¡¡¡¡¡¡¡ HEAD ======== ¿¿¿¿¿¿¿ 80c29f3238db03b9b611e13a448baea5a3e96da9
¡¡¡¡¡¡¡ HEAD ======= ¿¿¿¿¿¿¿ 80c29f3238db03b9b611e13a448baea5a3e96da9

RAMAN SPECTROSCOPY DECONVOLUTION USING STACKED AUTO-ENCODERS WITH NON-NEGATIVITY CONSTRAINTS

Jacob S. Larsen, Flavia D. Frumosu, Jakob Thrane, Maximillian F. Vording, Tommy S. Alstrøm

Department of Applied Mathematics and Computer Science, Technical University of Denmark

ABSTRACT

The standard tool to analyze raman spectroscopy signal is non-negative matrix factorization (NMF). In some of the cases, the observed signal is not a linear combination of pure spectra, thus removing the basic assumption behind using NMF for analyzing these signals. The research question addressed in this paper is whether the use of non-negative sparse autoenconders are able to model the nonlinearities and thus are able to correctly deconvolve Raman Spectre. A model used for this purpose has been initially tested on the classical MNIST dataset with good results. A simulated raman spectra dataset has been created and the same model has been compared with the classical approach NMF. As the results indicate, NMF performs better than the proposed model. However there is a strong belief that few changes in the model will improve the results significantly.

***Index Terms*—** Raman spectroscopy, autoencoder, non-negativity constraints, non-negative matrix factorization, sparsity

1. INTRODUCTION

Raman scattering is a technique used to detect and identify molecules using the interaction of photons. It is a technique that is commonly used in chemistry and physics for detecting molecules by observing their vibrational and rotational modes. This is completed with the use of laser light. The photon excites the sample which causes scattering, meaning the photon has a change in energy over a short time period. This energy-change is reflected as a shift in frequency, also called a stokes shift, and by analysing the spectrum the molecules and combination of molecules can be identified.

Surface-enhanced Raman Spectroscopy (SERS) is an enhancement of Raman Scattering, that uses surfaces such as metal or nanostructures to absorb molecules. This enables the identification of single molecules. For instance, noble metal nanostructures can concentrate light which greatly enhances the electromagnetic field near the nanostructure. These areas become so called "hot spots" that amplify weak Raman scattering signals. The placement and design of these nanostructers with high SERS preformance is beyond the scope of this work, however additional information can be found in [1] and references herein.

Missing reference on the complexity of mixtures in spectra, this includes the nonlinearities when mixing.

Modern SERS analysis is done with highly efficient nanostructures, and lasers with low linewidths resulting in very highly resoluted Raman intensity spectras. Traditional curve resolution methods such as MCR/NMF have increased runtimes of multiple days in decomposing these Raman maps.

This work aims to evaluate the application of Sparse Autoencoders with non-negativity constraints on Raman spectra obtained from SERS. The novelty of this work consists of comparing traditional curve resolution methods such as NMF to the methodology and work completed by [2].

This work is organized as follows. Section 2 presents the methodology of non-negative matrix factorization and sparse autoencoders. Section 3 is a description of the non-negativity constraint in sparse autoencoders as proposed by [2]. Section 4 details the implementation done in TensorFlow and verification using the well known MNIST dataset. Moreover this section also details the SERS dataset used for the primary results. Section 5 describes the results obtained. A discussion is provided in Section 6 and a conclusion in Section 7.

2. METHODS

This work seeks to compare traditional and proven methods for SERS analysis, i.e. curve resolution, such as NMF to denoising autoencoders with non-negativity constraints. **Add how NMF/MCR is used in raman spectroscopy.**

2.1. Non-negative matrix factorization

Non-negative matrix factorization (NMF) consists of factorizing a original matrix V, with only positive elements, into two positive matrices W and H. [3]

$$V \approx W \times H \tag{1}$$

Where columns of W are considered basis vectors, and each column in H is considered an encoding with a one-to-

Thanks to XYZ agency for funding.

one relationship with the columns of V. So for instance, a matrix V of size $n \times m$ can be factorized into a matrix W of size $n \times k$ and a matrix H of size $k \times m$, where k is the number of components. Approaching the intuitive point of view, W and H can be seen as components that combined approximate the original signal V.

An iterative algorithm is considered for NMF, which shares similar monotonic convergence as the EM algorithm. [4]. Moreover, the rules of update preserve non-negativity of W and H. The algorithm approaches the problem by initialization of W and H as non-negative, and then update the values in W and H until local maximima is obtained and both matrixes are considered stable. The rules of multiplicative update can be defined as:

$$H^{n+1} \leftarrow H^n \frac{(W^n)^T V}{(W^n)^T W^n H^n} \tag{2}$$

and

$$W^{n+1} \leftarrow W^n \frac{V(H^{n+1})^T}{W H^{n+1}(H^{n+1})^T} \tag{3}$$

This however leads to an optimization that is convex for W and H, which in turn can require many iterations and result in poor local minima. For this problem, alternating least squares (ALS) is commonly used. This consists by alternating the optimization between W and H, e.g. one interation consists of 1) keeping H fixed while minimizing least squares for W, and 2) keeping W fixed while minimizing least squares for H. Further details can be found in [5] and references herein. ALS is for the remainder for this work, assumed the default optimization method when mentioning NMF.

2.2. Sparse auto-encoder

A regular auto-encoder seeks to learn a hidden representation that makes it possible to reconstruct the original input, [6]. The mapping from an input vector to a hidden/latent representation, the encoding step, can be described as:

$$y = f(x) = \sigma(\mathbf{W}x + \mathbf{b}) \tag{4}$$

Where x is the input vector and $\mathbf{W}$ is a matrix of weights and $\mathbf{b}$ is a bias vector. σ is an element-wise logistic sigmoid. y is the latent representation. From the latent representation it is then mapped back, i.e. reconstructed, in input space and can be described as:

$$z = g(y) = \sigma(\mathbf{W}'x + \mathbf{b}') \tag{5}$$

Where z can be seen as the reconstructed vector, and the weight matrix W' is the reverse mapping. The weights at both the encoding and decoding layer are to be optimized as to minimize a "reconstruction error", thus the approach here is unsupervised. This can be formulated as the average reconstruction error using the squared error as a loss function

$$J_E(\mathbf{W}, \mathbf{b}) = \frac{1}{N} \sum_n^N ||x^i - z^i||^2 \tag{6}$$

Where N is the number of training samples. By limiting the size of the hidden layer, i.e. $\mathbf{W}$ and $\mathbf{b}$, the autoencoder will learn a representation of the input that is compressed. This can also be seen as a dimensionality reduction. This enables the discovery of latent stuctures in high dimensional spaces. Imposing a sparse representation can also be seen as limiting the "activation" of the hidden units, using for instance the Kullback-Leibler (KL) divergence. By using the average activation of the hidden layer, sparsity can be enforced. A sparsity parameter is seleted, denoted p, which is a positive number close to zero. The KL divergence between the average activation and the sparsity parameter is then minimized using the following:

$$J_{KL}(p||\hat{\mathbf{p}}) = \sum_{j=1}^{n} p \log \frac{p}{\hat{p_j}} + (1-p) \log \frac{1-p}{1-\hat{p}_j} \tag{7}$$

Here $\hat{\mathbf{p}}$ is a vector of average activations, i.e. per hidden-layer. This combined with a weight decay and the reconstruction error provides with the cost function used for a sparse autoencoder [2]

$$J(\mathbf{W}, \mathbf{b}) = J_E(\mathbf{W}, \mathbf{b}) + \beta J_{KL}(p||\hat{\mathbf{p}}) + \frac{\lambda}{2} \sum_{l=1}^{2} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (w_{ij}^l)^2 \tag{8}$$

Here β regulate the sparsity penality, and λ is a standard weight decay term while s_l and s_{l+1} is the size of connected hidden layers.

3. NON-NEGATIVITY CONSTRAINTS

Much like NMF, the idea of part-based representation, intuitively consists of combining in an additive way. By applying non-negativity constraints on a sparse autoencoder, the reconstruction of the input data is claimed by [2] to be improved, since the autoencoder is capable of decomposing data into sparse elements and additively combine them at the decoder. The work in [2] proposes non-negativity by replacing the weight decay in Eq. 2.2. This replaces λ with α which is greater or equal to zero. Moreover the function $f(x)$ is introduced on the weights which applies a non-negativity constraint, regularized by α.

$$f(w_{ij}) = \begin{cases} w_{ij}^2 & w_{ij} < 0 \\ 0 & w_{ij} > \end{cases} \tag{9}$$

Updating the weights and the biases is done with the use of backpropagation algorithm.

4. SETUP

As proposed [2] and [6], a layer-wise training is used to pretrain each layer unsupervised such that features are learned. In this work, the deep nonnegative constrained autoencoder (NCAE) is replicated using the layered training approach as proposed by [2]. This means each layer of the deep architecture is trained individually, using a stacked approach. In other words, the hidden activations of the previous trained autoencoder is used as input for the next. The final autoencoder is used as input to a softmax classifier. After the individual stacked training approach, the whole network is combined and trained in a supervised fashion, e.g. finetuned. The authors propose the use of a batch gradient descent algorithm called Limited-memory BFGS (L-BFGS) quasi-Newton method which approximates the hessian matrix. However, since the implementation was done in TensorFlow (TF) a native TF optimizer, e.g. Adam, was used instead.

4.1. Verification

MNIST verification

Add a reference with the parameters used for both models from the paper. The sparsity is different.

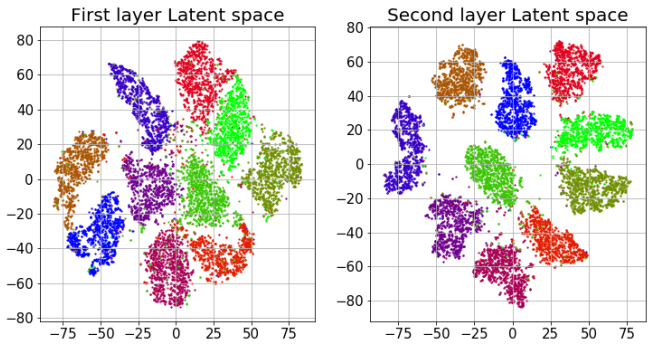

Fig. 1: MNIST latent space after finetuning

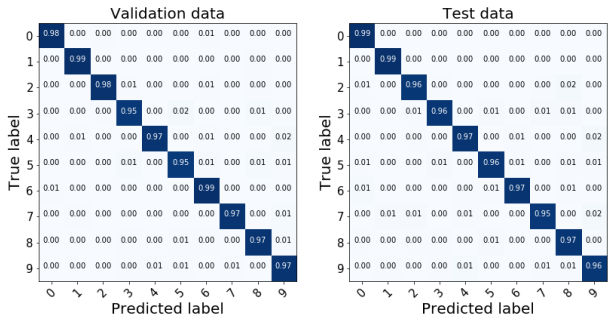

Fig. 2: MNIST confusion matrix

4.2. Simulated Raman data

Dataset origin, description of wavenumbers, raman map size. Definitions for the rest of the paper.

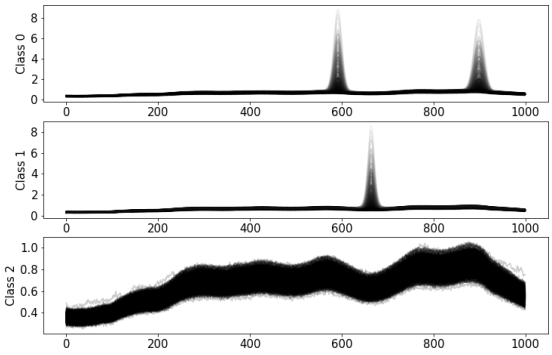

Fig. 4: Raman simulation data

5. RESULTS

This section contains the main results of the deep learning model described in the Setup Section 4 for the simulated raman data set.

As described, the model is a classification model with three classes with class 2 used for collecting the background signal.

In order to check that indeed the model works as expected, several testing figures have been generated. These figures are presented below with a small description.

The first indication that the model functions correctly is the component identification. In terms of number peaks and wave numbers the components in Figure 5 look very similar to the ones described in the Setup Section 4, Figure 4.

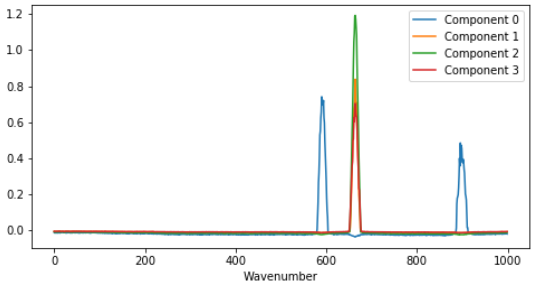

Fig. 5: Component identification

The weights for the second layer and softmax layer is visualized in Figure 6. What is interesting to observe is that weight 0 for class 0 is highly activated in both layers.WHAT SHOULD WE SAY HERE EXACTLY?

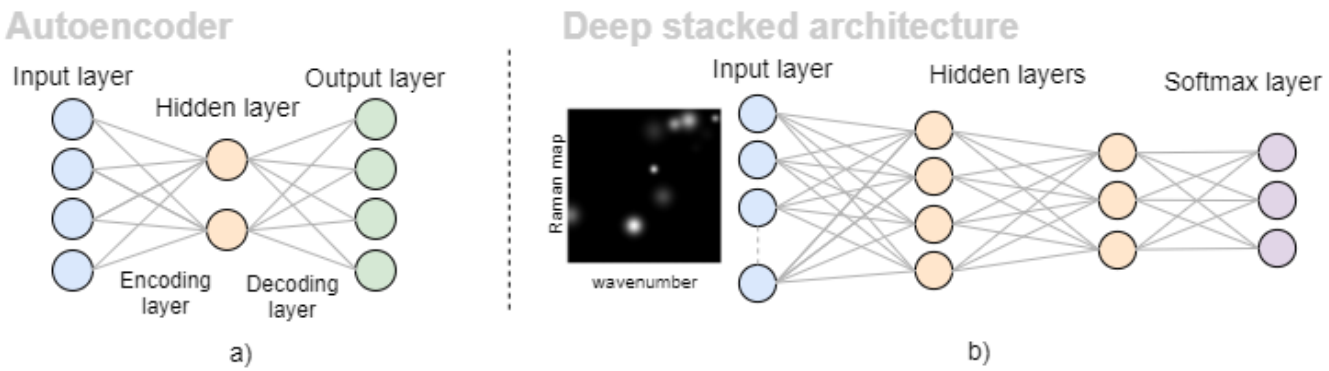

Fig. 3: a) shows a regular autoencoder consisting of an encoding and decoding layer. b) shows the deep stacked architecture with nonnegative constraints used for this work. Spectrum of each Raman map, consisting of 1000 wavenumbers is given as input.

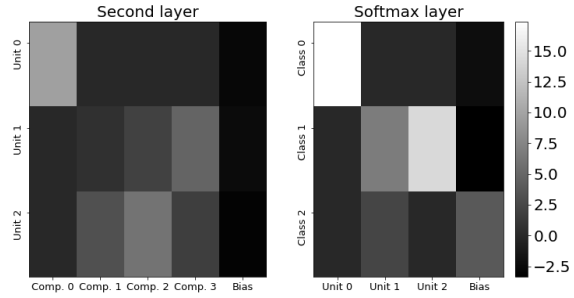

Fig. 6: Weights for the second and softmax layers

An overall overview of the activations and probabilities over the D space can be viewed in Figure 7. As it can be observed, the hot spots are activated (lighter tones) in both images similarly to the original hot spot simulation image, Figure **??**. Class 0 and class 1 represent the two substances while class 2 represents the background. Therefore, one might say that the model performs as expected. HOW TO EXPLAIN THE SIGMOID?

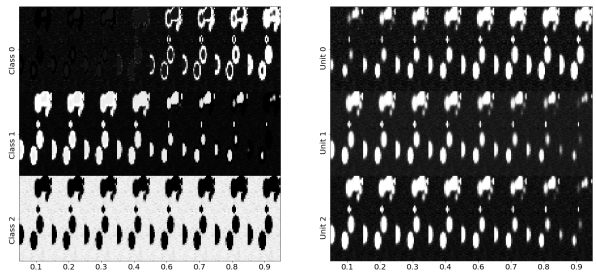

Fig. 7: Left is the probabilities of each class, to the right is the activation of each hidden unit when changing the concentration of each substrate.

We also plotted a running average to check that indeed the probabilities for class 0, 1 and 2 (blue, green and purple solid lines) over space D behave as expected. As one can see in 8, the behaviour is consistent with Figure 7.

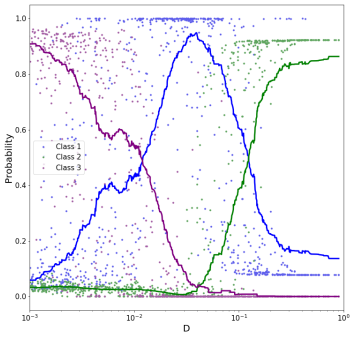

Fig. 8: Probability versus D space for a concentration of 0.7 for class 0 and 0.3 for class 1. Lines are running averages with a window size of 100.

We plotted the confusion matrices for the test and training data, Figure 9. When investigating the miss classifications, we find that when $D > 6 \cdot 10^{-3}$, the model is predicting correctly. This indicates another tuning parameter in terms of the threshold defining the background described in the Setup Section.

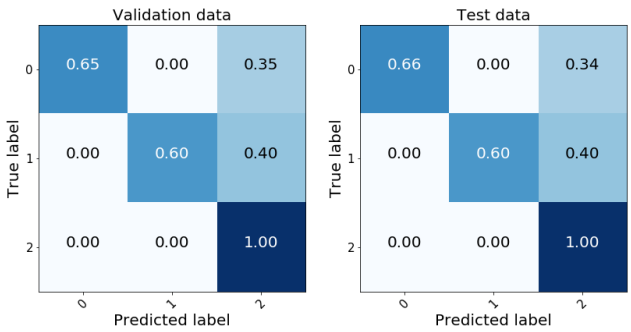

Fig. 9: Confuson matrix for the raman data

In order to check how the model deals with the concentrations, we also made a predicted concentration using a simple transformation of the activations, Figure 10. As the figure shows, the model performs quite poorly since the intersection should be close to the gray diagonals.

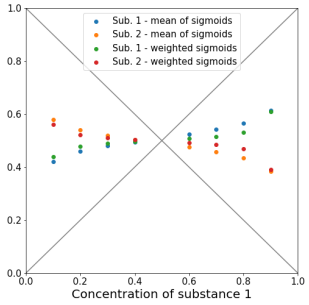

Fig. 10: Predicted concentration using activations from sigmoid function

Finally, we also generated a comparison between NMF and the deep learning model for predicting concentrations, Figure 11. As it can be seen in the figures, NMF performs much better than the deep learning model. The difference between these results is described into depth in the Discussion Section 6.

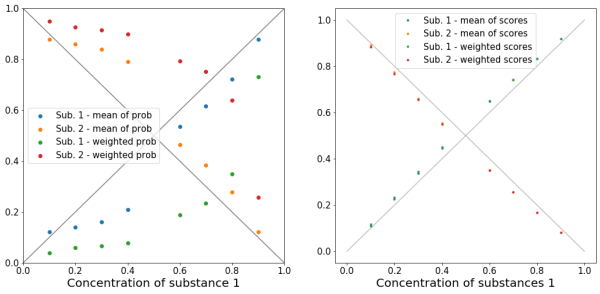

Fig. 11: Predicted concentration using probabilities (left) and NMF (right)

6. DISCUSSION

Comparison with NMF - main discussion here.

7. CONCLUSION

8. REFERENCES

List and number all bibliographical references at the end of the paper. The references can be numbered in alphabetic order or in order of appearance in the document. When referring to them in the text, type the corresponding reference number in square brackets as shown at the end of this sentence .

9. REFERENCES

[1] Hong Wei and Hongxing Xu, "Hot spots in different metal nanostructures for plasmon-enhanced Raman spectroscopy," *Nanoscale*, vol. 5, no. 22, pp. 10794, oct 2013.

[2] Ehsan Hosseini-Asl, Jacek M. Zurada, and Olfa Nasraoui, "Deep Learning of Part-Based Representation of Data Using Sparse Autoencoders With Nonnegativity Constraints," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 12, pp. 2486–2498, dec 2016.

[3] H. Sebastian Seung and Daniel D. Lee, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, oct 1999.

[4] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," 1977.

[5] Amy N. Langville, Carl D. Meyer, Russell Albright, James Cox, and David Duling, "Algorithms, Initializations, and Convergence for the Nonnegative Matrix Factorization," jul 2014.

[6] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol, "Extracting and Composing Robust Features with Denoising Autoencoders," .