Deep Linear Discriminant Analysis with Variation for Polycystic Ovary Syndrome Classification

Raunak Joshi, Abhishek Gupta, Himanshu Soni, and Ronald Laban

Abstract The polycystic ovary syndrome diagnosis is a problem that can be leveraged using prognostication based learning procedures. Many implementations of PCOS can be seen with Machine Learning but the algorithms have certain limitations in utilizing the processing power graphical processing units. The simple machine learning algorithms can be improved with advanced frameworks using Deep Learning. The Linear Discriminant Analysis is a linear dimensionality reduction algorithm for classification that can be boosted in terms of performance using deep learning with Deep LDA, a transformed version of the traditional LDA. In this result oriented paper we present the Deep LDA implementation with a variation for prognostication of PCOS.

Keywords Deep Learning · Deep LDA · Linear Discriminant Analysis

1 Introduction

The use of medical research data for various statistical tasks has been done from a prolonged period of time. The data after having consistent number of records can be utilized for deriving inference using prognostication methods. The methods that fall under the area of inferential statistics [1] which are extended with applied areas of statistics can be used, one of which can be used is Machine Learning [2]. Task of prognostication based on data can be done by learning patterns from the data using machine learning. The dataset that we have used in this paper is pertaining to polycystic ovary syndrome [3] diagnosis, which falls under the classification [4] category. Classification has 2 sub-divisions, viz. Binary and Multi-Class where the data used in this paper falls under binary classification [5] precisely. The methods using Machine Learning have already been performed on polycystic ovary syndrome

R. Joshi · A. Gupta (✉)
University of Mumbai, Mumbai 400032, India
e-mail: abhishek.gupta20001@gmail.com

H. Soni · R. Laban
St. John College of Engineering and Management, Palghar 401404, India

V. E. Balas et al. (eds.), *Intelligent Computing and Networking*, Lecture Notes in Networks and Systems 632, https://doi.org/10.1007/978-981-99-0071-8_3

abbreviated as PCOS using logistic regression [6], bagging ensemble methods [7], discriminant analysis [8], stacked generalization [9], boosting ensemble methods [10] and deep neural networks [11]. The use of deep learning is evident and we want to focus on the more variations that can be brought into the current state-of-the-art system. Deep Learning [12, 13] provides more depth of learning as compared to machine learning and is used when the amount of parameters in dimensions are high. The PCOS dataset has over 41 dimensions which are enough for instating the use of deep learning. The variation we wanted to perform was related to some machine learning algorithm that can be leveraged with power of deep learning. The implementation of any machine learning algorithm using a library like scikit-learn [14, 15] meets limitations in terms of utilization with GPU processing power. For the same reason, using a mature framework like Tensorflow [16] can definitely bring change to the working. This is where we decided to work with parametric learning method which works with simple and definite procedures. The parametric learning method we focused on using was discriminant analysis [17, 18] which has variations in it where we focused on Linear Discriminant Analysis [19]. This actually accounted for an idea that training the linear discriminant analysis with deep learning style will yield us Deep Linear Discriminant Analysis [20, 21] which has been already been discovered and we decided to proceed with out implementation using it. The variations that we brought in the network will be explained in further sections of this paper.

2 Methodology

This section of the paper gives detailed insights about the implementation and approach we have taken to solve the problem. The model considering the implementation with respect to Deep LDA [20, 21] revolves around the idea of the convolutional neural networks [22–25]. The modification can be done to work with numerical values. This has been implemented by various developers and names of the developers are given in the acknowledgement section of the paper. The Deep LDA is basically an implementation of latent representations in linearly separable method. The Deep LDA is an extensive implementation of the traditional Linear Discriminant Analysis which was intended for dimensionality reduction based classification methods. The implementation consists of 2 phases, first phase consists of linear discriminator as the deep neural network and second phase consists of support vector machine for detailed classification.

2.1 First Phase

The Fig. 1 gives depiction of first phase of the implementation. The input layer takes 41 dimension of features from the data. This is passed on to one dense layer that has

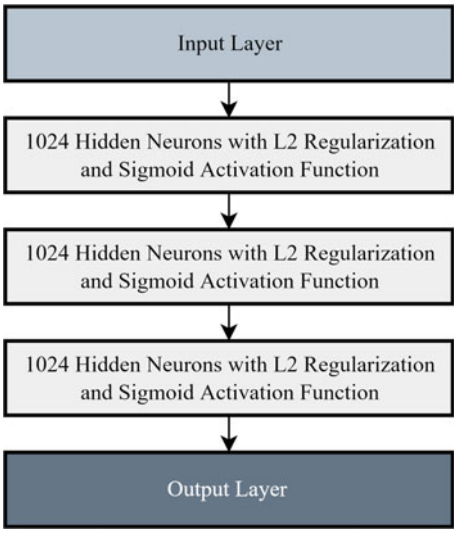

Fig. 1 First phase with LDA implementation

1024 hidden neurons. The L2 regularization [26] is applied as a kernel regularization for the layer. The activation function is used sigmoid [27]. The rectified linear unit abbreviated as ReLU [28] is the activation function commonly used for deep learning methods but using sigmoid ensures linear based system such as developed for linear discriminant analysis. The parameters learned from the first hidden layer are 43,008. Similar type of hidden layers are repeated twice, where second hidden layer learns 1,049,600 parameters and third hidden layer also learns same amount of parameters. The output layer consists of 1 hidden neuron with sigmoid activation function and learns 1025 parameters. The network learns total of 2,143,233 parameters where all the parameters are trainable. The loss function used is binary cross-entropy that differs from the original implementation of deep LDA paper. The loss optimizer used is Adam [29] optimizer with learning rate of $1 * 10^{-5}$ that roughly denotes 0.00001. The implementation is done using Keras [30] over Tensorflow [16] back-end trained for 100 epochs with 64 as batch size.

2.2 Second Phase

The Fig. 2 depicts the second phase implementation. This is done using Support Vector Machine [31] implementation with neural network inclination. The input layer is connected to hidden layer with 100 hidden neurons with ReLU [28] activation function. This layer learns 200 parameters. This layer is connected to dropout [32] layer with 50% threshold and does not learn any parameters. The output layer has 1 hidden neuron and has sigmoid activation function for binary classification and learns 101 parameters. The total parameters learned are 301 and the network uses

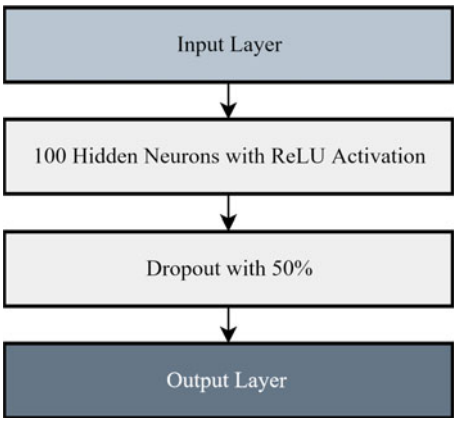

Fig. 2 Second phase with SVM implementation

binary cross-entropy. The loss optimizer used is Adam [29] optimizer with $1 * 10^{-5}$ that approximately denotes 0.00001 learning rate. The network is trained with 100 epochs and 64 as batch size. 2.3 Complete Network (Fig. 3).

The complete network is accumulation of first and second phase where the output of the first phase is the input for second phase. The output of the first phase is 1-dimensional array from 41-dimensional output. This is given as an input to the second phase of the network and final prediction which is 1-dimensional is achieved. The both phases are trained independently and the output is retained from first phase and given as second phase. The results of the network will be given in succeeding section of the paper.

3 Results

3.1 Accuracy and Loss for First Phase

The training and validation accuracy graph can be seen from Fig. 4 and the infractions between the training and validation accuracy seem a bit wider but the values for training are 98.35% and validation 90.909% respectively. Considering the loss, the validation loss has some infraction that can be seen with different metrics the training loss is 6.79% whereas the validation loss is 38.05% respectively.

3.2 Accuracy and Loss for Second Phase

The second phase contains support vector machine and not necessarily the graph depiction is right measure but we have included the graph. The inference can be

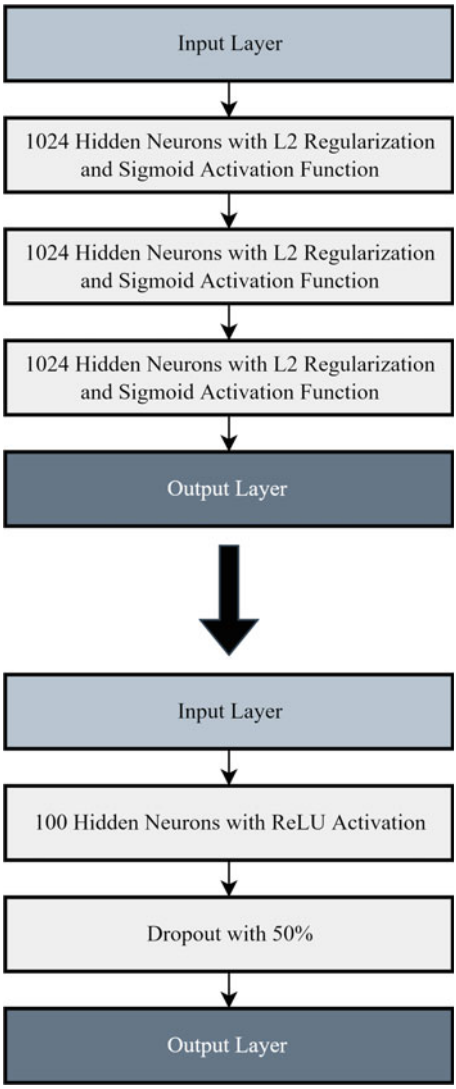

Fig. 3 Complete network

drawn as there are no significant changes in the accuracy or loss. All the metrics are learnt from the trained parameters of the first phase and it does not make much sense to make mappings out of it. The training accuracy for the support vector machine phase is generated as 98.354% and validation accuracy is obtained as 90.909% which is similar to the first phase training and validation accuracy. The training loss is 6.79% and validation loss is 38.052% which is again similar to the first phase. The better inference can be generated from different metrics intended for classification and not just the graph depictions of the training and validation accuracy as well as loss (Fig. 5).

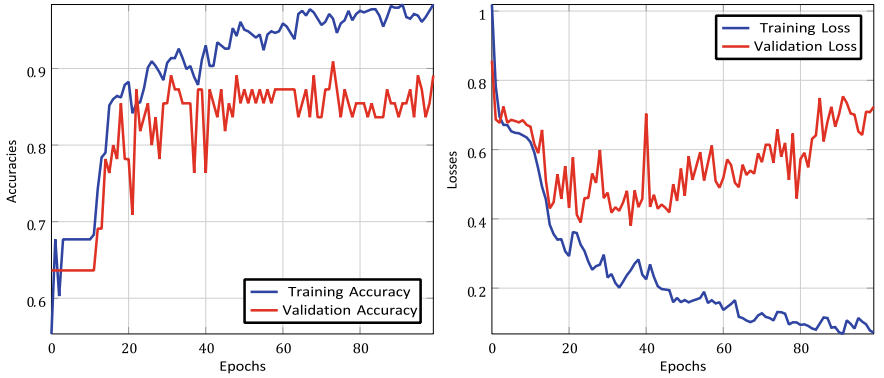

Fig. 4 Accuracy and loss for the first phase

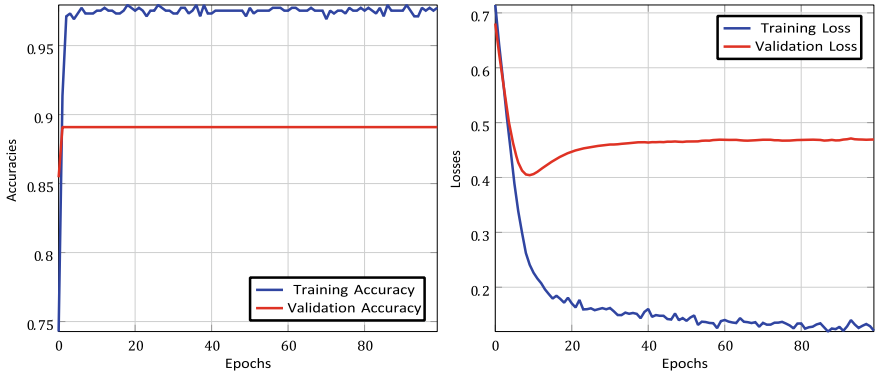

Fig. 5 Accuracy and loss for the second phase

3.3 Precision

The precision [33] is dependent on total number of samples that are predicted to be positive among all the set of samples. This is a popular metric for prognostication algorithms and requires basic elements of a confusion matrix [34], viz. true positives, true negatives, false positives and false negatives. The precision score obtained in 88.88% which is very close to 1 as expected.

3.4 Recall

The recall [33] is a metrics which states all the positive elements from the every single predicted element. The recall also utilizes every single element from the confusion

matrix same as precision. The recall generated for the model is 80% which is again a very good score and gives inference that model has performed adequately. The recall is not only metric that gives the final inference and more precise metric can be obtained.

3.5 F-Score

This is a subtle metric that gives the overall flow of how efficiently does the model perform. The building blocks of F-score [33, 35] are precision and recall. The F-Score we got for the model is 84.21%, which is adequately good and proves that model performs better on average.

4 Conclusion

The idea of entire paper revolves around an experimentation that can be performed for polycystic ovary syndrome diagnosis problem. We proved a point that simple machine learning algorithms can be leveraged using deep learning for efficient performance based inclination. The Deep Linear Discriminant Analysis idea was proven in this paper. We also introduced some of our personal variations in implementation and they turned out to be effective from the results section of the paper. The paper can definitely sum up many lost ideas into practical implementation and enforce many young researchers for better development perspective.

Acknowledgements We would genuinely like to thank Mr. Prasoon Kottarathil for making the polycystic ovary syndrome dataset available through Kaggle platform. We would also like to deeply thank for the contributions of VahidooX - https://github.com/VahidooX/DeepLDA and Thomas Chaton https://github.com/tchaton/DeepLDA for providing us the basic implementations of Deep LDA based from the original paper.

References

1. Marshall G, Jonker L (2011) An introduction to inferential statistics: a review and practical guide. Radiography 17
2. Sarker IH (2021) Machine learning: algorithms, real-world applications and research directions. SN Comput Sci 2(3):1–21
3. Ndefo UA, Eaton A, Green MR (2013) Polycystic ovary syndrome: a review of treatment options with a focus on pharmacological approaches. Pharmacy Therap 38(6):336
4. Cormack RM (1971) A review of classification. J Roy Stat Soc Ser A (Gen) 134(3):321–367. http://www.jstor.org/stable/2344237

5. Kumari R, Srivastava SK (2017) Machine learning: a review on binary classification. Int J Comput Appl 160(7):11–15. https://doi.org/10.5120/ijca2017913083, http://www.ijcaonline.org/archives/volume160/number7/27084-2017913083
6. Chauhan P, Patil P, Rane N, Raundale P, Kanakia H (2021) Comparative analysis of machine learning algorithms for prediction of pcos. In: 2021 international conference on communication information and computing technology (ICCICT), pp 1–7. https://doi.org/10.1109/ICCICT50803.2021.9510128
7. Kanvinde N, Gupta A, Joshi R (2022) Binary classification for high dimensional data using supervised non-parametric ensemble method. arXiv preprint arXiv:2202.07779
8. Gupta A, Soni H, Joshi R, Laban RM (2022) Discriminant analysis in contrasting dimensions for polycystic ovary syndrome prognostication. arXiv preprint arXiv:2201.03029
9. Nair S, Gupta A, Joshi R, Chitre V (2022) Combining varied learners for binary classification using stacked generalization. arXiv preprint arXiv:2202.08910
10. Gupta AM, Shetty SS, Joshi RM, Laban RM (2021) Succinct differentiation of disparate boosting ensemble learning methods for prognostication of polycystic ovary syndrome diagnosis. In: 2021 international conference on advances in computing, communication, and control (ICAC3). IEEE, pp 1–5
11. Gupta A, Nair S, Joshi R, Chitre V (2022) Residual-concatenate neural network with deep regularization layers for binary classification. arXiv preprint arXiv:2205.12775
12. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444
13. Goodfellow I, Bengio Y, Courville A (2016) Deep learning
14. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830
15. Buitinck L, Louppe G, Blondel M, Pedregosa F, Mueller A, Grisel O, Niculae V, Prettenhofer P, Gramfort A, Grobler J, Layton R, VanderPlas J, Joly A, Holt B, Varoquaux G (2013) API design for machine learning software: experiences from the scikit-learn project. In: ECML PKDD workshop: languages for data mining and machine learning, pp 108–122
16. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X (2015) TensorFlow: large-scale machine learning on heterogeneous systems. https://www.tensorflow.org/. Software available from tensorflow.org
17. King LJ (1970) Discriminant analysis: a review of recent theoretical contributions and applications. Econ Geogr 46:367–378. http://www.jstor.org/stable/143150
18. Das Gupta S (1980) Discriminant analysis. In: Fienberg SE, Hinkley DV, Fisher RA (eds) An appreciation. Springer New York, New York, NY, pp 161–170
19. Tharwat A, Gaber T, Ibrahim A, Hassanien AE (2017) Linear discriminant analysis: a detailed tutorial. AI Commun 30:169–190
20. Dorfer M, Kelz R, Widmer G (2015) Deep linear discriminant analysis. arXiv preprint arXiv:1511.04707
21. Tian Q, Arbel T, Clark JJ (2017) Deep lda-pruned nets for efficient facial gender classification. In: 2017 IEEE conference on computer vision and pattern recognition workshops (CVPRW), pp 512–521. https://doi.org/10.1109/CVPRW.2017.78
22. LeCun Y, Bengio Y et al. Convolutional networks for images, speech, and time series
23. Alzubaidi L, Zhang J, Humaidi AJ, Al-Dujaili A, Duan Y, Al-Shamma O, Santamaría J, Fadhel MA, Al-Amidie M, Farhan L (2021) Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions. J Big Data 8(1):1–74
24. Gupta A, Joshi R, Laban R (2022) Detection of tool based edited images from error level analysis and convolutional neural network. arXiv preprint arXiv:2204.09075
25. Joshi RM, Shah D (2022) Refactoring faces under bounding box using instance segmentation algorithms in deep learning for replacement of editing tools. In: Intelligent computing and networking. Springer, pp 236–247

26. Cortes C, Mohri M, Rostamizadeh A (2012) L2 regularization for learning kernels. arXiv preprint arXiv:1205.2653
27. Minai AA, Williams RD (1993) Original contribution: on the derivatives of the sigmoid. Neural Netw 6(6):845–853. https://doi.org/10.1016/S0893-6080(05)80129-7
28. Agarap AF (2018) Deep learning using rectified linear units (relu). arXiv preprint arXiv:1803.08375
29. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980
30. Chollet F et al (2015) Keras. https://github.com/fchollet/keras
31. Hearst M, Dumais S, Osuna E, Platt J, Scholkopf B (1998) Support vector machines. IEEE Intell Syst Their Appl 13(4):18–28. https://doi.org/10.1109/5254.708428
32. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(56):1929–1958. http://jmlr.org/papers/v15/srivastava14a.html
33. Powers DM (2020) Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. Xiv preprint arXiv:2010.16061
34. Ting KM (2017) Confusion matrix. Encyclop Mach Learn Data Mining 260
35. Goutte C, Gaussier E (2005) A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In: European conference on information retrieval. Springer, pp 345–359