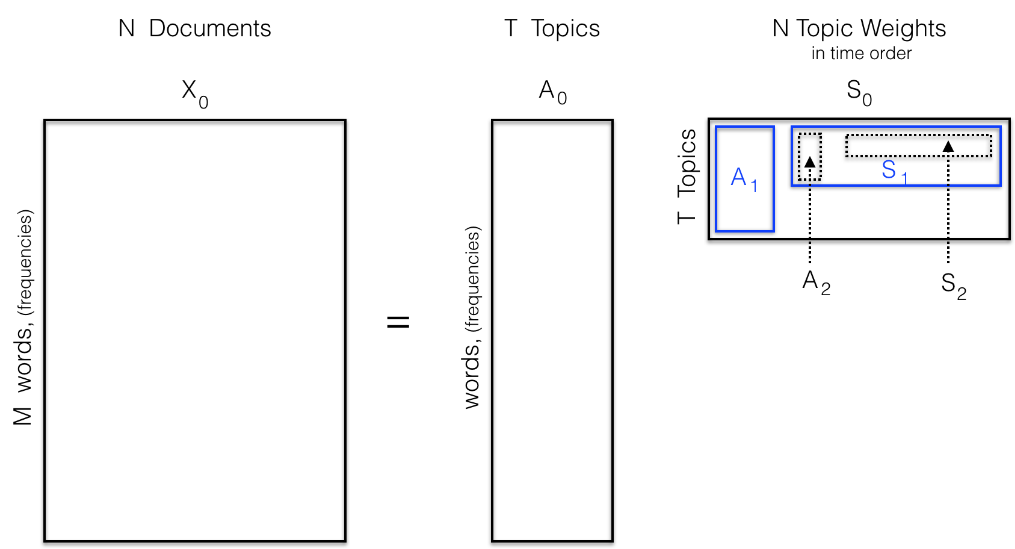


Fig. 4: Our deep NMF model without pooling is a hierarchical representation of the data at the zeroth layer and then the weights and lower levels. This is the nested decomposition of the S coefficient matrices, $X^{(0)} \approx A^{(0)}(A^{(1)}(A^{(2)} S^{(2)}))$.

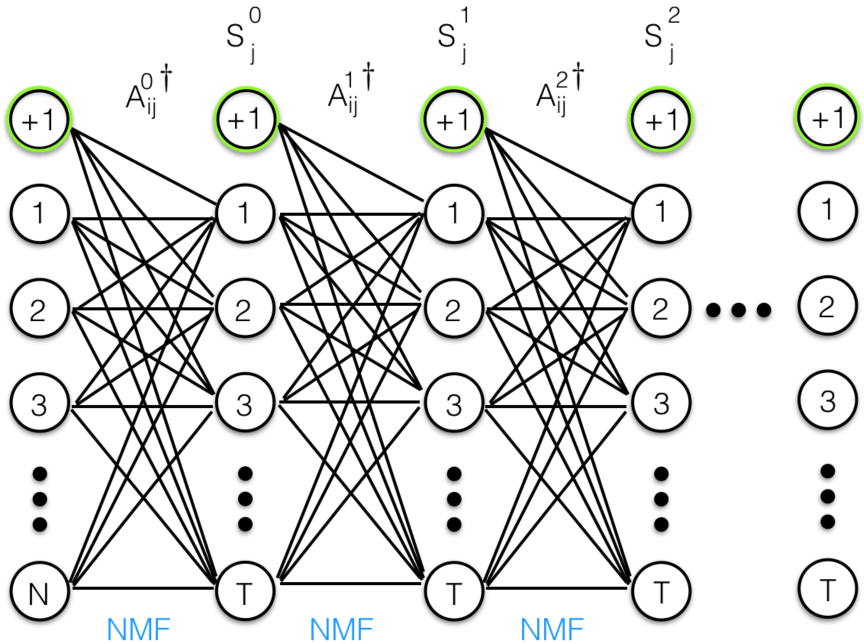


Fig. 5: The nested NMF model without pooling is a hierarchical representation of the data.

Now, let us define an operator on X. The operator defined as $A^\dagger = (A^T A)^{-1} A^T$ minimizes the ℓ_2 norm between X and AS such that $A^\dagger X = S$. This operator is called the Moore-Penrose pseudoinverse [Ben-Israel and Greville, 2003], [Moore, 1920] and it is used in figure 5 for the the multilayered NMF diagram.

3.2.3 Supervised Multilayered NMF with Pooling (Forward Propagation)

A pooling step is commonly found in Neural networks [Scherer et al., 2010], [Szegedy et al., 2015]. The reason a pooling step improves neural network results is not well understood [Bruna and Mallat, 2013]. However, it is hypothesized that the purpose of a pooling step in neural networks is to reduce the spatial size of the representation to control overfitting, to create robustness to small variations and to introduce nonlinearity into the system. The most common form of pooling is to replace a neighborhood of data points with their maximum value; this is called max pooling. Inclusion of a pooling step in the deep NMF model allows us to mimic a deep neural network and investigate the impact of this operation.

We include a pooling step in our deep NMF model by placing a pooling layer after NMF decomposition in each layer. The first NMF decomposition is the original data matrix, followed by max pooling step on the weight matrix, S. The max pooling step is a window of fixed size that is moved across the data matrix columns. Every pixel in the window is replaced with the maximum pixel value found in the window; the resulting data matrix is called $p(S)$. Once the max pooling step is performed the $p(S)$ matrix is decomposed by NMF.

Algorithm 1: Nested NMF Forward Propagation

Data: $\mathbf{X} \in \mathbb{R}^{m \times n}$ from a collection of n documents.
The number of layers L.
The number of columns to pool across, poolsize.
Result: A nonnegative matrix decomposition at each layer l.
Initialize using forward propagation
for *l to l-1* **do**
  while *not converged* **do**

$$A^{(l)} \leftarrow A^{(l)} \odot \frac{X^{(l)}(S^{(l)})^T}{A^{(l)}S^{(l)}(S^{(l)})^T},$$

$$S^{(l)} \leftarrow S^{(l)} \odot \frac{[A^{(l)}]^T X^{(l)}}{[A^{(l)}]^T A^{(l)} S^{(l)}},$$

  end
  $X^{(l+1)} \leftarrow S^{(l)}$
  $l \leftarrow l + 1$
end

Fig. 6: The nested NMF forward propagation algorithm.

It is important to note that the pooling layer typically operates along the direction of a symmetry group in the data. See [Bruna and Mallat, 2013] for more information. Consider the NMF decomposition of the data

$$\mathbf{x}_n \approx A\mathbf{s}_n.$$

The pooling layer takes as input the rows of the weight matrix S_l. We use superscripts to represent row vectors of S, thus $\mathbf{s}^k \in \mathbb{R}^M$ is the k^{th} row of matrix S.

Let S be the NMF matrix from layer l. Define the rows of the data matrix X for the next layer as the pooled weights from layer $l-1$, or

$$\mathbf{x}^{k+1} = p(\mathbf{s}^k).$$

We will also use the notation $X = p(S)$ to represent the matrix where each row of the matrix is the output of the pooling operator defined above.

The last pooling layer in the algorithm is followed by supervised NMF. This is necessary because we match labels only on the last layer, as in a neural network. The zeroth layer depends on the input data, while the last $\mathcal{L}$ layer is used for classification.

This completes the forward propagation of the deep NMF model, graphically shown in Figure 7 and outlined in 8.

The equation which describes the deep NMF forward propagation with pooling is an energy functional. The energy functional exploits the NMF generative model in order to describe the error between the input data and reconstruction at each layer. The energy functional written here relies

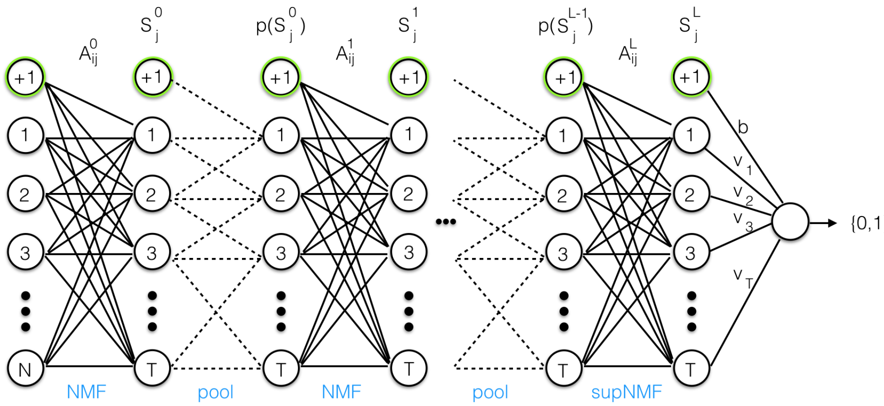


Fig. 7: A graphical representation of the deep topic model. The last layer is a linear classifier that can be learned from knowing the training data.

Algorithm 2: Supervised NMF Forward Propagation With Pooling

Data: $\mathbf{X} \in \mathbb{R}^{m \times n}$ from a collection of n documents.
The number of layers L.
The number of columns to pool across, poolsize.
Result: A nonnegative matrix decomposition at each layer l.
Initialize using forward propagation
while *not converged* **do**

$$
\begin{aligned}
A^{(l)} &\leftarrow A^{(l)} \odot \frac{X^{(l)}(S^{(l)})^T}{A^{(l)}S^{(l)}(S^{(l)})^T}, \\
S^{(l)} &\leftarrow S^{(l)} \odot \frac{[A^{(l)}]^T X^{(l)}}{[A^{(l)}]^T A^{(l)} S^{(l)}}, \\
X^{(l+1)} &\leftarrow p(S^{(l)}) \\
S^{(L)} &\leftarrow S^{(L)} \odot \frac{[A^{(L)}]^T X^{(L)} + B^T Y}{[A^{(L)}]^T A^{(L)} S + B^T B S^{(L)}}, \\
B &\leftarrow B \odot \frac{Y(S^{(L)})^T}{BS^{(L)}(S^{(L)})^T}.
\end{aligned}
$$

end

Fig. 8: The supervised NMF forward propagation algorithm with pooling. This algorithm deconstructs the final input layer, $p(S^{(\mathcal{L}-1)})$, with a semi-supervised NMF step.

Algorithm 3: Backpropagation

Data: $\mathbf{X} \in \mathbb{R}^{m \times n}$ from a collection of n documents.
The number of layers L.
The number of columns to pool across, poolsize.
Result: A nonnegative matrix decomposition at each layer l.
Backpropagation
while *not converged* **do**

$$
\begin{aligned}
A^{(l)} &\leftarrow A^{(l)} \odot \frac{X^{(l)}(S^{(l)})^T}{A^{(l)}S^{(l)}(S^{(l)})^T}, \\
J_p(S^{(l+1)}) &\leftarrow \text{Calc Jacobian } (S^{(l+1)}) \\
S^{(l)} &\leftarrow S^{(l)} \odot \frac{[A^{(l)}]^T X^{(l)} + J_p(S^{(l+1)})A^{(l+1)}S^{(l+1)}}{[A^{(l)}]^T A^{(l)} S^{(l)} + J_p(S^{(l+1)})X^{(l+1)}}, \\
X^{(l+1)} &\leftarrow p(S^{(l)}) \\
S^{(L)} &\leftarrow S^{(L)} \odot \frac{[A^{(L)}]^T X^{(L)} + B^T Y}{[A^{(L)}]^T A^{(L)} S + B^T B S^{(L)}}, \\
B &\leftarrow B \odot \frac{Y(S^{(L)})^T}{BS^{(L)}(S^{(L)})^T}.
\end{aligned}
$$

end

Fig. 9: The deep NMF backpropagation algorithm.

on the Frobenius norm, but it is possible to construct energy functionals using other norms. The semi-supervised Deep NMF energy functional takes the form

$$
\begin{aligned}
E(X^{(l)}, A^{(l)}, S^{(l)}, B) &= \frac{1}{2}\sum_{l=0}^{\mathcal{L}} ||W \odot (X - AS)||_F^2 + \frac{\lambda}{2}||L \odot (Y - BS^{(\mathcal{L})})||_F^2, \\
&= \frac{1}{2}||W \odot (X^{(0)} - A^{(0)}S^{(0)})||_F^2 + \frac{1}{2}\sum_{l=1}^{\mathcal{L}} ||W \odot \left(p(S^{(l-1)}) - A^{(l)}S^{(l)}\right)||_F^2 + \frac{\lambda}{2}||L \odot (Y - BS^{(\mathcal{L})})||_F^2 \\
&= \frac{1}{2}||W \odot (X^{(0)} - A^{(0)}S^{(0)})||_F^2 + \frac{1}{2}\sum_{l=0}^{\mathcal{L}-1} ||W \odot \left(p(S^{(l)}) - A^{(l+1)}S^{(l+1)}\right)||_F^2 + \frac{\lambda}{2}||L \odot (Y - BS^{(\mathcal{L})})||_F^2.
\end{aligned}
$$

3.2.4 Deep NMF with Backpropagation

The final algorithm is backpropagation, or the learning step. This algorithm optimizes the weights, or basis coefficients, in order to minimize the energy functional found through forward propagation. For the deep NMF model, recall the energy functional

$$
\begin{aligned}
E(X^{(l)}, A^{(l)}, S^{(l)}, B) = {} & \frac{1}{2}||W \odot (X^{(0)} - A^{(0)}S^{(0)})||_F^2 + \frac{1}{2}\sum_{l=0}^{\mathcal{L}-1} ||W \odot \Big(p(S^{(l)}) - A^{(l+1)}S^{(l+1)}\Big)||_F^2 \\
& + \frac{\lambda}{2}||L \odot (Y - BS^{(\mathcal{L})})||_F^2.
\end{aligned}
$$

Recall that $p(S^{(l)}) = X^{l+1}$, we use this substitution when calculating the gradient of the energy functional.

The gradient with respect to the $A^{(l)}$ matrix is the same as the original supervised NMF algorithm and is given by the Jacobian partial derivative of the NMF energy functional E such that

$$
\begin{aligned}
J_E^{A^{(l)}}(A^{(l)}) &= -\Big(W \odot (X^{(l)} - A^{(l)}S^{(l)})\Big)(S^{(l)})^T, \\
&= -(W \odot X^{(l)})(S^{(l)})^T + (W \odot A^{(l)}S^{(l)})(S^{(l)})^T.
\end{aligned}
$$

The Jacobian derivative with respect to $S^{(l)}$ is more difficult since the layer $l+1$ depends on $S^{(l)}$ through the pooling step. The Jacobian can be found in the appendix.

The last layer $\mathcal{L}$ will be different from the other l layers. This is because the last layer introduces the classification labeling matrices L, Y and B. The last layer of the energy functional will take the form

$$
E^{(\mathcal{L})} = \frac{1}{2}||W \odot \Big(p(S^{(\mathcal{L}-1)}) - A^{(\mathcal{L})}S^{(\mathcal{L})}\Big)||_F^2 + \frac{\lambda}{2}||L \odot (Y - BS^{(\mathcal{L})})||_F^2.
$$

The Jacobian derivative with respect to B, which only occurs in the final layer $\mathcal{L}$ is

$$
\begin{aligned}
J_E^B(B) &= \lambda\Big(L \odot (Y - BS^{(\mathcal{L})})\Big)[S^{(\mathcal{L})}]^T \\
&= \lambda\Big(L \odot Y\Big)[S^{(\mathcal{L})}]^T - \lambda\Big(L \odot BS^{(\mathcal{L})}\Big)[S^{(\mathcal{L})}]^T.
\end{aligned}
$$

It does not come as a surprise that the $\mathcal{L}$ layer update equations work out to be the same as the supervised NMF algorithm update equations. The intermediate layers $S^{(l)}$ are different. The intermediate layers do not update through labeling, but rather through the Jacobian. Given the Jacobians and following the discussion above , we obtain the multiplicative update equations for deep NMF as

$$
\begin{aligned}
A^{(l)} &\leftarrow A^{(l)} \odot \frac{[W \odot X^{(l)}](S^{(l)})^T}{[W \odot A^{(l)}S^{(l)}](S^{(l)})^T}, \\
S^{(l)} &\leftarrow S^{(l)} \odot \frac{(A^{(l)})^T\Big[W \odot X^{(l)}\Big] + \Big[W \odot A^{(l+1)}S^{(l+1)}\Big][J_p(S^{(l)})]^T}{(A^{(l)})^T\Big[W \odot A^{(l)}S^{(l)}\Big] + \Big[W \odot X^{(l+1)}\Big][J_p(S^{(l)})]^T}, \\
S^{(\mathcal{L})} &\leftarrow S^{(\mathcal{L})} \odot \frac{(A^{(\mathcal{L})})^T\Big[W \odot X^{(\mathcal{L})}\Big] + \lambda B^T\Big[L \odot Y\Big]}{(A^{(\mathcal{L})})^T\Big[W \odot A^{(\mathcal{L})}S^{(\mathcal{L})}\Big] + \lambda B^T\Big[L \odot BS^{(\mathcal{L})}\Big]}, \\
B &\leftarrow B \odot \frac{\Big[L \odot Y\Big](S^{(\mathcal{L})})^T}{\Big[L \odot BS^{(\mathcal{L})}\Big](S^{(\mathcal{L})})^T}.
\end{aligned} \tag{3}
$$

Architecture	Convolutional Deep NN	NMF	Deep NMF
Representation	A	A	A
Activation Function	ReLu: $g(z) = max(z, 0)$	non-negative restriction	non-negative restriction
f(x,w)	$Ag(A^{\dagger}x_n)$	$gAgA^{\dagger}x_n$	$gAgA^{\dagger}x_n$ multilayered
loss function	$\|\|z - f(x, w)\|\|$	$\|\|X - AS\|\|$	$\sum_{\ell} \|\|X^{(\ell)} - A^{(\ell)}S^{(\ell)}\|\|$
Layer Model	additive network weights	-	matrix multiplication
Nonlinearity	pooling or ReLu	none	pooling
Representation	convolutional	multiplicative	multiplicative
Layers	Multilayered Network	one	Multilayered NMF
Optimization	backpropagation	multiplicative updates	backpropagation & multiplicative updates
Training Time	slow	fastest	faster

4 Results and Discussion

4.1 Text Data Preprocessing

The corpus was processed using the Bag of Words model. The first step was to select a fixed number of classes; these classes were used to label the known data. Then, each text document was broken into paragraphs and each paragraph was broken into a histogram of words and word frequencies. These histograms were compiled into a matrix of word frequencies representing the entire corpus, every row contains a distinct word and each column is a paragraph from a specific text document. Next, stop words were removed from the corpus. Stop words are words that do not contribute to the meaning of the content of the corpus, such as articles and pronouns. Next, we calculate the term frequency inverse document frequency statistic , TF-IDF [Salton et al., 1975], which is used to de-emphasized words that are common among all documents, while emphasizing words found with high frequency in a specific document. The final step is to label the training set using the predetermined classes. In this particular case, labels were known for the entire text corpus, so we randomly selected 1% of the data from each of the classes to label as known. The resulting word frequency matrix, with 1% of the data labeled, is the input for the deep NMF algorithm.

4.2 Experiments

The text corpus was deconstructed by the deep NMF algorithm in a series of experiments. The NMF rank restriction was fixed to 20. The training set was created from 1% of the text corpus. Next, the deep NMF algorithm was run for two and three layer cases for each of the pool window sizes from the set $\{0, 3, 5, 7\}$. The classification results from these experiments were compared to the result of the semi-supervised NMF, SSNMF, algorithm using identical rank restriction and training parameters.

Both deep NMF and SSNMF contain a supervision layer. Recall, that the multiplicative update equations trade off between labeling by minimizing $||Y - BS||$ and data reconstruction by minimizing $||X - AS||$. In the multiplicative update equations, the B matrix is created in the last layer and is used for class labeling. Therefore, the density of the B matrix indicates the number of basis elements necessary for classification. The deep NMF algorithm found a sparser basis than NMF, SSNMF, and multilayered NMF. The only two differences between deep NMF and multilayered NMF are the pooling and backpropagation steps. Pooling and backpropagation are encouraging sparsity, as seen in figure 10.

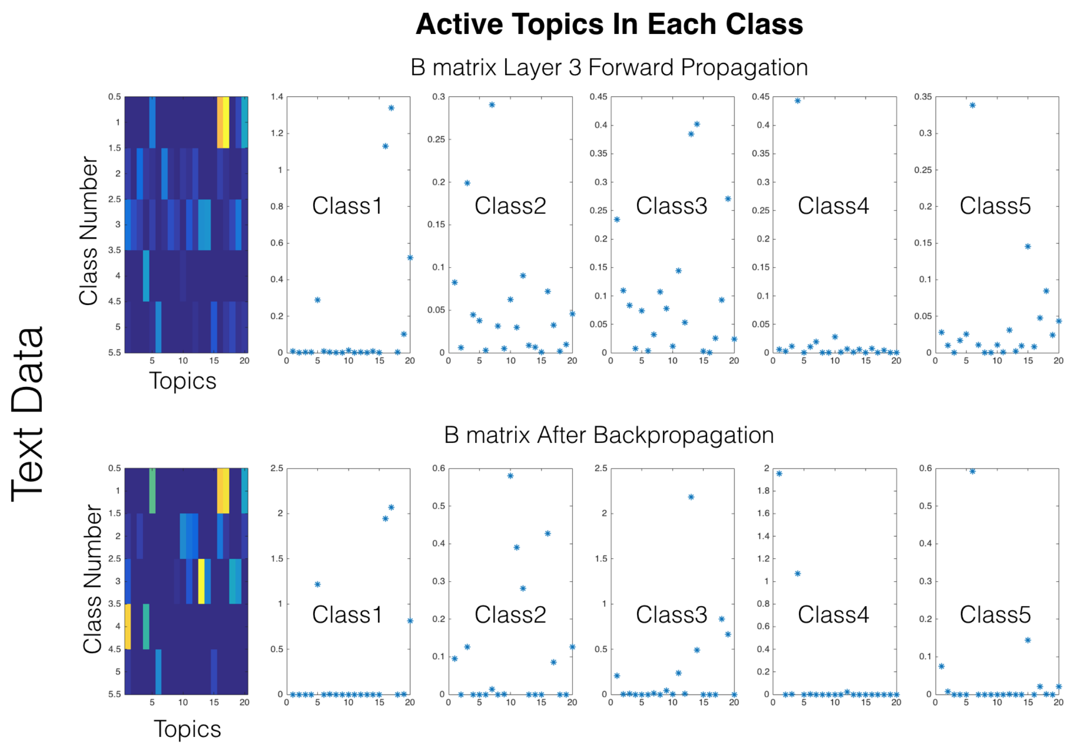


Fig. 10: The B matrix is a result of the multiplicative update rules derived in Equation 3. The second row shows the B matrix becomes sparser after backpropagation.

The forward propagation algorithms, multilayered NMF, learn a set of topics through the first layer NMF decomposition. The topic list does not change until backpropagation. Consider the example of the philosophy class initially deconstructed into 6 topics in layer 1 deconstruction shown in figure ??. However, after backpropagation, a new dictionary and basis weights are learned which allow us to reconstruct the original philosophy class using only two final topics and updated weights. The philosophy class is described by 6 initial topics which are condensed into 2 new final topics after backpropagation. The topic consolidation found was consistent with the B matrix results which show that deep NMF learns the sparsest basis.

Unsupervised learning is performed by classification using clustering. One way to assess the algorithmic result is to analyze cluster purity [Handl et al., 2005]. Purity is an external evaluation method used to rate the homogeneity of clustered data. Note that while purity is related to classification, it is not classification. Specifically, it is possible for purity to be high and classification to be low, or for purity to be low and classification high. The purity results of our clustered data were very similar to the correct classification rate of that data shown in table 12 next to the classification rates.

Finally, classification rates of SSNMF, multilayered NMF and deep NMF were compared. Our deep NMF produced higher classification rates than SSNMF or multilayered NMF. The one layer SSNMF case obtained a classification rate of 46%. It is not possible to perform pooling or backpropagation on a one layer algorithm, so these steps were omitted. The classification results are given in figure 12.

4.3 Insights into Neural Networks

In addition to classification, euclidean distance and angle changes within layers [Giryes et al., 2015] can be used to evaluate the learned representation found by a neural network. If we follow points through deep layers, points within the same class

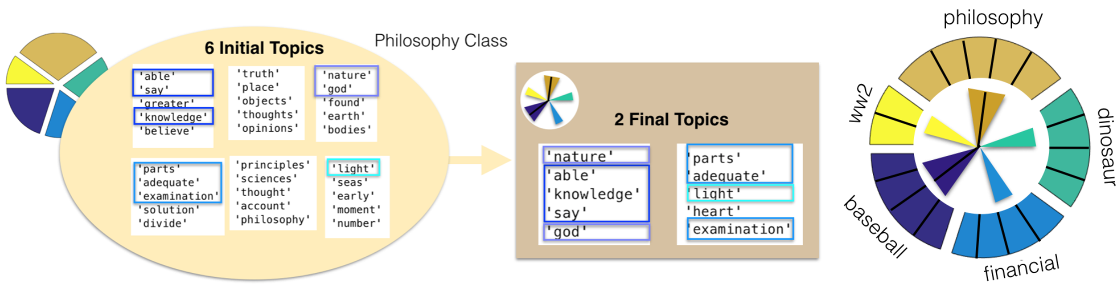
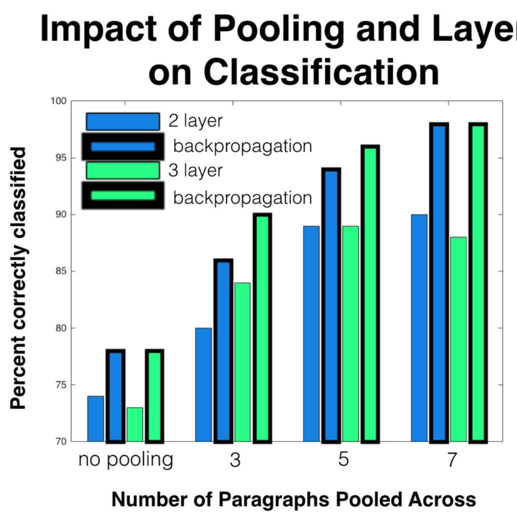


Fig. 11: The outer circle is constructed of 5 classes decomposed into an overcomplete basis of 20 initial topics. Deep NMF, after backpropagation, learns a new representation of both dictionary and weights. The inner circle shows the new restricted overcomplete basis, which reconstructs the original 5 classes into 7 learned final topics and updated weights.



Purity Score

Pooling Across # Documents	Multi - Layer NMF 1 Layer	2 Layer	3 Layer	Deep NMF 2 Layer	3 Layer
0	.46	.74	.76	.78	.78
3	N/A	.79	.85	.86	.90
5	N/A	.86	.83	.94	.96
7	N/A	.88	.95	.98	.98

Fig. 12: This histogram shows the impact of 2 and 3 layers with varying pool size on classification. These results were obtained by randomnly training on 1% of the data with a rank restriciton of 20. These classification rates were compared with the the single layer SSNMF algorithm given the same rank and training parameters. The classification rate of single layer SSNMF was 46%.

are expected to remain close together while points from different classes are expected to move apart. There are two criteria that must be satisfied for this premise to hold: first, the input data must lie on the surface of a sphere and the activation function is a rectified linear unit (ReLU). Although deep NMF does not use a ReLU activation function, it deconstructs a positive corpus into two non-negative matrices which provide the same non-negative initial restriction as the ReLU activation function; the outputs remain approximately on the surface of a sphere. Therefore, the input text data under the deep NMF model satisfy both conditions.

For each layer, a sample of points from a distinct class can be represented by the feature vectors $S^{(l)}$. The angular distance between these feature vectors as the representation becomes deep will change. The maximum angle of separation is 90 degrees. Let $S_a^{(l)}$ be a feature vector from class a and $S_a^{(l+1)}$ be the feature vector at the next layer. Using the

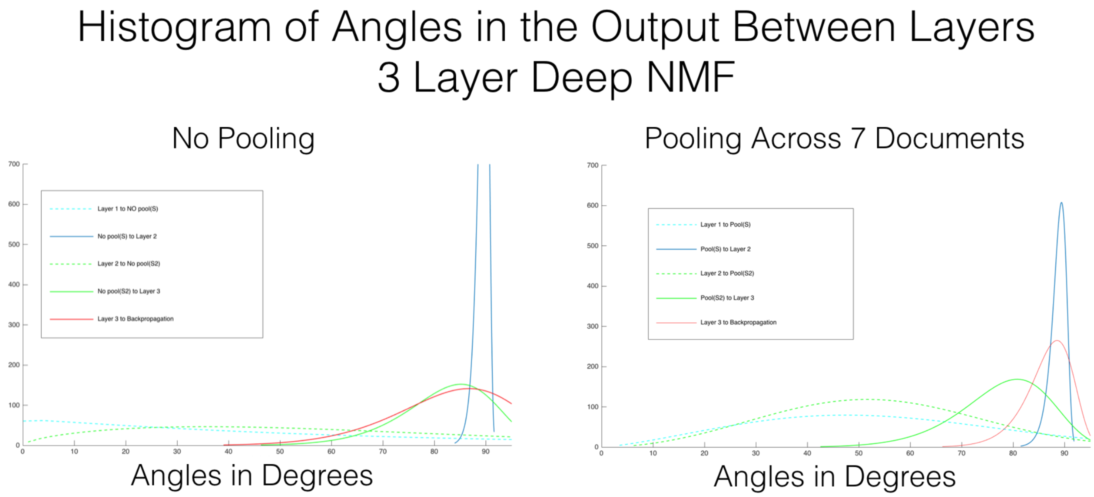


Fig. 13: Histogram of angles in the output between layers, comparing class 1 to the other classes. Note that without pooling the angle separation, through multiple layers, does not occurr.

polarization identity, we can define the angle between two feature vectors as

$$\theta_a^{(l)} = \text{acos}\left(\frac{||S_a^{(l)} + S_a^{(l+1)}|| - ||S_a^{(l)} - S_a^{(l+1)}||}{||S_a^{(l+1)}||\,||S_a^{(l+1)}||}\right).$$

Geometrically, if the minimum angle between the input vectors and output vectors of different classes increases then the sets have an increase in separation. Let $S^{(l)}$ be a feature vector at level l. We calculate the histograms for $H(l, a)$ and $H(l+1, a)$. If the histogram is shifted away from zero, then the class is further separated, indicating the feature vectors are from different classes. If the angle between classes shrinks, the feature vectors are from the same class. The ratio of the angle change between the last layer and the first layer can be used to determine how the points are moving through the layers. If the ratio is less than 1, the points are from the same class. If the ratio is greater than or equal to 1, then the points are from different classes. The deep NMF was able to correctly classify points from within class and between classes in the expected way, shown in figure 14

We compare the input and output angles at each layer for deep NMF with pooling and without pooling. Figure 14 shows that without pooling the representation does not retain important within class information shown in figure 14. Pooling is adding stability as the number of layers increases; this can be seen in Figure 13.

The deep NMF model was judged using standard criteria from deep neural network evaluation which compares points between classes and points from the same class. The NMF portion of the algorithm prevents over fitting, however, it is the pooling layer which creates feature stability, and backpropagation allows us to find the features that determine the classification labels of interest.We were able to show that in general, points from the same classes remain close, while points from different classes are spread apart. We were also able to show that in order to obtain proper distance between points that the inclusion of a non-linear pooling layer was necessary.

5 Conclusion

This paper combined ideas from NMF and deep neural networks, such as the Imagenet model [Krizhevsky et al., 2012], into a deep NMF model. Our novel algorithm combines backpropagation step and the multiplicative update equations to construct a unique set of deep NMF update equations. Two main areas were investigated in this work. First, the performance of deep NMF topic model was compared to NMF, SSNMF, and multilayered NMF and we showed that Deep NMF learned a sparser basis and classifies better than NMF, SSNMF and multilayered NMF.

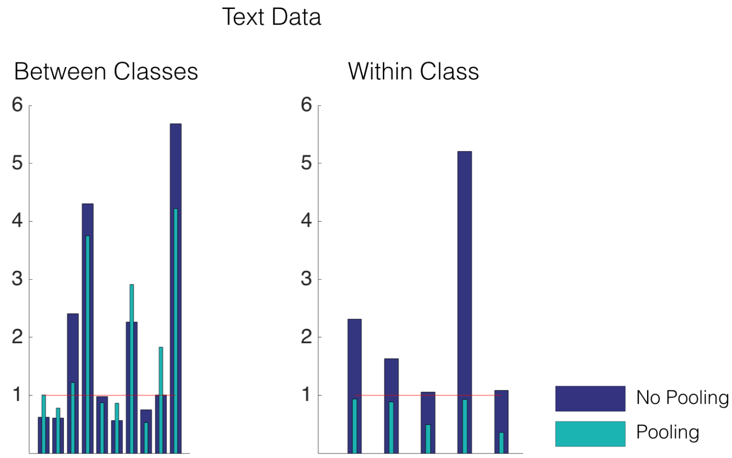


Fig. 14: As we move through the layers points from the same class should remain close while points from different classes should spread apart. The angles between the layers are used to find a distance. If the distance is greater than one, the points are spreading apart. If the distance is less than one the points remain close. The figure on the left reveals the distance between classes. In the figure on the left it is desireable to obtain a distance greater than or equal to one indicating points from different classes are spread apart. The graph on the right reveals the distance between points from the same class. In the figure on the right it is desirable to find a normed distance that is less than one, indicating the points from the same class remain close.

Second, the structure of the deep NMF algorithm, modeled after a DNN, was used to gain insight into neural networks. We demonstrated that classic NMF algorithms can produce the quality of results only seen in deep neural networks. In particular, using our deep NMF we showed that points from the same class remain close while points from different classes become spread apart as the layers increase. We also compared a subset of neural network algorithms with a generative restraint to that of NMF, and this generative constraint produces physically interpretable output at each layer. However, not all generative models learn interesting features from the data, and the implementation of the generative constraint matters. The NMF generative modeling constraint contrasts with autoencoder models, the neural network generative models, which do not retain features of interest despite this restraint. The way in which NMF disentangles source signals from observed data enable deep NMF to learn and retain features of interest in the data. Additionally, this generative constraint allows NMF to perform well on data sets of any size, whereas neural networks only perform well on extremely large data sets. Another common problem that can occur during training of a neural network is overfitting or underfitting the data. The deep NMF model allows us to control the rank of the data deconstruction at each layer, giving us a way to avoid this issue. Finally, we do not know the mathematical properties of DNN optimization algorithm at each layer, it is not clear when a DNN will learn an operator or when it will learn an ill defined mapping. In comparison, for deep NMF, the operators in each layer are linear algebra operators and the function and behavior of these operators is well understood. Knowledge of the specific algebraic operator at each layer allows us to exploit domain information from the data.

Future work involves investigation of deep NMF with specific NMF algorithms to exploit domain information, for example, convolutional NMF, overlapping NMF, sparsity, smoothness of components, semi-blind source separation, or NMF tensor decompositions and factorizations [?], [Cichoki et al., 2009]. There are many implementations of NMF algorithms; relaxed forms yield different but consistent and predictable results. It makes sense to choose a specific NMF algorithm in relation to a data set, knowledge of domain information, and the specific task such as classification or denoising.

References

[Amodei et al., 2015] Amodei, D., Anubhai, R., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Chen, J., Chrzanowski, M., Coates, A., Diamos, G., et al. (2015). Deep speech 2: End-to-end speech recognition in english and mandarin. *arXiv preprint arXiv:1512.02595*.

[Baldi and Hornik, 1989] Baldi, P. and Hornik, K. (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58.

[Ben-Israel and Greville, 2003] Ben-Israel, A. and Greville, T. N. (2003). *Generalized inverses: theory and applications*, volume 15. Springer Science & Business Media.

[Bengio, 2013] Bengio, Y. (2013). Deep learning of representations: Looking forward. In *Statistical language and speech processing*, pages 1–37. Springer.

[Blei, 2012] Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, 55(4):77–84.

[Blei and Lafferty, 2009] Blei, D. M. and Lafferty, J. D. (2009). Topic models. *Text mining: classification, clustering, and applications*, 10(71):34.

[Boureau et al., 2010] Boureau, Y.-L., Ponce, J., and LeCun, Y. (2010). A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118.

[Bruna and Mallat, 2013] Bruna, J. and Mallat, S. (2013). Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886.

[Chicco et al., 2014] Chicco, D., Sadowski, P., and Baldi, P. (2014). Deep autoencoder neural networks for gene ontology annotation predictions. In *Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 533–540. ACM.

[Cichoki et al., 2009] Cichoki, A., Zdunek, R., Phan, A., and Amari, S. (2009). Nonnegative matrix and tensor factorization.

[Cybenko, 1989] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.

[Demartines and Hérault, 1997] Demartines, P. and Hérault, J. (1997). Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets. *IEEE Transactions on neural networks*, 8(1):148–154.

[Deng, 2014] Deng, L. (2014). A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Transactions on Signal and Information Processing*, 3:e2.

[Deng et al., 2013] Deng, L., Li, J., Huang, J.-T., Yao, K., Yu, D., Seide, F., Seltzer, M., Zweig, G., He, X., Williams, J., et al. (2013). Recent advances in deep learning for speech research at microsoft. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8604–8608. IEEE.

[Flenner et al., 2015] Flenner, A., Culp, M., McGee, R., Flenner, J., and Garcia-Cardona, C. (2015). Learning representations for improved target identification, scene classification, and information fusion. In *SPIE Defense+ Security*, pages 94740W–94740W. International Society for Optics and Photonics.

[Gan et al., 2015] Gan, Z., Chen, C., Henao, R., Carlson, D., and Carin, L. (2015). Scalable deep poisson factor analysis for topic modeling. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1823–1832.

[Giryes et al., 2015] Giryes, R., Sapiro, G., and Bronstein, A. M. (2015). Deep neural networks with random gaussian weights: A universal classification strategy? *IEEE Transactions on Signal Processing*, 64(13):3444–3457.

[Glorot et al., 2011] Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Aistats*, volume 15, page 275.

[Handl et al., 2005] Handl, J., Knowles, J., and Kell, D. B. (2005). Computational cluster validation in post-genomic data analysis. *Bioinformatics*, 21(15):3201–3212.

[Hassoun, 1995] Hassoun, M. H. (1995). *Fundamentals of artificial neural networks*. MIT press.

[He et al., 2015a] He, K., Zhang, X., Ren, S., and Sun, J. (2015a). Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

[He et al., 2015b] He, K., Zhang, X., Ren, S., and Sun, J. (2015b). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034.

[Hecht-Nielsen, 1989] Hecht-Nielsen, R. (1989). Theory of the backpropagation neural network. In *Neural Networks, 1989. IJCNN., International Joint Conference on*, pages 593–605. IEEE.

[Hornik et al., 1989] Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

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

[Jia et al., 2014] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[Le Roux et al., 2015] Le Roux, J., Hershey, J. R., and Weninger, F. (2015). Deep nmf for speech separation. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 66–70. IEEE.

[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

[LeCun et al., 2012] LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. (2012). Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer.

[Lee and Seung, 1999] Lee, D. D. and Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791.

[Lee et al., 2010] Lee, H., Yoo, J., and Choi, S. (2010). Semi-supervised nonnegative matrix factorization. *Signal Processing Letters, IEEE*, 17(1):4–7.

[Leen et al., 2001] Leen, T. K., Dietterich, T. G., and Tresp, V. (2001). *Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference*, volume 13. MIT Press.

[Long et al., 2015] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440.

[Moore, 1920] Moore, E. (1920). On the reciprocal of the general algebraic matrix. *Bulletin of the American Mathematical Society*, 26(9):394–395.

[Nair and Hinton, 2010] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814.

[Nguyen et al., 2015] Nguyen, A., Yosinski, J., and Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 427–436. IEEE.

[Oyallon et al., 2013] Oyallon, E., Mallat, S., and Sifre, L. (2013). Generic deep networks with wavelet scattering. *arXiv preprint arXiv:1312.5940*.

[Rajabi and Ghassemian, 2015] Rajabi, R. and Ghassemian, H. (2015). Sparsity constrained graph regularized nmf for spectral unmixing of hyperspectral data. *Journal of the Indian Society of Remote Sensing*, 43(2):269–278.

[Russakovsky et al., 2015] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

[Salton et al., 1975] Salton, G., Wong, A., and Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

[Scherer et al., 2010] Scherer, D., Müller, A., and Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. In *International Conference on Artificial Neural Networks*, pages 92–101. Springer.

[Szegedy et al., 2015] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9.

[Wang, 2016] Wang, Y.-Q. (2016). Small neural networks can denoise image textures well: a useful complement to bm3d. *Image Processing On Line*, 6:1–7.

[Williams and Hinton, 1986] Williams, D. and Hinton, G. (1986). Learning representations by back-propagating errors. *Nature*, 323:533–536.