Deep Probabilistic NMF Using Denoising Autoencoders

Satwik Bhattamishra

***Abstract*—Non Negative Matrix Factorization (NMF) has received considerable attention due to its application in pattern recognition and computer vision. However, the algorithm is sensitive to noise and assumes that the signals in the data can be linearly reconstructed. In this paper, we propose a robust non-linear probabilistic model and develop its optimization algorithm. The proposed model reduces the data to a lower dimensional manifold to get a more meaningful representation and takes into account the noisy nature of the data to improve the clustering performance of NMF. Additionally, our empirical study validates the effectiveness of the proposed method on some benchmark datasets.**

***Index Terms*—Clustering, denoising autoencoders, dimension reduction, non-negative matrix factorization.**

I. Introduction

Non negative matrix factorization (NMF) is a widely used method for factorizing a matrix into a product of two matrices such that all the elements are non-negative. One of the matrices contains the degree of membership of each sample in each cluster and the other matrix contains a latent representation of the dataset. The method popularized by Lee and Seung [1], [2], has been found to have widespread applications in many areas such as clustering [3], pattern recognition [4], dimensionality reduction [5] and analyzing gene expression data [6], [7].

Many types of datasets in today's world are usually represented as vectors of high dimensionality and NMF has gained considerable attention as it can learn interpretable part based representations of the data by decomposing multivariate data into a linear combination of bases. Several extensions of NMF have been derived to enhance its performance by adding additional constraints and penalty terms to induce sparsity [8], smoothness [9] and spatial localization [10]. However, there are two inherent problems with these methods, the first being the assumption that the signals can be linearly reconstructed from the bases and second being the deterministic nature of the algorithm.

Denoising autoencoders [11] have been shown to learn meaningful higher level representations from the data. In this paper, we propose a method to address the issues mentioned above by embedding stacked denoising autoencoders to non-linearly transform the data and considering a probabilistic case of NMF to take into account the stochastic nature of the data.

The remainder of the paper is organized as follows. In Section II, we discuss some background and prior work related to this paper. Section III contains a brief explanation of the NMF algorithm. We introduce our proposed method in Section IV and in Section V we show some experimental results on commonly used datasets. Finally, Section VI ends with conclusion.

Manuscript received September 30, 2017; revised January 3, 2018.

Satwik Bhattamishra is with Birla Institute of Technology and Science Pilani, Pilani Campus, India (e-mail: satwik55@gmail.com).

II. Related Work

Methods have been created to address the non-linearity issue and the deterministic nature of the algorithm separately. Kernel technique was initially used to address the non-linearity issue and recently there have been some methods that combine NMF with deep learning. Some variations of NMF have taken into account the noisy nature of the data and proposed robust extensions of NMF [12]. Here we briefly describe some prior work relevant to our method.

Kernel based methods [13] have been used to deal with non-linear correlation between data points [14] by using kernel induced non-linear mapping to extract useful latent features from the data. However, the selection of kernels and their parameters is crucial for its performance.

In the Deep NMF model developed by [15], they use a multi-layered NMF combined with a pooling layer which is optimized with backpropagation. The last layer of the multi-layered NMF is decomposed by semi-supervised NMF, instead of NMF. In [16], they attempt to improve encoding vector estimation by using deep neural network to learn a mapping between data vectors and the corresponding encoding vectors. The method proposed by [17] addresses the non-linearity issue by embedding autoencoders to the NMF framework and by training the network and the factor matrices together. However, denoising autoencoders are known to extract more robust and meaningful representation of data as compared to standard autoencoders [18]. Here we initially train the network separately as a pretraining phase and then in a fine-tuning phase, we jointly optimize the weights of the network and the factor matrices to get the final set of clusters. Additionally, the learning criterion for standard autoencoders is deterministic in nature whereas it is stochastic in case of denoising autoencoders.

III. The NMF Algorithm

A. Standard NMF

Given a non-negative matrix $V \in \mathbb{R}^{n \times m}$, NMF aims to find two non-negative matrices $W \in \mathbb{R}^{n \times k}$ and $H \in \mathbb{R}^{k \times m}$ such that

$$V \approx WH \tag{1}$$

NMF is formulated as a constrained optimization problem where an error function is minimized. The two most

doi: 10.18178/ijmlc.2018.8.1.662

commonly used error functions are the Frobenius norm and the Kullback-Leibler divergence between two matrices. The standard approximation problem can be formulated as follows:

$$\min_{W\in\mathbb{R}^{n\times k},H\in\mathbb{R}^{k\times m}} \| V\text{-}WH \|_F^2 \text{ s.t. } W,H \geq 0 \quad (2)$$

The non-negativity constraints allow interpretation of the basis elements in the same way as the data. The constraints lead to a parts-based representation because they allow only additive combinations.

Lee and Seung [2] presented the multiplicative update rules for solving the above problem:

$$W_{i,j} \leftarrow W_{ij}\frac{(VH^T)_{ij}}{(WHH^T)_{ij}} \quad (3)$$

$$H_{i,j} \leftarrow H_{ij}\frac{(W^TV)_{ij}}{(W^TWH)_{ij}} \quad (4)$$

The formulation provides a generative model of the data through linear non-negative constraints. One should note that the minimization problem is convex in W and H separately but the minimization with constraints on both the matrices is highly non-convex [19].

B. Probabilistic Extension

In [20], the authors present a robust extension of NMF which assumes that the data is not deterministic and is noisy in nature. The algorithm assumes that the data is corrupted by noise and follows the conditional distribution,

$$p(V|W,H,\sigma^2) = \prod_{i=1}^{N}\prod_{j=1}^{M}[\mathcal{N}(V_{ij}|u_i^T j_j,\sigma^2)] \quad (5)$$

where $\mathcal{N}(.|\mu,\sigma^2)$ is the probability density function of the Gaussian distribution with mean μ and standard deviation σ, u_i and h_j denote the i^{th} row of matrix W and j^{th} column of matrix H, respectively. They show that under this assumption the problem reduces to the following weighted regularized matrix factorization problem,

$$\min_{W\in\mathbb{R}^{n\times k},H\in\mathbb{R}^{k\times m}} \| V\text{-}WH \|_F^2 + \alpha \| W \|_F^2 + \beta \| H \|_F^2 \quad (6)$$

such that $W,H,\alpha,\beta \geq 0$. The rules to update the factors W and H for the given problem are,

$$W_{i,j} \leftarrow W_{ij}\frac{(VH^T)_{ij}}{(WHH^T+\alpha W)_{ij}} \quad (7)$$

$$H_{i,j} \leftarrow H_{ij}\frac{(W^TV)_{ij}}{(W^TWH+\beta H)_{ij}} \quad (8)$$

where α and β denote the weights of the regularization terms.

IV. Proposed Method

We propose a robust extension of NMF that performs well on non-linearly structured data by using deep neural networks as compared to the standard NMF.

Using stacked denoising autoencoders we aim to learn a stochastic non-linear mapping $\mathbb{R}^n \rightarrow \mathbb{R}^d$ to transform the data to a lower d-dimensional space. A standard autoencoder takes an input vector $v \in [0,1]^n$ and maps it to a latent representation $y \in [0,1]^d$ through a deterministic mapping $y = f_\theta(v) = g(Av+b)$, parameterized by $\theta = \{A,b\}$. $A \in \mathbb{R}^{n\times d}$ is a weight matrix and b is a bias vector. The latent representation is then mapped back to a reconstructed vector $\hat{v} \in [0,1]^n$ in the input space $\hat{v} = h_{\theta'} = g(A'y+b')$ where $\theta' = \{A', b'\}$. The parameters (θ) of the model are optimized to get the minimum reconstruction error between $\hat{v}$ and v.

In case of denoising autoencoders the initial input vector v is corrupted by means of a stochastic mapping $\tilde{v} \sim q_D(\tilde{v}, v)$. The two most common forms of corruption are adding Gaussian noise to the input data or randomly forcing a proportion μ of each input vector to 0. As with standard autoencoders, the corrupted input is then mapped to a latent representation and then reconstructed back to the input space. One should note that although the mapping function is still deterministic but it is a function of $\tilde{v}$ rather than v resulting in a stochastic mapping of v. It must be mentioned that once the mapping f_θ has been learned, the original uncorrupted inputs will be used to produce the higher level representations. While training, we transform the input vectors using,

$$f_\theta^{(1)}(\tilde{v}) = g_1(A_1 \cdot \tilde{v} + b_1) \quad (9)$$

and for subsequent layers,

$$f_\theta^{(k+1)}(v) = g_{k+1}(A_{k+1} \cdot \tilde{f_\theta^{(k)}}(v) + b_{k+1})$$

where $\theta = \{A,b\}$. A and b are the same weight matrix and bias vector as defined before. $g(.)_i$ is any non-negative non-linear activation function and the last layer must be sigmoid activation function ($\sigma(v) = 1/(1+e^{-v})$) so that the output vector $f_\theta(\tilde{v})$ is in $[0,1]^d$.

The layers of the stacked denoising autoencoders are trained layerwise. Only the layer which is being trained will receive corrupted input from the previous layer. Once the layer has been trained, it will receive clean input and the next layer will receive corrupted input for training [11]. Once the whole network has been trained, we use the encoder to map the input vectors to a lower d dimensional space,

$$f_\theta(v) = g_3(A_3 \cdot g_2(A_2 \cdot g_1(A_1 \cdot v + b_1) + b_2)b_3) \quad (10)$$

The traditional squared error and cross-entropy are the most commonly used loss functions for autoencoders. We use the cross-entropy error function for our model and for each vector v^i, it is defined as,

$$L_{CE_i} = -\Sigma_k[v_k^i \log \hat{v}_k^i + (1-v_k^i)\log \hat{v}_k^i)] \quad (11)$$

where

$$L_{CE} = \Sigma_i L_{CE_i}$$

Although the encoder receives corrupted input while training, the reconstruction error is minimized over the clean input $v$and the reconstructed vector $\hat{v}$. This enables denoising autoencoders to learn more robust and meaningful

representations [18] of the input data.

Based on the non-linear transformation $f_\theta(v)$, the adapted loss function for PNMF (6) is defined as,

$$L_{PNMF} = \min_{W,H\geq 0} \| f_\theta(V)\text{-}WH \|_F^2 + \alpha \| W \|_F^2 + \beta \| H \|_F^2 \quad (12)$$

We take a weighted combination of the transformed PNMF error function and the cross-entropy function to form our overall loss function for the network,

$$J(\theta, W, H) = L_{PNMF} + \lambda L_{CE} \quad (13)$$

Once the network has been trained layerwise to learn a suitable mapping based on minimizing only the reconstruction error, we fine-tune the overall network and update the matrices W and H to minimize the overall loss function (13) and get the desired factorization.

In the overall loss function (13), the factorized matrices W and H only influence the first term L_{PNMF} whereas the parameters θ of the neural network influence both the terms.

Thus, we update the factor matrices and the parameters of the network separately for each iteration. Since optimizing W and H is equivalent to minimizing L_{PNMF} we use the multiplicative update rules (7), (8) with respect to $f_\theta(V)$,

$$W_{i,j} \leftarrow W_{ij} \frac{(f_\theta(V)H^T)_{ij}}{(WHH^T + \alpha W)_{ij}} \quad (14)$$

$$H_{i,j} \leftarrow H_{ij} \frac{(W^T f_\theta(V))_{ij}}{(W^T WH + \beta H)_{ij}} \quad (15)$$

We use gradient descent to fine-tune the parameters θ of the neural network. To minimize the overall loss function (13) with matrices W and H fixed, we need the derivatives of the function $J(\theta)$ with respect to θ. The derivatives with respect to each parameter θ_i in $\theta = \{A, b\}$ are,

$$\frac{\partial J(\theta)}{\partial \theta_i} = \frac{\partial L_{PNMF}}{\partial \theta_i} + \lambda \frac{\partial L_{CE}}{\partial \theta_i}$$

$$\frac{\partial J(\theta)}{\partial \theta_i} = \frac{\partial f_\theta(V)}{\partial \theta_i} \frac{\partial L_{PNMF}}{\partial f_\theta(V)} + \lambda \frac{\partial L_{CE}}{\partial \theta_i}$$

$$\frac{\partial J(\theta)}{\partial \theta_i} = \frac{\partial f_\theta(V)}{\partial \theta_i} \cdot 2[f_\theta(V)\text{-}WH] + \lambda \frac{\partial L_{CE}}{\partial \theta_i}$$

The partial derivatives $\frac{\partial f_\theta(V)}{\partial A}$ and $\frac{\partial f_\theta(V)}{\partial b}$ can be computed using backpropagation algorithm. The derivatives of L_{CE} with respect to A and b are computed as in the case of a general autoencoder using standard backpropagation. Thus, in each iteration we update the factor matrices W and H with the parameters of the network fixed and then using gradient descent, we update the parameters of the network with the factor matrices fixed.

The overall algorithm can be divided into two phases, i.e. Pretraining and Training. Initially the matrices W and H are set to random non-negative numbers. The parameters of the network are initialized randomly. During pretraining the network is trained layerwise by minimizing the reconstruction error (11) and by feeding corrupted input to the layer being trained. During the training phase, the factor matrices and the parameters of the network are updated circularly in each iteration to minimize the overall loss function (13).

Algorithm I DP-NMF

Input Data: $V \in R^{n\times m}$

Initialize $W^{(0)}, H^{(0)}$ with Non-negative numbers
Randomly initialize $A^{(0)}, b^{(0)}$
Set learning rate γ_i

For Each layer l **do**

Set corruption level μ
Corrupt output from $(l\text{-}1)^{th}$ layer
repeat K-times
$A_l \leftarrow A_l\text{-}\eta\nabla_A L_{CE}^{(l)}$
$b_l \leftarrow b_l\text{-}\eta\nabla_b L_{CE}^{(l)}$

repeat I-times
$W_{i,j} \leftarrow W_{ij} \frac{(f_\theta(V)H^T)_{ij}}{(WHH^T+\alpha W)_{ij}} H_{i,j} \leftarrow H_{ij} \frac{(W^T f_\theta(V))_{ij}}{(W^T WH+\beta H)_{ij}}$
repeat J-times
$A := A\text{-}\eta\nabla_A J(\theta)$
$b := b\text{-}\eta\nabla_b J(\theta)$

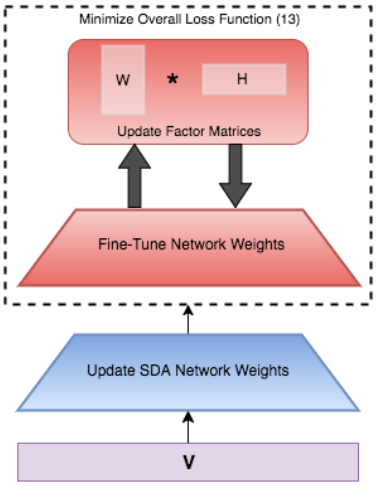


Fig. 1. DP-NMF using stacked denoising autoencoders (SDA).

V. Numerical Experiments

In the experiments we use four image datasets, including UMIST[1], YALE[2]and the ORL dataset [21]. UMIST is a face database, consisting of 575 images of 20 different people. The size of each cropped image is 32×32 with 256 gray levels per pixel. The YALE database contains 15 subjects and 11 different pictures of each subject. The images are resized to 32×32 pixels. The ORL dataset includes 400 images of 40 subjects and the images are resized to 23×28 pixels. Table I summarizes the details of the used datasets.

TABLE I: Dataset Description

Dataset	Size	Features	Classes
UMIST	575	1024 (32×32)	20
ORL	400	644 (23×28)	40
YALE	165	1024 (32×32)	15

We evaluate our method using two standard metrics used for clustering, i.e, Clustering Accuracy and Normalized

[1] https://www.sheffield.ac.uk/eee/research/iel/research/face
[2] http://cvc.cs.yale.edu/cvc/projects/yalefaces/yalefaces.html

Mutual Information, whose definition can be found in [12]. Performance of our proposed method (DP-NMF) is compared with standard NMF and k-means is chosen as a baseline method.

TABLE II: SETUP AND PARAMETER SPECIFICATIONS

Dataset	Layer Specifications	α, β	λ
UMIST	1024-600-250-100	0.15, 0.25	0.2
ORL	644-400-200-100	0.15, 0.20	0.4
YALE	1024-600-200	0.30, 0.25	0.5

Before applying our method, we pre-process the images by scaling the pixel values to $[0,1]$. The details regarding the neural network architecture and other parameters are described in Table II.

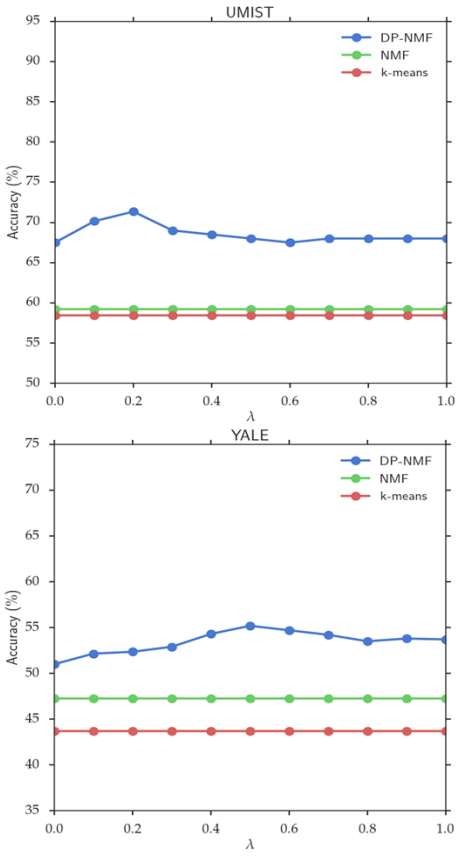
In our experiments we randomly set a proportion (μ) of the inputs to 0. A value of μ in the range from 0.2 to 0.3 for every layer works well for the chosen datasets. To employ our method, we choose the parameters as specified in Table II and iterate (I in Algorithm I) the update rules 250-400 times depending on the size of the dataset and the learning rate. Initially we set J as 50 to update the factor matrices but to reduce time, we scale it down by a factor of 2 a few times as we iterate through the outer loop.

TABLE III: CLUSTERING ACCURACY (%)

Data\Method	K-Means	NMF	DP-NMF
UMIST	58.47	59.23	**71.35**
YALE	43.71	47.24	**55.16**
ORL	62.55	60.07	**69.38**

TABLE IV: NORMALIZED MUTUAL INFORMATION (%)

Data\Method	K-Means	NMF	DP-NMF
UMIST	68.70	67.59	**76.01**
YALE	53.22	56.94	**62.88**
ORL	77.81	76.32	**83.19**



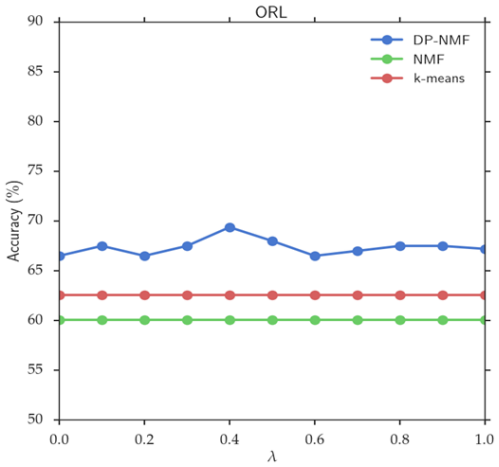


Fig. 2. Accuracy vs. λ.

The comparison of the performance between DP-NMF and other mentioned algorithms are shown in Table III and Table IV. We can observe that the proposed method outperforms the baseline methods for all three datasets. We used k-means by randomly initializing the centroids and iterating multiple times and similarly for the standard NMF, we initialized the factor matrices randomly and repeated the update rules until convergence.

The influence of the parameter λ over clustering accuracy for each dataset is depicted in Fig. 2. Since the curve is neither monotonous nor concave, it is difficult to choose the optimum value of λ without grid search. Low values of parameters α and β improve the clustering performance but the performance decreases for higher values of the parameters in case of the datasets chosen for our experiments. Although the performance of the model seems sensitive to the choice of λ, based on our experiments the model will outperform traditional clustering methods in most cases irrespective of the choice of λ. This is important since, when applied to unlabeled real data, the hyperparameters cannot based on cross-validation.

VI. CONCLUSION

In this paper, we introduced a robust extension of NMF by embedding denoising autoencoders with a probabilistic form of NMF. The proposed model learns a mapping from the input space to a lower-dimensional feature space and then optimizes a clustering objective. The experiments validate that the non-linear dimension reduction is effective and improves clustering performance. We develop the optimization model for the proposed DP-NMF model and show an iterative method to optimize it by extending Lee and Seung's multiplicative update rules and by using gradient descent. Due to the robustness and the ability to find non-linear structure in the data, the proposed method can be applied to various high-dimensional practical datasets which inherently contain noise such as biological and geological datasets.

REFERENCES

[1] D. D. Lee and H. S. Seung, "Learning the parts of objects by nonnegative matrix factorization," *Nature*, vol. 401, pp. 788-791, 1999.

[2] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," *Advances in Neural Information Processing Systems*, 2001.
[3] C. Ding, X. He, and H. D. Simon, "On the equivalence of nonnegative matrix factorization and spectral clustering," in *Proc. the 2005 SIAM International Conference on Data Mining*, 2005.
[4] W. Liu, N. Zheng, and Q. You, "Nonnegative matrix factorization and its applications in pattern recognition," *Chinese Science Bulletin*, vol. 51, pp. 7-18, 2006.
[5] S. Tsuge, M. Shishibori, S. Kuroiwa and K. Kita, "Dimensionality reduction using non-negative matrix factorization for information retrieval," in *Proc. 2001 IEEE International Conference on Systems, Man, and Cybernetics*, 2001.
[6] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns," *Proceedings of the National Academy of Sciences*, vol. 95, pp. 14863-14868, 1998.
[7] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E. S. Lander, and T. R. Golub, "Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation," *Proceedings of the National Academy of Sciences*, vol. 96, pp. 2907-2912, 1999.
[8] P. O. Hoyer, "Non-negative sparse coding," in *Proc. the 2002 12th IEEE Workshop on Neural Networks for Signal Processing*, 2002.
[9] T. Virtanen, "Sound source separation using sparse coding with temporal continuity objective," in *Proc. ICMC*, 2003.
[10] S. Z. Li, X. W. Hou, H. J. Zhang, and Q. S. Cheng, "Learning spatially localized, parts-based representation," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, 2001.
[11] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, pp. 3371-3408, 2010.
[12] L. Zhang, Z. Chen, M. Zheng and X. He, "Robust non-negative matrix factorization," *Frontiers of Electrical and Electronic Engineering in China*, vol. 6, pp. 192-200, 2011.
[13] V.-H. Duong, W.-C. Hsieh, P. T. Bao, and J.-C. Wang, "An overview of kernel based nonnegative matrix factorization," in *Proc. 2014 IEEE International Conference on Orange Technologies*, 2014.
[14] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, 2004.
[15] J. Flenner and B. Hunter, *A Deep Non-Negative Matrix Factorization Neural Network*.
[16] T. G. Kang, K. Kwon, J. W. Shin, and N. S. Kim, "NMF-based target source separation using deep neural network," *IEEE Signal Processing Letters*, vol. 22, pp. 229-233, 2015.
[17] H. Zhang, H. Liu, R. Song and F. Sun, "Nonlinear non-negative matrix factorization using deep learning," in *Proc. 2016 International Joint Conference on Neural Networks*, 2016.
[18] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. the 25th International Conference on Machine Learning*, 2008.
[19] A. Cichoki, R. Zdunek, A. H. Phan, and S. Amari, *Nonnegative Matrix and Tensor Factorization*, Wiley, 2009.
[20] B. Bayar, N. Bouaynaya, and R. Shterenberg, "Probabilistic non-negative matrix factorization: Theory and application to microarray data analysis," *Journal of Bioinformatics and Computational Biology*, vol. 12, p. 1450001, 2014.
[21] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in *Proc. the Second IEEE Workshop on Applications of Computer Vision*, 1994.

Satwik Bhattamishra is currently pursuing a bachelor's degree in computer science and an integrated master's degree in biological science at Birla Institute of Technology and Science Pilani, Pilani Campus, India. His research interests lie in machine learning, optimization, probabilistic modeling and bioinformatics.