Respiratory Rate Estimation using Dynamic Baysian Network

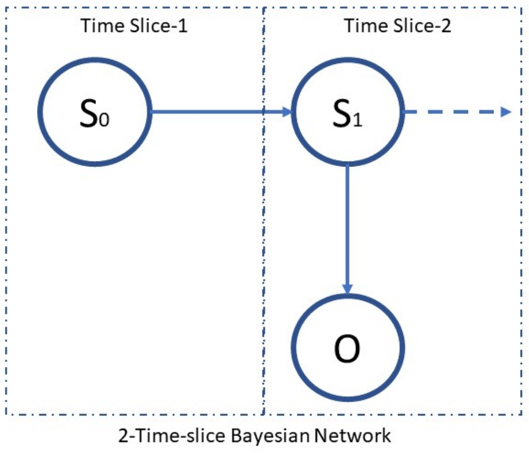
I. Abstract

Respiratory rate is a highly informative indicator of physical health and a vital sign of health deterioration. Continuous monitoring of respiratory rate helps in estimating the health status and body stability of an individual. The primary aim of this project is to estimate the respiratory rate of an individual throughout the day based on inertial, heart rate and temperature measurements. For this, two regression models are implemented to estimate the respiratory rate. The first method consists of a neural network model which ignores the temporal dynamics of the data. In the second method, the temporal dynamics of the real time data is incorporated by implementing a Dynamic Bayesian Network.

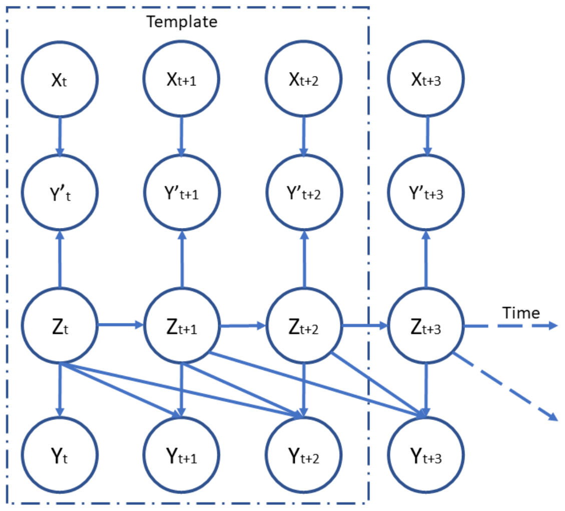
II. Introduction

Temporal models are models that capture the dynamics of a system over time. In temporal setting, models represent distributions over the systems variables which take different states over the time. Temporal models are represented using template based model. A template based model consists of a template that has the essential state variable and observation information. This template is repeated over a time interval to display the distribution of the states over the entire time sequence. In temporal models, the prime interest lies in reasoning about the states as they evolve over time. Such states are represented by a random variable which more specifically are called template variable. This template variable is instantiated after every time interval over the entire time sequence and every such instantiated variable acts as the variable of the probability distribution that characterizes the temporal model. Dynamic Bayesian Network [20] is a kind of temporal model that utilizes the properties of Bayesian network to provide information such as initial state distribution and state transition. Thus, any sequence whose value changes over time can be modelled using a Dynamic Bayesian Network to predict it's future values

In this project, we aim to estimate the respiratory rate over a certain time sequence. This calls for modelling using temporal models. The following hypothesis, describes briefly, the realization of the temporal model to estimate the respiratory rate. The model uses observations from various sensors that vary over time. These observations depict the physiological processes associated with human body. The data comes from the inertial, heart rate, and the temperature measurements of an individual. The inertial data comes from accelerometer, gyroscopic values from a wristband sensor in addition to an ankle sensor, while the heart rate and body temperature values are obtained from the chest strap sensor. A Ridge regression model is trained using these features after performing dimensionality reduction. The resulting predictions are then used in modelling



(a) Generic 2-TBN



(b) Model Implemented

Fig. 1. Implementation of HMM

the states that represent the variations in the input data. These variations prevails as a result of the change in the physiological processes engendered by the different actions performed by the human body. Once the model is able to represent the states that manifest these variations as per the activities performed by the human body, the final goal is to estimate the value and the dynamics of the respiratory rate from the state sequence generated over the time. To achieve this, we maximize the likelihood of an observation given the sequence of the states.

In the sections to follow, we expound on the analysis and implementation of the above-said hypothesis which is organized as follows. Section III describes about the literature survey that enlists most of the related works. Section IV explicates about the Temporal model(Hidden Markov Models) proposal to its implementation. This section also provides a detailed description about various algorithms in Hidden Markov Models (HMM), various parameters specific to this implementation and finally explains about the implementation of HMM to estimate the respiratory rate. Section V provides an idea about methods that do not take temporal dynamics into consideration to estimate the respiratory rate. One such method, Artificial Neural Network(ANN), is discussed in detail along with its implementation. Section VI defines the metric for error on basis of which all the analysis has been made. Within this, discussion is made regarding two most common error metrics, namely, Root Mean Square Error (RMSE) and R-Squared Score (R2). Finally, section VII gives a detailed evaluation of the analyzed test cases for HMM as well as ANN by providing quantifiable details about the same which outlines the performance deviation among a regress-or based model and a temporal based model. Section VII summarizes the report and discusses the potential applications for the proposed method

III. Literature Survey

Hidden Markov Models are well-established models with a wide range of applications. The two main components of the HMM are its hidden states sequence, which encodes abrupt changes in the data, and a set of observation models which model the within-state dynamics of the data. HMMs are often used in conjunction with other machine learning models to improve the overall accuracy [14]. For instance, earlier papers have created hybrid models of HMMs and ANNs/RNNs as in [13] and [16]. But those papers use Neural Network only to refine the parameters of the HMM.
Even in our case where we use Ridge Regression, the problem gets computationally intensive with the increase in number of features. So a simple approach [15] to overcome this problem is to first project the inputs on to a lower dimensional feature space using techniques like Principal Component Analysis (PCA), and then apply the function fitting in the new space.
Most of these statistical models require one assumption [26]: the form of population distribution such as Gaussian distributions or binomial distributions for the output values. We consider the same, and try find a Maximum Likelihood of the observation to the mean of the Gaussian distribution of the the respiratory rates.

IV. Temporal Model based Respiratory Rate Estimation

We propose a Dynamic Bayesian Method using Hidden Markov Models for the estimation of respiratory rate.

A. Hidden Markov Model [2] [10] [22]

A Hidden Markov Model (HMM) is a statistical Markov Model in which the system being modeled is assumed to be a Markov process. It can be considered as a generalization of a mixture model where the hidden variables that control the mixture of components to be selected for each observations, are related through a Markov process. It is the simplest example of temporal models where the future time slice information depends only on the current time and not on the past values. The template for 2-TBN (2-Time Slice Bayesian Network) model as shown in Fig. 1a for a generic HMM, conveys the same information.

In an HMM, each state has a probability distribution over the possible observations. So the sequence of tokens generated by an HMM gives some information about the sequence of states. We model the state-transition probabilities from the available observations. In our model, Fig.1b, the observations are the predictions $Y^{'}$ obtained by passing the 52-dimensional training data (X_t) to a Ridge regression model. The states themselves are unknown, but their observations (i.e. $Y^{'}$) are known and so the probabilities of their transition will affect the future observations. In our implementation, we consider the hidden states(Z) as having 2 observations, that are independent of each other given the states. One of the observation is the scalar respiratory rate(Y) whereas the other observation is the predictions from a Ridge Regression model when 52 dimensional input features are given to it. The Bayesian Network will extend in time, spanning more states, as we gather new measurements. From the design Fig.1b, the observations Y are dependent on the state. So once we get a new reading (i.e. $Y^{'}$) from the Ridge Regression output, we map the state(Z) based on the transition probabilities. Once the state Z is known, we can infer about its second observation Y which yields out the Respiratory Rate. We also consider inputs to an observation from the previous 2 states. This is done to incorporate smoothness in the predicted respiratory rate. Information about previous time states, helps the Markov Network to model gradual transitions from in respiratory rate from one state to another.

1) Inference [11] [10]: Once we develop the required model $\Delta = (A, B, \pi)$, the next step is to efficiently find the probability of output sequence given the model parameters and the observations Y. The probability of observing an output sequence Y is given by:

$$P(Y) = \sum P(Y|X)P(X) \tag{1}$$

Where the sum runs all over the possible hidden nodes of X. This can be done by a forward-backward algorithm The algorithm basically has three steps:
1. Computing forward probabilities
2. Computing backward probabilities
3. Computing smoothed values

At each single observation in the sequence, probabilities to be used for calculations at the next observation are computed.

2) Learning [22] [10]: In learning we estimate the state transition probability A and the output emission probability B that makes the (observation) data more likely. Solving for maximum likelihood parameters using a data set can allow us to effectively train the HMM before implementing maximum likelihood state assignment of a certain data sequence. For our case we are implementing Baum Welch algorithm which uses Expectation Maximization for HMM. In EM Optimization problem, M-set is constrained such that A and B contains valid probabilities. This optimization problem is solved using

Lagrange's multiplier. We will also make use of Forward and Backward algorithms to compute a set of sufficient statistics for our E-step and M-step. The detailed algorithm is mentioned in [15].

3) Recognition [22] [10]: In this, given a set of observations and the model, we have to choose a corresponding sequence of states that are optimal. The Viterbi algorithm, which being a dynamic programming algorithm, is used for finding the most likely sequence of hidden states. It recursively finds optimal solution to the problem of estimating the state sequence of a discrete-time finite-state Markov process. It is just like the forward procedure of inference, except that instead of tracking the total probability of generating the observation, we track the maximum probability and record its corresponding state sequence.

4) Viterbi Algorithm [21] [2]: The Viterbi Algorithm (VA) is a dynamic programming algorithm for finding the most likely sequence of hidden states called the Viterbi path that results in a sequence of observed events. In its most general form the VA can be viewed as a solution to the problem of maximum a posteriori probability (MAP) estimation of a finite state discrete-time Markov process. It requires knowledge of the parameters of the HMM model and a particular output sequence and it finds the state sequence that is most likely to have generated that output sequence. It works by finding a maximum over all possible state sequences.

5) Baum Welch Algorithm [17]: The BaumWelch algorithm is used to find the unknown parameters [24] of an HMM by using the forward-backward algorithm [25]. An HMM can be characterized by the joint probability of a collection of hidden and observed random variables. It relies on the assumption that ith hidden state is independent of the previous hidden states given the (i-1)th hidden variable and the current observation variables depend only on the current hidden state. Given a Markov model with the parameters, $\Delta = [A, B, \pi]$, BaumWelch algorithm [2] finds a local maximum for Δ that maximizes the probability of the observation.

B. Implementation

1) Pre-processing:

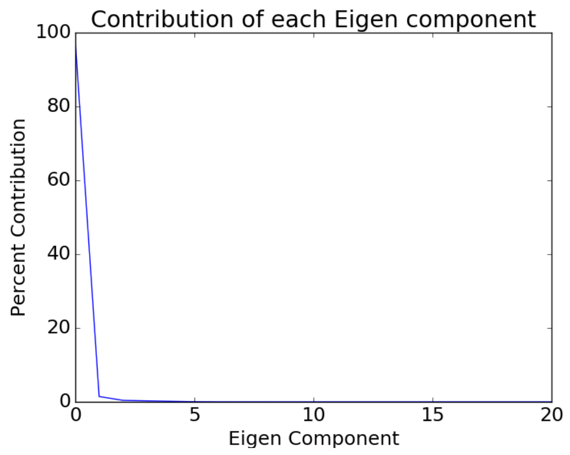
Feature segregation: The Respiratory Rate depends on various factors such as the body temperature, the environmental temperature, the humidity in the environment and the actions the person is doing at any given time. First we applied PCA on all the 52 features directly and tried reducing the dimension.

Remaining features after PCA	RMSE	R2
20	6.54	-0.84
35	6.51	-0.83
40	6.30	-0.71

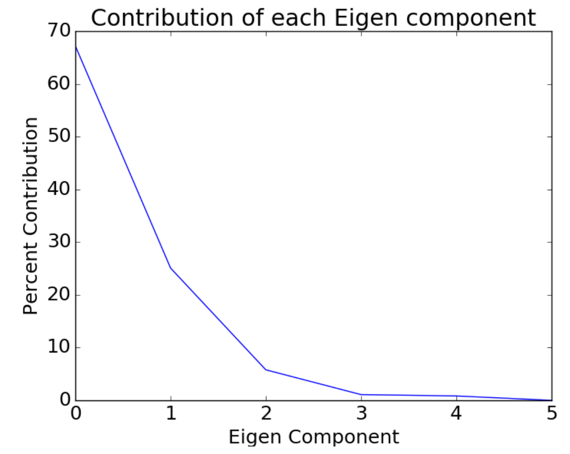
TABLE I. COMPARISON OF PCA ON BASIS OF ERROR METRICS

The Table I contains the RMSE (Root Mean Square Error) and R2 (R-Squared Score) values obtained after performing PCA on all the features together. It is evident from the table that it is not giving good performance. Separately applying PCA to different clusters of similar data is an alternative to this approach.

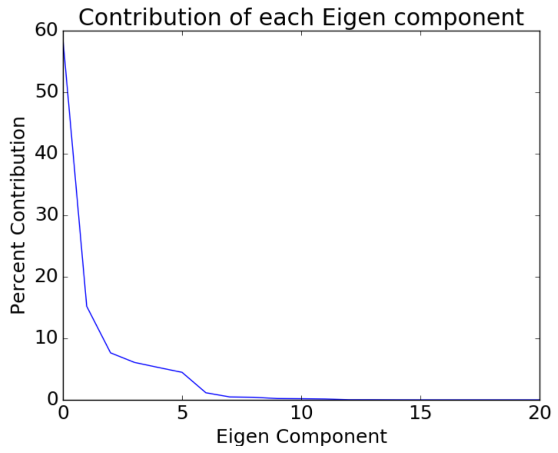
Since each factor may not correlate to each other, we first segregate the data into different groups.



(a) Ankle measurement Contribution to variance



(b) EDA measurement contribution to variance



(c) Wrist measurement contribution to variance

Fig. 2. PCA VS Contribution

Since we know that the external factors like environmental temperature, humidity have no correlation to the IMU data, we can safely seclude them from the rest of the features. Therefore, we group all the 52 features into 3 basic clusters: External features with dimension of 2; features from wristband and ankle sensors with dimension of 48 and features from chestband sensors with dimension of 2. We separately apply PCA on 48 features from the wristband sensors which contains IMU and EDA sensor data since these factors definitely have some correlation to each other and reduce it's dimension to 11.

Principle Component Analysis [12]: The main idea of PCA is to reduce the dimensionality of a data set with many variables correlated with each other, while retaining the variation present in the dataset, up to the maximum extent. PCA does this by transforming the existing set of variables to a new set of variables (called principal components) which are orthogonal. The order of this principal component vectors is such that the the variance in the original vectors decreases as we move down the order. We use scikit tool's explained variance ratio which gives the percentage of variance explained by each feature. We separately apply PCA on the wristband sensors such that 20 dimensional ankle IMU features and 20 dimensional hand IMU features are reduced to 2 and 6 features respectively while the 8 dimensional EDA features are reduced to 3 dimensions. These optimum values are obtained by plotting the number of PCA components along with it's contribution to the variance which is showed in figure 2.

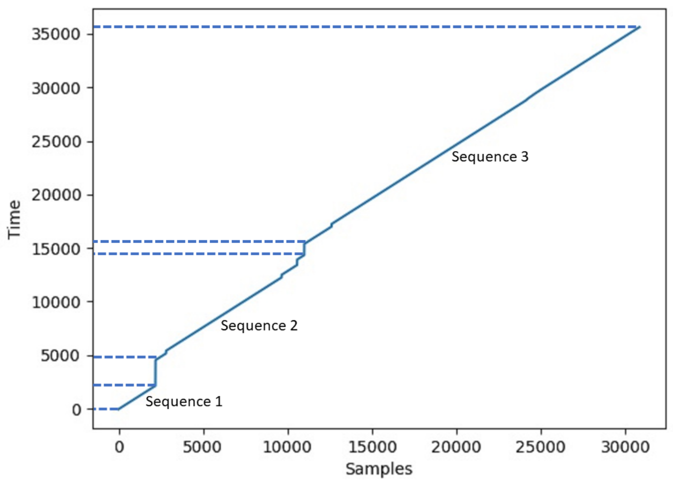
The optimum values for the PCA components are again cross-checked by evaluating the RMSE and R2 score for each case. Again, the same results are obtained with 2 features for ankle IMU, 6 features for hand IMU and 3 features for EDA. This is shown in table II

Ankle IMU features	**Hand IMU features**	**EDA features**	**RMSE**	**R2**
2	**6**	**3**	**3.064**	**0.325**
4	8	5	3.5	0.1
3	7	4	4.72	-0.6

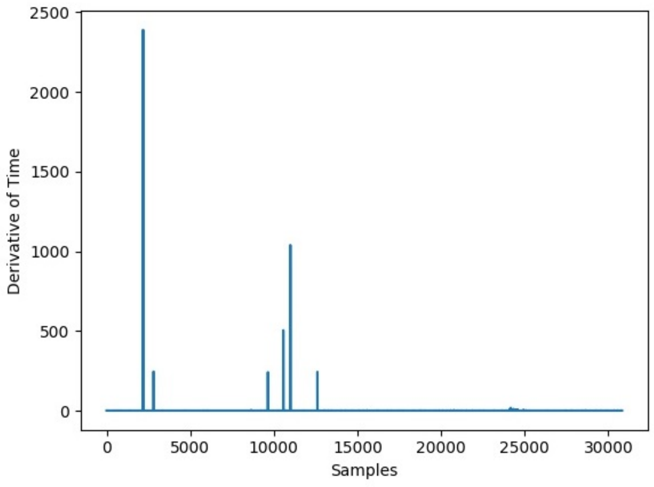
TABLE II. COMPARISON OF PCA WITH ERROR MATRIX AFTER FEATURE SEGREGATION

On total, we have 15 features where 3 features come from the EDA output along with 8 features from chest-band and ankle sensors and 4 features from external variables.

Time Sequence Processing: The data on each day contains several sudden jumps on it's time feature which makes it less sequential. To make it sequentially increasing, the data is divided into separate sets whenever a sudden time jump occurs. Each set is trained one after the other on the same model. Figure 3a shows the variation of time per samples of a particular day and figure 3b shows the derivative of corresponding time which highlights the sudden increase in time with large impulses. Each linear element is treated as a separate sample while training. If the time gap is small, the missing values are interpolated.



(a) Time Vs Samples



(b) Derivative of Time Vs Samples

Fig. 3. Time Sequence Pre-processing

2) Hyper-parameter selection: A plot is made on the RMSE and the R2 value obtained by varying the number of hidden states, without changing the test data. The different states represent different actions the person wearing the sensors could be doing, and since the total number of actions are not known, the number of states that resulted in the least RMSE and maximum R2 score is selected. The analysis is shown in table III

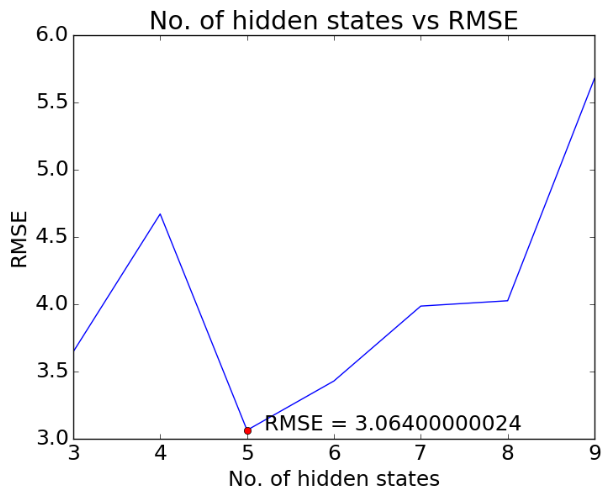
From the table it is evident that with 5 hidden states we get the minimum root mean squared error and maximum R2 score. The same information is delineated by Fig 4a and Fig 4b.

The Fig 4a represents the variation of RMSE with the no. of hidden states where as Fig 4b represents the variation of R2 score with the no. of hidden states.

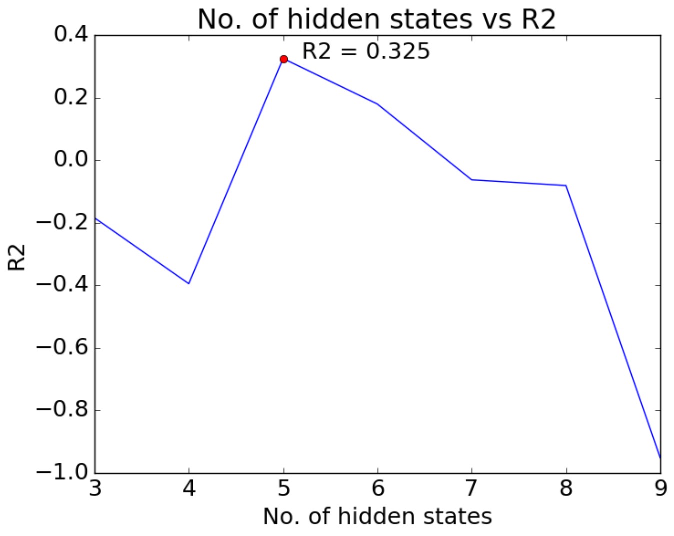
The figure 5 shows the variation of the 5 states over a sequence of 80 samples. Each state represents an action and when the action is changed, the model tries to switch to the best possible state.

No. of Hidden States	RMSE	R2
3	3.647	-0.184
4	4.671	-0.394
5	**3.064**	**0.325**
6	3.429	-0.179
7	3.986	-0.062
8	4.026	-0.081
9	5.678	0.090

TABLE III. COMPARISON OF NUMBER OF HIDDEN STATES WITH ERROR METRICS



(a) No. of hidden states vs RMSE



(b) No. of hidden states vs R2

Fig. 4. Hyper-parameter Estimation: States

3) Implementation of HMM [10] [7]: We use Ridge Regression to get an estimate of the output and refine this

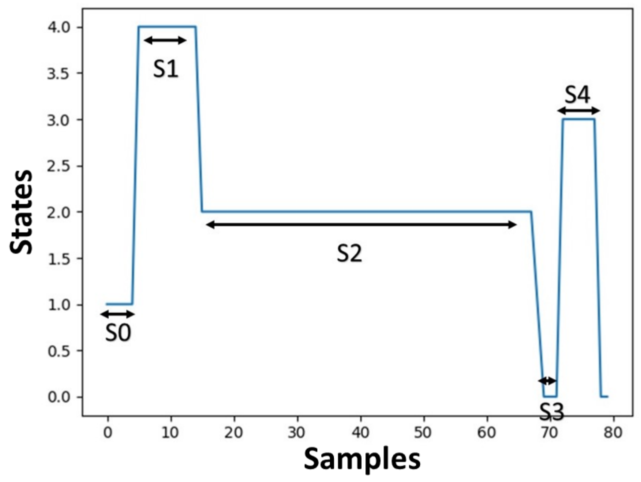


Fig. 5. States variation over Samples

value using Dynamic Bayesian approach. The advantage of this procedure is that the feature information is captured by Regression whereas temporal information is captured by Dynamic Network. We train the Regression model on the data X (52 dimensional) from first three days (all the posts) and test it on the fourth day. This yields us the output Y', which not only gives an estimate of the actual output but also helps in reducing the dimensionality of the data for optimization. We now pass this output Y' to a Gaussian Hidden Markov Model to get the series of hidden states Z_i to which each time step belongs. This series of hidden states coupled with the corresponding respiratory values, forms the training data set for probability calculation. The trained Regression model and these values are saved for future evaluation.
Now, when we get the new test data, we employ the same Regression model to get the intermediate output Y'. This Y' is used to compute the corresponding hidden state using the Baum Welch algorithm. Now again VA is used to find the optimum sequence of hidden states after which we perform a Maximum Likelihood Estimation (MLE) of the observation to the hidden states by scanning through the entire training data and counting the number of instances of particular occurrences of the given observation. We discretize the Respiratory Rate by flooring it to the nearest integer. This is done so as to ease probability calculation.

V. Regression based Respiratory Rate Estimation

A. Description

Regression based models are models that make prediction with supervised learning over the features under consideration. Such models have internal parameters which are updated as the models are trained over the features. Such trained models are used to make estimation of the dependent variables for future test cases. However, such models do not take into consideration the temporal dynamics of the features while making the prediction. Various such regression based methods are available, for instance, Linear Regression, Ridge Regression, Lasso regression, Artificial Neural Networks etc. In our specific case we used Artificial Neural Network which can

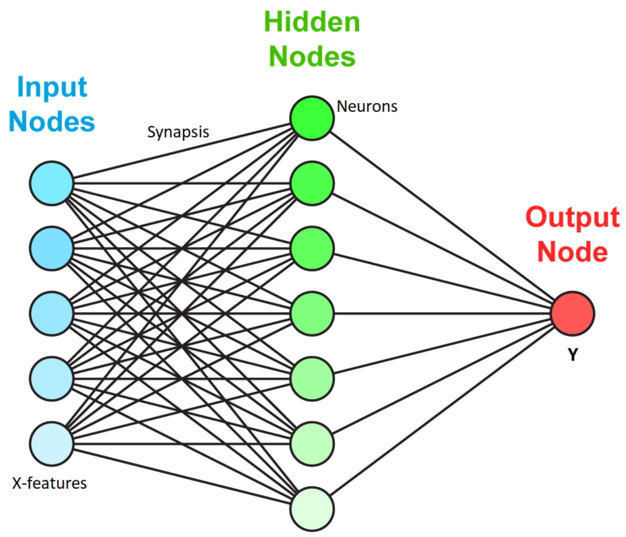


Fig. 6. Artificial Neural Network

efficiently capture the complex non linear relationship between dependent and independent variables.

B. Implementation: ANN [19]

Neural Networks are non-linear statistical data modeling tools. A general schematic of ANN is as shown in Fig. 6. Since the data we are dealing with, varies non linearly with the output(respiratory rate), Neural Networks can be considered as a potential candidate for model selection. To reduce the dimensions of the data, we applied PCA to the training data and reduced the number of features to 20. This reduces the number of computations the network has to perform.

1) Pre-processing [12]: We used the same pre-processing technique as mentioned in section IV-B. Specifically, we carried out dimensionality reduction in two stages: First we manually computed the correlation between all the features and eliminated highly correlated features; Second we passed this reduced number of features as an input to PCA for further reduction in dimension. For the former manual method of reduction,we found the correlation between each features using the toolbox numpy[13]. We considered values of correlation of 0.9 and above to be highly correlated. We found a set of 6 features to be highly correlated with each other and another set of 4 features with high correlation. So basically, instead of considering all these 10 features, we took only one feature from each set to predict the output reducing the number of features from 52 to 44. The purpose behind doing manual reduction using correlation was to eliminate the need to perform additional computations by PCA. Further, we used PCA to ensure optimum dimensionality reduction. We found the best possible PCA reduced feature by manual trial and error. The number of features found to give least possible error were 20.

We standardized the features by removing the mean and scaling the variance to unit value. Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set. Mean and standard deviation are then stored to be used on later data using the transform method. If a feature has a variance that is orders of magnitude larger than others, then it might dominate a node and make the network unable to learn the features correctly.

2) Hyper-parameter selection [4]: Hyper parameters are parameters whose values are kept fixed before the learning process and will not be changed during the whole process. Given these hyper parameters, the algorithm learns the value of parameters from the data. Since the ANN is being used as a regressor, the number of nodes in the output layer is l. The number of nodes in the input layer is equal to the number of features to be trained (in this case 20).

Hidden Layer: Initially, we kept the number of neurons to be in between the number of inputs and number of outputs. Further we also tested the network for higher number of neurons in the hidden layer. Finally we set the number of neurons to 200 with minimum possible RMSE and highest R2-score. This is evident from table IV that illustrates RMSE and R2-score for various hidden layer values.

Neurons in Hidden Layer	**MSE**	**RMSE**	**R2**
200	28.62	5.35	-0.31
220	28.51	5.34	-0.34
250	30.01	5.49	-0.39
150	26.88	5.18	-0.35
130	29.05	5.39	-0.36
40	28.72	5.36	-0.47

TABLE IV. Number of Neurons

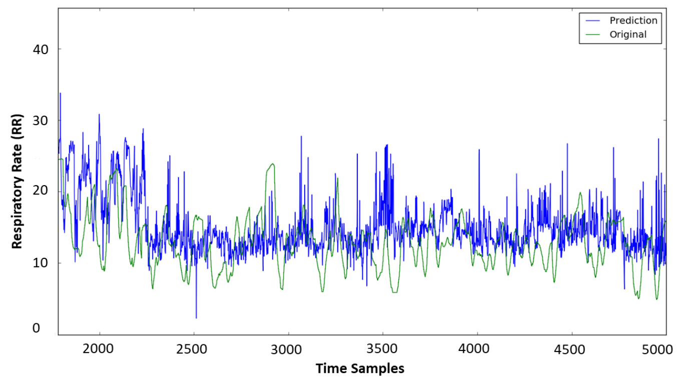
Batch Size: The gradient descent method operates on a small batch which is a fraction of the training data. There is a trade off between the batch size and the computational complexity of the Network. We have chosen the batch size as 100 for our network.

Epoch: An epoch refers to one forward pass and one backward pass of all the training examples. In our case, the number of epochs is taken as 100.

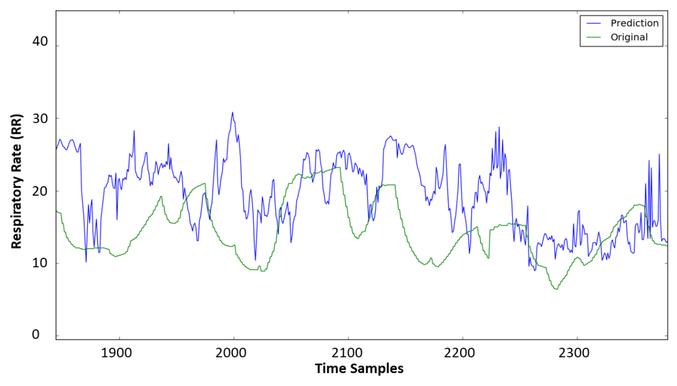
Learning Rate: The learning rate is chosen automatically by the RMSProp optimizer in Keras package.

Activation function: Since the value of some of the features are not in order of same magnitudes, the sigmoid activation function can cause the problem of vanishing or exploding gradients. Hence, we use the ReLU (Rectied Linear Unit) activation function which not only solves this problem but also makes the computation efcient.

3) Implementation of ANN: We first used PCA to reduce the dimensionality of the data before passing it to the ANN. We consider one hidden layer for the network with 200 neurons. Since the training is done in batches for a large amount of data, we set the batch size of 100. Also the epochs is set to 100 thus increasing the number of forward and backward passes through the network.

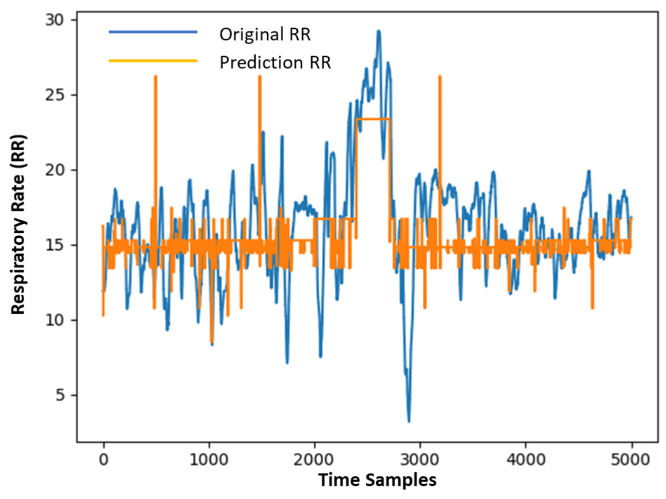


(a) All Samples

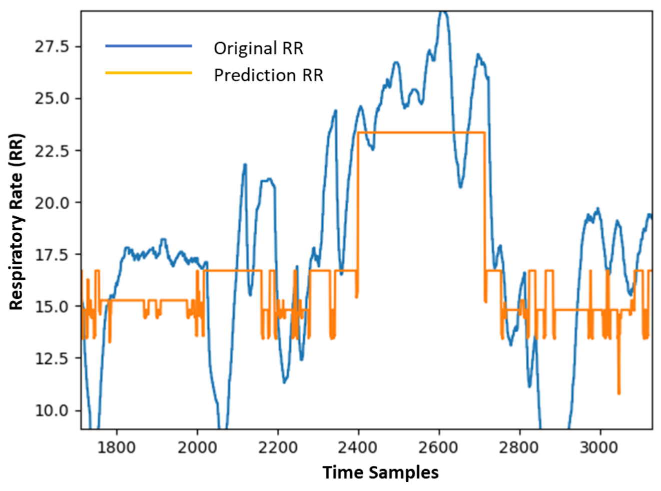


(b) Zoomed

Fig. 7. Results: ANN



(a) All Samples



(b) Zoomed

Fig. 8. Results: HMM

VI. Error Metric

1) Root Mean Square Error: The RMSE is a quadratic score that measures the average magnitude of the error. It is indifferent to the direction of the error. RMSE gives relatively high weight to large errors.

2) R Squared Error: R squared score or coefcient of determination is a measure of how well the future samples are likely to be predicted. It is calculated by the following equation:

$$R^2(y_i, \hat{y}_i) = 1 - \frac{U}{V} \tag{2}$$

where,

$$U = \sum_{i=0}^{n-1} (y_i - \hat{y}_i)^2 \tag{3}$$

$$V = \sum_{i=0}^{n-1} (y_i - \bar{y}_i)^2 \tag{4}$$

$$\bar{y}_i = \sum_{i=0}^{n-1} \frac{y_i}{n} \tag{5}$$

VII. Evaluation

A. *ANN: Test Case with Training Data*

For the purpose of testing, we divided the entire data into training and testing parts. The training data consisted of 80 percent of the data which is 92,494 samples while the remaining 20 percent which is 23,123 samples are saved for testing. The training data is divided into train and cross-validation set. We now train the model on this new training data set and validate it on the cross validation set. The model is tweaked for different values of hyper parameter. We varied the number of neurons in the hidden layer and obtained the RMSE and R-Squared error. This is shown in Table IV.

Fig. 8 shows the performance of neural network with hidden layer size as 200. Now using this value as the hidden layer size, we train a Neural Network on the training data(which is

80 percent of the entire data) and nd its error on the testing set.

B. HMM: Test Case

The HMM model is trained on 1,05,528 samples which comes around 91.2% of the total data and tested on 10,089 samples which comes around 8.8% The figure 8 shows the test results for HMM when the trained model was tested with the 8.2% of the total samples. Figure 8a shows the prediction on the whole data while Figure 8b is it's zoomed version

C. Result and Analysis

	Neural Network	**HMM**
RMSE	5.18	3.064
R2	-0.35	0.325

TABLE V. Performance Comparison of ANN and HMM

The table V shows the comparison between the two models that we used to estimate the Respiratory rate. A qualitative comparison is made among both the methods. HMM model is efficiently able to obtain a low RMSE and high R2 score with a significant reduction of 40% in the RMSE value and an increase of 192% in the R2 score.

Both the models are tested on the samples from day 4 data. The following reasons can confirm the better efficiency of HMM over Neural network:

1) The time dependencies are captured in an HMM model where it predicts new values depending on the previous observations. But in the neural network model, we are not incorporating the time dependencies which fails to follow the sequential arrangement of data over time. Since we are trying to predict the respiratory rate of an individual which changes over time, HMM model has the ability to adapt over each time step by changing it's model parameters such that each output value has some dependence on the past history.

2)HMM is a generative, probabilistic model while Neural network is a deterministic, discriminative model. HMM tries to model the process generating the training sequences, or more precisely, the distribution over the sequences of observations. On the other hand, a neural network leans a mapping from the input space into the output space and once the network has been trained, i.e. the values of all weights and thresholds have been determined, the response of the network to every input is deterministic. Since our data which depends on time can be learned well with probability, HMM is relatively a better method for prediction.

VIII. Toolboxes

The entire project was implemented in Python 2.7
The python packages we used are:-
- numpy [9] (for numerical computations)
- matplotlib [5] (for graph plotting)
- scikit-learn [3] (for implementing regression model)
- pandas [6] (to handle data)
- keras [4] with tensorflow backend (for Neural Networks)
- scipy [8] (for optimization)
- hmmlearn [1] from sckit-learn (for implementing the Hidden Markov Model)

IX. Conclusion

In this report, we implemented two different methods to predict the respiratory rate of an individual using multi-dimensional features. Artificial Neural Network is able to predict the output efficiently to some extent and is able to adapt to some of the variations present in the input. On the other hand, Hidden Markov Model is able to predict the respiratory rate efficiently with the least error. By incorporating the time dependence while predicting the output, the model is able to adapt more towards the variations in the input features thereby giving more accurate predictions.

References

[1] http://hmmlearn.readthedocs.io/en/latest/.

[2] http://modelai.gettysburg.edu/2017/hmm/description.html.

[3] http://scikit-learn.org/stable/.

[4] https://keras.io/getting-started/sequential-model-guide/.

[5] https://matplotlib.org.

[6] https://pandas.pydata.org.

[7] https://towardsdatascience.com/hidden-markov-models-for-time-series-classification-basic-overview-a59b74e5e65b.

[8] https://www.scipy.org.

[9] http://www.numpy.org/.

[10] http://www.shokhirev.com/nikolai/abc/alg/hmm/hmm.html.

[11] www.cs.cmu.edu/ guestrin/class/10701-s07/handouts/recitations/hmm-inference.pdf.

[12] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[13] Yoshua Bengio, Renato De Mori, Giovanni Flammia, and Ralf Kompe. Global optimization of a neural network-hidden markov model hybrid. *IEEE transactions on Neural Networks*, 3(2):252–259, 1992.

[14] Saurabh Bhardwaj, Smriti Srivastava, S Vaishnavi, and JRP Gupta. Chaotic time series prediction using combination of hidden markov model and neural nets. In *Computer Information Systems and Industrial Management Applications (CISIM), 2010 International Conference on*, pages 585–589. IEEE, 2010.

[15] Jeff A Bilmes et al. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. *International Computer Science Institute*, 4(510):126, 1998.

[16] Rohitash Chandra and Christian W Omlin. Evolutionary training of hybrid systems of recurrent neural networks and hidden markov models. *World Academy of Science, Engineering and Technology*, 3:58–63, 2006.

[17] Shai Fine, Yoram Singer, and Naftali Tishby. The hierarchical hidden markov model: Analysis and applications. *Machine learning*, 32(1):41–62, 1998.

[18] Zoubin Ghahramani. An introduction to hidden markov models and bayesian networks. *International journal of pattern recognition and artificial intelligence*, 15(01):9–42, 2001.

[19] Anil K Jain, Jianchang Mao, and K Moidin Mohiuddin. Artificial neural networks: A tutorial. *Computer*, 29(3):31–44, 1996.

[20] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[21] H-L Lou. Implementing the viterbi algorithm. *IEEE Signal processing magazine*, 12(5):42–52, 1995.

[22] Daniel Ramage. Hidden markov models fundamentals. *Lecture Notes. http://cs229. stanford. edu/section/cs229-hmm. pdf*, 2007.

[23] Iead Rezek and Stephen J Roberts. Ensemble hidden markov models for biosignal analysis. In *Digital Signal Processing, 2002. DSP 2002. 2002 14th International Conference on*, volume 1, pages 387–391. IEEE, 2002.

[24] Stephen Tu. Derivation of baum-welch algorithm for hidden markov models, 2015.

[25] Xueying Zhang, Yiping Wang, and Zhefeng Zhao. A hybrid speech recognition training method for hmm based on genetic algorithm and baum welch algorithm. In *Innovative Computing, Information and Control, 2007. ICICIC'07. Second International Conference on*, pages 572–572. IEEE, 2007.

[26] Yingjian Zhang. *Prediction of financial time series with Hidden Markov Models*. PhD thesis, Applied Sciences: School of Computing Science, 2004.