ONLINE BAYESIAN TRANSFER LEARNING FOR SEQUENTIAL DATA MODELING

Priyank Jaini[1], **Zhitang Chen**[4], **Pablo Carbajal**[1,5], **Edith Law**[1], **Laura Middleton**[2], **Kayla Regan**[2], **Mike Schaekermann**[1], **George Trimponias**[4], **James Tung**[3], **Pascal Poupart**[1,5]
pjaini@uwaterloo.ca, chenzhitang2@huawei.com, pablo@veedata.io,
{edith.law,lmiddlet,kregan}@uwaterloo.ca,
g.trimponias@huawei.com,
{mschaekermann,james.tung,ppoupart}@uwaterloo.ca
[1] David R. Cheriton School of Computer Science, University of Waterloo, Ontario, Canada
[2] Department of Kinesiology, University of Waterloo, Ontario, Canada
[3] Dept. of Mechanical and Mechatronics Engineering, University of Waterloo, Ontario, Canada
[4] Noah's Ark Laboratory, Huawei Technologies, Hong Kong, China
[5] Veedata Inc., Kitchener, Ontario, Canada

ABSTRACT

We consider the problem of inferring a sequence of hidden states associated with a sequence of observations produced by an individual within a population. Instead of learning a single sequence model for the population (which does not account for variations within the population), we learn a set of basis sequence models based on different individuals. The sequence of hidden states for a new individual is inferred in an online fashion by estimating a distribution over the basis models that best explain the sequence of observations of this new individual. We explain how to do this in the context of hidden Markov models with Gaussian mixture models that are learned based on streaming data by online Bayesian moment matching. The resulting transfer learning technique is demonstrated with three real-word applications: activity recognition based on smartphone sensors, sleep classification based on electroencephalography data and the prediction of the direction of future packet flows between a pair of servers in telecommunication networks.

1 INTRODUCTION

In several application domains, data instances are produced by a population of individuals that exhibit a variety of different characteristics. For instance, in activity recognition, different individuals might walk or run with different gait patterns. Similarly, in sleep studies, different individuals might exhibit different patterns for the same sleep stages. In telecommunication networks, software applications might generate packet flows between two servers according to different patterns. In such scenarios, it is tempting to treat the population as a homogeneous source of data and to learn a single average model for the population. However, this average model will perform poorly in recognition tasks for individuals that differ significantly from the average. Hence, there is a need for transfer learning techniques that take into account the variations between individuals within a population.

We consider the problem of inferring a sequence of hidden states based on a sequence of observations produced by an individual within a population. Our first contribution is an online Bayesian moment matching technique to estimate the parameters of a hidden Markov model (HMM) with observation distributions represented by Gaussian mixture models (GMMs). This approach allows us to learn separate basis models for different individuals based on streaming data. The second contribution is an unsupervised online technique that infers a probability distribution over the basis models that best explain the sequence of observations of a new individual. The classification of hidden states can then be refined in an online fashion based on the individuals that most resemble the new individual. Furthermore, since the basis models are fixed at classification time and we only learn the weight of each model, good classification accuracy can be obtained more quickly as the stream of observations of the new individual are processed. The third contribution of this work is

the demonstration of this approach across different real-world applications, which include activity recognition, sleep classification and the prediction of packet flow direction in telecommunication networks.

The paper is organized as follows. Section 2 reviews some related work on transfer learning. Section 3 provides some background regarding hidden Markov models Bayesian Moment Matching algorithm Gaussian mixture models. Section 4 describes the proposed online transfer learning technique. Section 5 illustrates the transfer learning technique in three real-world tasks: activity recognition, sleep stage classification and flow direction prediction. Finally, Section 6 concludes the paper and discusses directions for future work.

2 Related Work

There is a large literature on transfer learning (Pan & Yang, 2010; Taylor & Stone, 2009; Shao et al., 2015; Cook et al., 2013). Depending on the problem, the input features, the output labels or the distribution over the features and the labels may be different for the source and target domains. In this work, we assume that the same input features are measured and the same output labels are inferred in the source and target domains. The main problem that we consider is *subject variability* within a population of individuals, which means that different individuals exhibit different distributions over the features and the labels. The problem of subject variability has been studied in several papers. Chieu et al. (2006) describe how to augment conditional random fields with a subject hidden variable to obtain a mixture of conditional random fields that can naturally infer a distribution over the closest subjects in a training population when inferring the activities of a new individual based on physiological data. Rashidi & Cook (2009) proposed a data mining technique with a similarity measure to facilitate the transfer of activity recognition across different people. Chattopadhyay et al. (2011) describe a similarity measure with an intrinsic manifold that preserve the topology of surface electromyography (SEMG) while mitigating distributional differences among individuals. Zhao et al. (2011) proposed a transfer learning technique that starts by training a decision tree to recognize the activities of a user based on smartphone accelerometry. The decision tree is gradually adjusted to a new user by a clustering technique that successively re-weights the training data based on the unlabeled data of the new individual. These approaches mitigate subject variability by various *offline* transfer learning techniques. In contrast, we propose an *online* transfer learning technique since the applications that we consider exhibit sequences of observations that arrive in a streaming fashion and therefore require an online technique that can infer the hidden state of each observation as it arrives.

In the next section, we describe an online transfer learning technique for hidden Markov models with Gaussian mixture models. The approach learns different transition and emission models for each individual in the training population. Those models are then treated as basis models to speed up the online learning process for new individuals. More specifically, a weighted combination of the basis models is learned for each new individual. This idea is related to boosting techniques for transfer learning (Dai et al., 2007; Yao & Doretto, 2010; Al-Stouhi & Reddy, 2011) that estimate a weighted combination of base classifiers. However, note that we focus on sequence modeling problems where the classes of consecutive data points are correlated while transfer learning by boosting assumes that the data points are identically and independently distributed.

3 Background

In this section, we give a brief overview of hidden Markov models (HMMs) and review the Bayesian moment matching (BMM) algorithm in detail with an example. We will use both HMMs and BMM subsequently in our transfer learning algorithm described in Section 4.

3.1 Hidden Markov Models

In a hidden Markov model (HMM), each observation X_t is associated with a hidden state Y_t. The Markov property states that the current state depends only on the previous state. HMMs have been widely used in domains involving sequential data like speech recognition, activity recognition, natural language processing etc. An HMM is represented by two distributions

- **Transition distribution**: The transition distribution models the change in the value of the hidden state over time. The distribution over the current state Y_t given that the previous state is $Y_{t-1} = j$ is denoted by $\theta_j = \Pr(Y_t|Y_{t-1} = j)$ where $\theta_j = \{\theta_{1j}, ..., \theta_{Nj}\}$, N is the total number of states and $\theta_{ij} = \Pr(Y_t = i|Y_{t-1} = j)$.
- **Emission distribution**: The emission distribution models the effect of the hidden state on the observation X_t at any given time t and is given by $\Pr(X_t|Y_t)$. In this work, we model the emission distribution as a mixture of Gaussians with M components, i.e., $\Pr(X_t|Y_t = j) = \sum_{i=1}^{M} w_i \mathcal{N}\big(\mathbf{X}_t; \boldsymbol{\mu}_i^j, \Sigma_i^j\big)$

In this paper, we will first estimate the parameters of the transition and emission distributions by Bayesian learning from a set of source domains (individuals). Subsequently, we will use these distributions as basis functions when estimating the transition and emission distributions of a target domain in which we wish to predict the hidden state for each observation. Parameter learning of an HMM using Bayesian learning is done by calculating the posterior over the parameters given a prior distribution.

$$\Pr\Big(\Theta, \Phi, Y_t = j|X_t, Y_{t-1} = i\Big) \propto \overbrace{\Pr(X_t|Y_t = j)}^{\text{Emission distribution}} \overbrace{\Pr(Y_t = j|Y_{t-1} = i)}^{\text{Transition Probability}} \overbrace{\Pr(\Theta, \Phi, Y_{t-1} = i|X_{1:t-1})}^{\text{Prior for } t-1}$$

$\forall j \in \{1, 2, ..., N\}$ where Θ and Φ parametrize the transition and emission distributions respectively.

3.2 Bayesian Moment Matching Algorithm

The Bayesian moment matching (BMM) algorithm for Gaussian Mixture Models was proposed by Jaini & Poupart (2016); Jaini et al. (2016). Exact Bayesian learning of mixture models based on streaming data is intractable because the number of terms in the posterior after observing each observation increases exponentially. BMM circumvents this issue by projecting the distribution of the exact posterior P on a tractable family of distributions $\tilde{P}$ by matching a set of sufficient moments. In this section, we give a brief overview of the BMM algorithm with an example.

Note that Variational Bayes (VB) and Markov Chain Monte Carlo (MCMC) techniques can also be used for approximate Bayesian learning as an alternative to BMM. However, MCMC is difficult to run in an online fashion. A recent comparison by Omar Omar (2016) showed that BMM achieves better results than online Vational Bayes (oVB) Sato (2001) and Stochastic Variational Inference (SVI) Wang et al. (2011) in the context of topic modeling. BMM was also shown to work better than other online techniques in several papers Rashwan et al. (2016); Hsu & Poupart (2016); Jaini et al. (2016). This is due to the fact that BMM is naturally online and therefore does not require mini-batches. In contrast, in order to run in an online fashion Variational Bayes requires mini-batches and a decreasing learning rate, however the size of the mini-batches and the decay procedure for the learning rate require some fine tuning. In general, the use of mini-batches always leads to some information loss since data in previous mini-batches is not accessible. BMM does not suffer from this type of information loss and there is no batch size nor learning rate to fine tune. Hence, we will adapt BMM to transfer learning in this work.

Let $\mathbf{X}^{1:n}$ be a set of d-dimensional *i.i.d* observations following $\Pr(\mathbf{X}|\Theta) = \sum_{i=1}^{M} w_i \mathcal{N}\big(x; \boldsymbol{\mu}_i, \Lambda^{-1}\big)$ where $\Theta = \{(w_1, \boldsymbol{\mu}_1, \Lambda_1^{-1}), (w_2, \boldsymbol{\mu}_2, \Lambda_2^{-1}), ...(w_M, \mu_M, \Lambda_M^{-1})\}$ and $\boldsymbol{M}$ is known.

We choose the prior as a product of a Dirichlet distribution over the weights $\mathbf{w}$ and M Normal-Wishart distributions corresponding to the parameters $(\boldsymbol{\mu}, \Lambda^{-1})$ of each Gaussian component. Such a prior forms a conjugate probability pair of the likelihood and is hence desirable. Concretely, $P_0(\Theta) = Dir(\mathbf{w}|\boldsymbol{\alpha}) \prod_{i=1}^{M} \mathcal{NW}(\boldsymbol{\mu}_i, \Lambda_i|\boldsymbol{\delta}_i, \kappa_i, \mathbf{W}_i, \nu_i)$ where $\mathbf{w} = (w_1, w_2, ..., w_M)$, $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_M)$, $\mathbf{W}$ is a symmetric positive definite matrix, $\kappa > 0$ is real, $\boldsymbol{\delta} \in \mathbb{R}^d$ and $\nu > d - 1$ is real. The posterior $P_1(\Theta|\mathbf{X}_1)$ after observing the first data point $\mathbf{X}_1$ is given by

$$\begin{aligned} P_1(\Theta|\mathbf{X}_1) &\propto P_0(\Theta) \Pr(\mathbf{X}_1|\Theta) \\ &\propto Dir(\mathbf{w}|\boldsymbol{\alpha}) \prod_{i=1}^{M} \mathcal{NW}(\boldsymbol{\mu}_i, \Lambda_i|\boldsymbol{\delta}_i, \kappa_i, \mathbf{W}_i, \nu_i) \sum_{j=1}^{M} w_j \mathcal{N}\big(\mathbf{X}_1; \boldsymbol{\mu}_j, \Lambda_j^{-1}\big) \end{aligned}$$

Since a Normal-Wishart distribution is a conjugate prior for a Normal distribution with unknown mean and precision matrix, $\mathcal{NW}(\boldsymbol{\mu}_i, \Lambda_i|\boldsymbol{\delta}_i, \kappa_i, \mathbf{W}_i, \nu_i)\mathcal{N}\big(\mathbf{X}_1; \boldsymbol{\mu}_i, \Lambda_i^{-1}\big) =$

$c\mathcal{NW}(\boldsymbol{\mu}_i, \Lambda_i|\hat{\boldsymbol{\delta}}_i, \hat{\kappa}_i, \hat{\mathbf{W}}_i, \hat{\nu}_i)$ where c is some constant. Similarly, $w_j Dir(\mathbf{w}|\alpha_1, \alpha_2, .., \alpha_j, .., \alpha_M) = kDir(w_1, w_2, ..., w_M|\alpha_1, \alpha_2, ..\hat{\alpha}_j .., \alpha_M)$ where k is some constant. Therefore, $P_1(\Theta|\mathbf{X}_1)$ is

$$P_1(\Theta|\mathbf{X}_1) = \frac{1}{Z}\sum_{j=1}^{M}\left(c_j Dir(\mathbf{w}|\hat{\boldsymbol{\alpha}}_j)\mathcal{NW}(\boldsymbol{\mu}_j, \Lambda_j|\hat{\boldsymbol{\delta}}_j, \hat{\kappa}_j, \hat{\mathbf{W}}_j, \hat{\nu}_j)\prod_{i\neq j}^{M}\mathcal{NW}(\boldsymbol{\mu}_i, \Lambda_i|\boldsymbol{\delta}_i, \kappa_i, \mathbf{W}_i, \nu_i)\right)$$

where $\hat{\boldsymbol{\alpha}}_j = (\alpha_1, \alpha_2, .., \hat{\alpha}_j, .., \alpha_M)$ and Z is the normalization constant. The equation above suggests that the posterior is a mixture of product of distributions where each product component in the summation has the same form as that of the family of distributions of the prior $P_0(\Theta)$. It is evident that the terms in the posterior grow by a factor of *M* for each iteration, which is problematic. The Bayesian moment matching algorithm approximates this mixture $P_1(\Theta)$ with a single product of Dirichlet and Normal-Wishart distributions $\tilde{P}_1(\Theta)$ by matching all the sufficient moments of P_1 with $\tilde{P}_1$ which belongs to the same family of distributions as the prior:

$$\tilde{P}_1(\Theta) = Dir(\mathbf{w}|\boldsymbol{\alpha}^1)\prod_{i=1}^{M}\mathcal{NW}(\boldsymbol{\mu}_i, \Lambda_i|\boldsymbol{\delta}_i^1, \kappa_i^1, \mathbf{W}_i^1, \nu_i^1)$$

We evaluate the parameters $\boldsymbol{\alpha}^1, \boldsymbol{\delta}_i^1, \kappa_i^1, \mathbf{W}_i^1, \nu_i^1 \;\; \forall i \in \{1, 2, .., M\}$ by matching a set of sufficient moments of $\tilde{P}_1(\Theta)$ with $P_1(\Theta)$. The set of sufficient moments in this case is $S = \{\boldsymbol{\mu}_j, \boldsymbol{\mu}_j\boldsymbol{\mu}_j^T, \boldsymbol{\Lambda}_j, \Lambda_{j_{km}}^2, w_j, w_j^2\} \; \forall j \in 1, 2, ..., M$ where $\Lambda_{j_{km}}^2$ is the $(k, m)^{th}$ element of the matrix $\boldsymbol{\Lambda}_j$. The expressions for sufficient moments are given by $\mathbb{E}[g] = \int_\Theta g P_1(\Theta)d(\Theta)$. The parameters of $\tilde{P}_1$ can be computed from the following set of equations

$$\begin{aligned}
\mathbb{E}[w_i] &= \frac{\alpha_i}{\sum_j \alpha_j}; & \mathbb{E}[w_i^2] &= \frac{(\alpha_i)(\alpha_i + 1)}{\left(\sum_j \alpha_j\right)\left(1 + \sum_j \alpha_j\right)} \\
\mathbb{E}[\boldsymbol{\Lambda}] &= \nu\mathbf{W}; & Var(\boldsymbol{\Lambda}_{ij}) &= \nu(\mathbf{W}_{ij}^2 + \mathbf{W}_{ii}\mathbf{W}_{jj}) \\
\mathbb{E}[\boldsymbol{\mu}] &= \boldsymbol{\delta}; & \mathbb{E}[(\boldsymbol{\mu} - \boldsymbol{\delta})(\boldsymbol{\mu} - \boldsymbol{\delta})^T] &= \frac{\kappa + 1}{\kappa(\nu - d - 1)}\mathbf{W}^{-1}
\end{aligned}$$

Using this set of equations, the exact posterior $P_1(\Theta)$ can be approximated with $\tilde{P}_1(\Theta)$. This posterior will then be the prior for the next iteration and we keep following the steps above iteratively to finally have a distribution $\tilde{P}_n(\Theta)$ after observing a stream of data $\mathbf{X}^{1:n}$. The estimate is $\hat{\Theta} = \mathbb{E}[\tilde{P}_n(\Theta)]$. The exact calculations for the Bayesian Moment Matching algorithm are given in appendix A.

4 Transfer Learning using BMM

In this section, we first motivate the need for an online transfer learning algorithm for sequential data modeling and then explain in detail the different steps of the algorithm. The complete algorithm is given in Alg. (1).

4.1 Motivation

Several applications produce data instances from a population of individuals that exhibit a variety of different traits. For example, for the task of activity recognition, different individuals will have different gait patterns despite the fact that they are performing the same activity (e.g., walking, running, standing, etc.). Therefore, it is problematic to make predictions in such domains by considering the population to be homogeneous; however, every population will have individuals resembling each other in some characteristics. This suggests that we can use individuals in a population to make predictions about similar individuals by identifying those individuals who closely resemble each other. However, identifying individuals with similar traits is not straightforward. Alternatively, weights can be assigned to each individual in a population based on a target individual (individual on whom predictions are to be made). All those individuals who resemble closely the target individual will receive higher weights than those with dissimilar traits. Subsequently, predictions about the behavior of the target individual will be based mostly on the observed behavior of the similar individuals.

Our transfer learning algorithm addresses precisely these issues. It has three main steps - first, it learns a model (transition and emission distributions) for each source domain (or individual in a population) that best explains the observations of that source domain. Next, given a target domain (or target individual), it identifies those individuals that closely resemble the target individual by estimating a basis weight associated to each source domain. A higher weight for a source domain implies that the corresponding individual resembles more closely the target individual. Finally, it predicts the hidden states for each observation in the target domain by using the models learned in the source domain and the basis weights that are given to each transition and emission distribution of the source domains. We now explain each step of the algorithm in detail below.

4.2 SOURCE DOMAIN - TRAINING

The first step is to learn a model for each source domain in the training data. Suppose that we have labeled sequence data for K different source domains. Let

$$Y_t^k = \text{hidden state label at time step } t \text{ for source domain } k$$
$$X_t^k = \text{feature vector at time step } t \text{ for source domain } k$$

Let the sequence of observations be given by $X_{1:T}^k = \{X_1^k, X_2^k, ..., X_T^k\}$ and the hidden states be $\{Y_1^k, Y_2^k, ..., Y_T^k\}$ where $Y_t^k \in \{1, 2, .., N\}\ \forall t$. Furthermore, let us define

$$\Theta_{ij}^k = \Pr(Y_t^k = i | Y_{t-1}^k = j) \text{ i.e. the transition probability from state } i \text{ to state } j$$

We denote the transition matrix for the k^{th} source domain with Θ^k. Let the emission distribution be modeled by a mixture of Gaussian with M components. This implies

$$\Pr(X_t^k | Y_t^k = j) = \sum_{m=1}^{M} w_{j_m}^k \mathcal{N}(X_t^k | \boldsymbol{\mu}_{j_m}^k, \Sigma_{j_m}^k) \quad \forall j \in \{1, 2, .., N\}$$

Our aim is to learn the parameters characterizing the transition and the emission distribution for each source domain. More precisely, if

$$\Phi^k = \{\phi_1^k, \phi_2^k, ..., \phi_N^k\} \text{ where } \phi_i^k = \{(w_{i_1}^k, \boldsymbol{\mu}_{i_1}^k, \Sigma_{i_1}^k), ..., (w_{i_M}^k, \boldsymbol{\mu}_{i_M}^k, \Sigma_{i_M}^k)\}$$

then we want to learn the parameters Θ^k for the transition distribution and Φ^k for the emission distribution for each source domain $k \in \{1, 2, ..., K\}$. Since, we use a hidden Markov model, the update equation at each time step for a source domain k is

$$Pr\Big(\Theta, \Phi, Y_t^k = j | X_t^k, Y_{t-1}^k = i\Big) \propto \overbrace{\Pr(X_t^k | Y_t^k = j)}^{\text{Emission distribution}} \overbrace{\Pr(Y_t^k = j | Y_{t-1}^k = i)}^{\text{Transition Probability}} \overbrace{\Pr(\Theta^k, \Phi^k, Y_{t-1}^k = i | X_{1:t-1}^k)}^{\text{Prior for } t-1}$$
$$\forall j \in \{1, 2, ..., N\} \quad (1)$$

The prior over (Θ^k, Φ^k) is given by

$$\Pr(\Theta^k, \Phi^k) = \Big[\prod_{i=1}^{N} Dir(\theta_i^k | \boldsymbol{\alpha}_i^k)\Big] \Big[\prod_{j=1}^{N} Dir(\mathbf{w}_j^k; \boldsymbol{\beta}_j^k) \prod_{u=1}^{M} \mathcal{NW}(\boldsymbol{\mu}_{j_u}^k, \boldsymbol{\Lambda}_{j_u}^k; \boldsymbol{\delta}_{j_u}^k, \kappa_{j_u}^k, \mathbf{W}_{j_u}^k, v_{j_u}^k)\Big] \quad (2)$$

After substituting the relevant terms in Eq (1), we get

$$\Pr\Big(\Theta, \Phi, Y_t^k = j | X_t^k, Y_{t-1}^k = i\Big) \propto \sum_{m=1}^{M} w_{j_m}^k \mathcal{N}(X_t^k | \boldsymbol{\mu}_{j_m}^k, \Sigma_{j_m}^k) \theta_{ji}^k \Big[\prod_{i=1}^{N} Dir(\theta_i^k | \boldsymbol{\alpha}_i^k)\Big]$$
$$\Big[\prod_{j=1}^{N} Dir(\mathbf{w}_j^k; \boldsymbol{\beta}_j^k) \prod_{u=1}^{M} \mathcal{NW}(\boldsymbol{\mu}_{j_u}^k, \boldsymbol{\Lambda}_{j_u}^k; \boldsymbol{\delta}_{j_u}^k, \kappa_{j_u}^k, \mathbf{W}_{j_u}^k, v_{j_u}^k)\Big] \quad \forall j \in \{1, 2, ..., N\} \quad (3)$$

Further, $\boldsymbol{\Lambda}_{j_u}^k = (\boldsymbol{\Sigma}_{j_u}^k)^{-1}$. The prior in Eq (2) can be understood as having the following components

- **Transition Distribution** : Each column of the $N \times N$ transition matrix specifies the probability of making a transition from that column index to another state given by the row index. We define a Dirichlet distribution as a prior over each column of the transition matrix. Hence, $\prod_{i=1}^{N} Dir(\theta_i^k|\boldsymbol{\alpha}_i^k)$ is the prior over Θ^k.
- **Emission Distribution** : $Dir(\mathbf{w}_j^k;\boldsymbol{\beta}_j^k)\prod_{u=1}^{M}\mathcal{NW}(\boldsymbol{\mu}_{j_u}^k,\boldsymbol{\Lambda}_{j_u}^k;\boldsymbol{\delta}_{j_u}^k,\kappa_{j_u}^k,\mathbf{W}_{j_u}^k,v_{j_u}^k)$ defines a prior over a mixture of Gaussians for hidden state j with M components where the Dirichlet distribution is the prior over the mixture weights and the Normal-Wishart distribution is the prior over the mean and precision matrix of the mixture components. We take a product over j to obtain a prior over all emission distributions.

The posterior distribution (Eq (3)) after each observation is a mixture of products of distributions where each component has the same form as the prior distribution since $\Pr(X_t^k|Y_t^k = j)$ is a mixture of Gaussians. Therefore, the number of terms in the posterior increases exponentially if we perform exact Bayesian learning. To circumvent this, we use BMM for Gaussian Mixture Models as described in (Jaini et al., 2016; Jaini & Poupart, 2016)3. The complete calculations for learning in the source domain are given in appendix B.

The main computation in the learning and updating routine is the calculation of the sufficient set of moments using the Bayesian posterior given in Eq. (9) in appendix B. Let M be the number of components in the mixture model for emission distributions, N the number of hidden states and d the number of features in the data. The computational complexity for updating the parameters in the source domain learning step for each iteration is $\mathcal{O}(M^2N^2)$ for each scalar parameters and is $\mathcal{O}(M^2N^2d^3)$ for the parameters of the distribution over the precision matrix because that involves a matrix multiplication step.

4.3 Target Domain - Prediction

The goal is to predict the hidden states for a target individual (or domain) as we observe a sequence of observations. In the previous step, we learned the transition and emission distributions individually for K different sources. These sources can be thought of as individuals in a population. The transition and emission distributions learned from the individual sources form a basis for the transition and emission distributions of the target domain. Specifically, let the transition distribution for the k^{th} source be denoted by $g(\Theta^k)$ and emission distribution be denoted by $f(\Phi_j^k)$ for a certain hidden state j. Then, the transition and emission distributions for the target domain is a weighted combination given by

$$\Pr(Y_t = j|Y_{t-1} = i) = \sum_{m=1}^{K} \lambda_m \Pr(Y_t^m = j|Y_{t-1}^m = i) = \sum_{m=1}^{K} \lambda_m g(\Theta_{ji}^m) \tag{4}$$

$$\Pr(X_t|Y_t = j) = \sum_{k=1}^{K} \pi_k \Pr(X_t^k|Y_t^k = j) = \sum_{k=1}^{K} \pi_k f(\Phi_j^k) \tag{5}$$

We first need to compute the basis weights $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, ..., \lambda_K)$ and $\boldsymbol{\pi} = (\pi_1, \pi_2, ..., \pi_K)$. We estimate $(\boldsymbol{\lambda}, \boldsymbol{\pi})$ in an unsupervised manner using BMM. We define a Dirichlet prior over $\boldsymbol{\lambda}$ and $\boldsymbol{\pi}$, i.e. $\Pr(\boldsymbol{\lambda}, \boldsymbol{\pi}) = Dir(\boldsymbol{\lambda};\boldsymbol{\gamma})Dir(\boldsymbol{\pi};\boldsymbol{\nu})$. The posterior after observing a new data point is

$$\Pr\Big(\boldsymbol{\lambda},\boldsymbol{\pi}, Y_t = j|X_t\Big) \propto \Pr(X_t|Y_t = j)\sum_{i=1}^{N}\Pr(Y_t = j|Y_{t-1} = i)\Pr(\boldsymbol{\lambda}, \boldsymbol{\pi}, Y_{t-1} = i) \tag{6}$$

$$\propto \sum_{k=1}^{K} \pi_k f(\Phi_j^k) \sum_{i=1}^{N}\sum_{m=1}^{K} \lambda_m g(\Theta_{ji}^m) Dir(\boldsymbol{\lambda};\boldsymbol{\gamma})Dir(\boldsymbol{\pi};\boldsymbol{\nu}) \tag{7}$$

$$\propto \sum_{k,m}^{K}\sum_{i=1}^{N} C(i,j,k,m)\ Dir(\boldsymbol{\pi};\hat{\boldsymbol{\nu}})Dir(\boldsymbol{\lambda};\hat{\boldsymbol{\gamma}}) \tag{8}$$

where $f(\Phi_j^k)g(\Theta_{ji}^m)$ are known from the source domains, $\pi_k Dir(\boldsymbol{\pi};\boldsymbol{\nu}) = a_k Dir(\boldsymbol{\pi};\hat{\boldsymbol{\nu}})$, $\lambda_m Dir(\boldsymbol{\lambda};\boldsymbol{\gamma}) = b_m Dir(\boldsymbol{\lambda};\hat{\boldsymbol{\gamma}})$ and $C(i,j,k,m) = a_k b_m f(\Phi_j^k)g(\Theta_{ji}^m)$. The exact calculations are given in Appendix C. We approximate the posterior in Eq (8) by projecting it onto a tractable family

of distributions with the same set of sufficient moments as the posterior using the Bayesian Moment Matching approach. Finally, the estimate of $(\boldsymbol{\lambda}, \boldsymbol{\pi})$ is the expected value of the final posterior. This completes the learning stage.

The transition and emission distributions for the target domain are the weighted combination of transition and emission distributions learned in the source domain respectively. The advantage of this linear combination is to account for heterogeneity in the data. The learning step in the target domain will ensure that only those source domains that resemble closely the target domain are given higher weights. This helps to bias the predictions according to the closest basis models when the population is not homogeneous.

Predictions can be made in two different manners

- **Online** - initialize the prior over $\boldsymbol{\lambda}$ and $\boldsymbol{\pi}$ to be uniform. As each new data point is observed in a sequence, a prediction is made based on the mean of the current posterior over $\boldsymbol{\lambda}$ and $\boldsymbol{\pi}$ and subsequently the posterior is updated based on Eq (8).
- **Offline** - compute the posterior of $\boldsymbol{\lambda}$ and $\boldsymbol{\pi}$ based on Eq (8) by using the entire sequence of observations of the target individual. Once, the posterior is computed, predict the hidden states for each observation in the sequence based on the mean estimates of the posterior.

In Fig. 1, we show the schematic for the proposed online transfer learning algorithm. The figure shows the learning phase for each source domain where the emission and transition distributions are learned using Bayesian Moment Matching technique. After learning in the source domain, we learn the weights of the basis models in the target domain for each new observation and make predictions in an online manner.

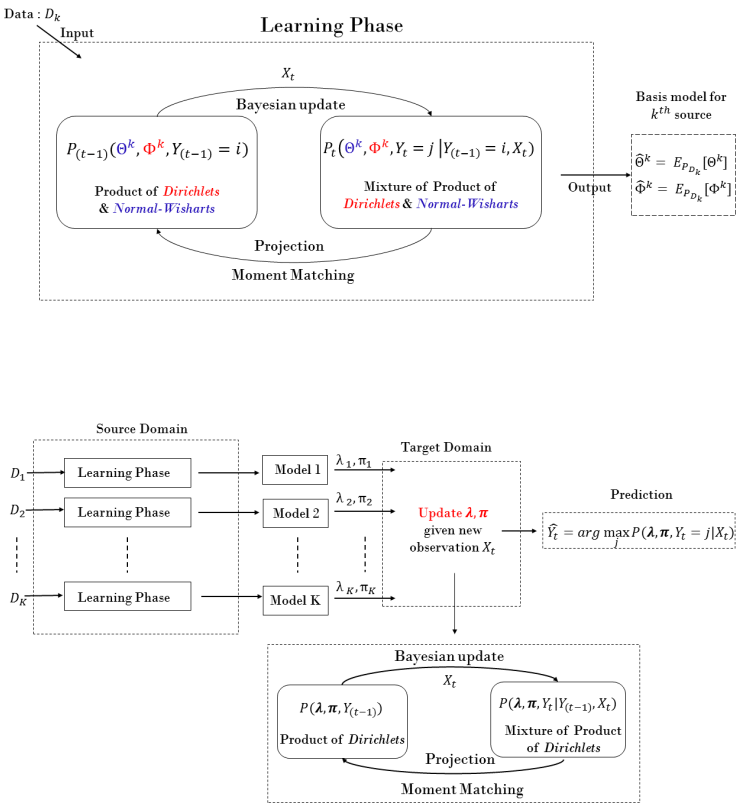


Figure 1: Transfer Learning architecture

Algorithm (1) gives the complete algorithm for transfer learning by Bayesian Moment Matching.

The target domain step involves two routines :

- **Update step** - In this step, the hyper-parameters $(\boldsymbol{\gamma}, \boldsymbol{\nu})$ over the weights $(\boldsymbol{\lambda}, \boldsymbol{\pi})$ are updated. The main computation in this step is the calculation of the set of sufficient moments from the updated Bayesian posterior given in Eq. (8). Hence, the computational complexity of the update step in the target domain for each observation is $\mathcal{O}(K^2N^2)$ where *K* is the number of source domains and *N* is the number of hidden states.

- **Prediction step** - In the prediction step, a hidden label is assigned to the observation based on the model obtained from the update step. The main computation is calculation of the likelihood of each hidden state for the observation. The computational complexity of the prediction step is hence $\mathcal{O}(MKN)$ where *M* is the number of components in the mixture model, *K* is the total number of source domains and *N* is the number of hidden states.

Algorithm 1 Online Transfer Learning by Bayesian Moment Matching

Input (Learning): labeled sequence data from multiple domains (individuals)
Input (Prediction): unlabeled sequence data from individuals
Output: labels for hidden states

Source Domain - learning transition and emission distribution

Input: labeled sequence data from *K* domains
specify # of hidden states : nClass
specify # of components in GMM : nComponents
procedure LEARNSOURCEHMM(data, nClass, nComponents)
  for k = 1 : *K* **do**
    Let $f(\boldsymbol{\Theta}, \boldsymbol{\Phi})$ be a family of probability distributions with parameters $\boldsymbol{\gamma}$
    Initialize a prior $P_0^k(\boldsymbol{\Theta}, \boldsymbol{\Phi})$ from f over transition and emission parameters respectively
    for n = 1 : D_k **do** ▷ D_k : size of data for k^{th} source domain
      Compute $P_n(\boldsymbol{\Theta}, \boldsymbol{\Phi})$ from $P_{n-1}(\boldsymbol{\Theta}, \boldsymbol{\Phi})$ using Eq. 3
      Using BMM approximate P_n with $\tilde{P_n}(\boldsymbol{\Theta}, \boldsymbol{\Phi}) = f(\boldsymbol{\Theta}, \boldsymbol{\Phi}|\boldsymbol{\gamma})$
    Return : $\hat{\boldsymbol{\Theta}} = \mathbb{E}_{\boldsymbol{\Theta}}[\tilde{P_n}(\boldsymbol{\Theta}, \boldsymbol{\Phi})]$
    Return : $\hat{\boldsymbol{\Phi}} = \mathbb{E}_{\boldsymbol{\Phi}}[\tilde{P_n}(\boldsymbol{\Theta}, \boldsymbol{\Phi})]$
Return : emission and transition distributions for each source

Target Domain - learning basis weights for each source domain & prediction

Input: unlabeled sequence data
procedure PREDICTTARGETDOMAIN(data, sourceDistributions)
  Let $g(\boldsymbol{\lambda}, \boldsymbol{\pi}) = Dir(\boldsymbol{\lambda}; \boldsymbol{\gamma}) Dir(\boldsymbol{\pi}; \boldsymbol{\nu})$ be a family of probability distributions
  Initialize a prior $P_0(\boldsymbol{\lambda}, \boldsymbol{\pi})$ from g with equal weights to each source distribution
  for n = 1 : D **do** ▷ D : size of data for target domain
    Compute $P_n(\boldsymbol{\Theta}, \boldsymbol{\Phi})$ from $P_{n-1}(\boldsymbol{\Theta}, \boldsymbol{\Phi})$ using Eq. 8
    Using BMM approximate P_n with $\tilde{P_n}(\boldsymbol{\lambda}, \boldsymbol{\pi}) = g(\boldsymbol{\lambda}, \boldsymbol{\pi})$
    Predict : $\hat{Y_n} = \text{argmax}_j Pr\Big(\boldsymbol{\lambda}, \boldsymbol{\pi}, Y_n = j | X_n\Big)$ using Eq (8)
    Return : $\hat{\boldsymbol{\lambda}} = \mathbb{E}_{\boldsymbol{\lambda}}[\tilde{P_n}(\boldsymbol{\lambda}, \boldsymbol{\pi})]$
    Return : $\hat{\boldsymbol{\pi}} = \mathbb{E}_{\boldsymbol{\pi}}[\tilde{P_n}(\boldsymbol{\lambda}, \boldsymbol{\pi})]$
    Return : prediction $\hat{Y_n}$

5 EXPERIMENTS AND RESULTS

This section describes experiments on three tasks from different domains - activity recognition, sleep cycle prediction among healthy individuals and patients suffering from Parkinson's disease and packet flow prediction in telecommunication networks.

EXPERIMENTAL SETUP

For each task, we compare our online transfer learning algorithm to EM (trained by maximum likelihood) and a baseline algorithm (that uses Bayesian moment matching) that both learn a single HMM with mixtures of Gaussians as emissions by treating the population as homogeneous. Furthermore, we conduct experiments using recurrent neural networks (RNNs) due to their popularity in sequence learning.

The baseline algorithm uses Bayesian Moment Matching to learn the parameters of the HMM. Concretely, we have data collected from several individuals (or sources) in a population for each task. For transfer learning, we train an HMM with mixture of Gaussian emission distributions for each source (or individual) except the target individual. For the target individual, we estimate a posterior over the basis weights in an online and unsupervised fashion and make online predictions about the hidden states. We compare the performance of our transfer learning algorithm against the EM and baseline algorithms that treat the population as homogeneous, i.e., we train an HMM by combining the data from all the sources except the target individual. Then, using this model, we make online predictions about the hidden states of the target individual.

We report the results based on leave-one-out cross validation where the data of a different individual is left out in each round. For each task, we treat every individual as a target individual once. For a fair comparison, the HMM model learned for both the baseline algorithm and the EM algorithm has the same number of components as the HMM model learned by the online transfer learning algorithm.

Regarding RNNs, we used architectures with as many input nodes as the number of attributes, one hidden layer consisting of long short term memory (LSTM) units (Hochreiter & Schmidhuber, 1997) and one softmax output layer with as many nodes as the number of classes. We use the categorical cross-entropy loss as the cost function. We select LSTM units instead of sigmoid or hyperbolic tangent units due to their popularity and success in sequence learning (Sutskever et al., 2014).

We perform grid search to select the best hyper-parameters for each setting. For the training method, we either use Nesterov's accelerated gradient descent (Nesterov, 1983; Sutskever et al., 2013) with learning rates [0.001,0.01,0.1,0.2] and momentum values [0,0.2,0.4,0.6,0.8,0.9], or rmsprop (Tieleman & Hinton, 2012) having $\varepsilon = 10^{-4}$ and decay factor 0.9 (standard values) with learning rates [0.00005,0.0001,0.0002,0.001] and momentum values [0,0.2,0.4,0.6,0.8,0.9]. The weight decay takes values from [0.001,0.01,0.1], whereas the number of LSTM units in the hidden layer takes the possible values [2,4,6,9,12].

We experimented with various architectures before we ended up with the aforementioned values; in particular, architectures with a single hidden layer consistently performed better than multiple layers, possibly because our datasets are not very complex. We train the network by backpropagation through time (bptt) truncated to 20 time steps (Williams & Peng, 1990). The RNNs are trained for a maximum number of 150 epochs, or until convergence is reached. Our implementation is based on the Theano library (Theano Development Team, 2016) in Python.

For each task, we run experiments 10 times with each individual taken as target and the rest acting as source domains for training. We report the average percentage accuracy and use the Wilcoxon signed rank test (Wilcoxon, 1950) to compute a p-value and report statistical significance when the p-value is less than 0.05. In the following sections, we discuss the results for each task in detail.

ACTIVITY RECOGNITION

As part of an on-going study to promote physical activity, we collected smartphone data with 19 participants and tested our transfer learning algorithm to recognize 5 different kinds of activities: sitting, standing, walking, running and in-a-moving-vehicle. While APIs already exist to automatically recognize walking, running and in-a-moving-vehicle by Android and Apple smartphones, sitting and standing are not available in the standard APIs. Furthermore, our long term goal is to obtain robust recognition algorithms for older adults and individuals with perturbed gait (e.g., due to a stroke). Labeled data was obtained by instructing the 19 participants to walk at varying speeds for 4 min, run for 2 min, stand for 2 min, sit for 2 min and ride a moving vehicle to a destination of their choice. The data collected was segmented in epocs of 1 second where 48 features (means and standard deviations of the 3D accelerometry in each epoch) were computed by the smartphone.

The online transfer learning algorithm learned an HMM over 18 individuals which acted as basis models for prediction on the 19^{th} individual. In this manner, we ran experiments for each individual 10 times to get a statistical measure of the results.

Table 1: Average percentage accuracy of prediction for activity recognition on 19 different individuals. The best results among the Baseline, the EM algorithm, RNN and Transfer Learning algorithm are highlighted in bold font. ↑(or ↓) indicates that Transfer Learning has significantly better (or worse) accuracy than the the best algorithm among the baseline, EM and RNN under the Wilcoxon signed rank test with p-value < 0.05.

TARGET DOMAIN	BASELINE	EM	RNN	TRANSFER LEARNING
PERSON 1	**91.29**	83.57	71.15	88.36 ↓
PERSON 2	81.37	79.87	79.58	**87.65**↑
PERSON 3	74.68	75.91	69.56	**93.15**↑
PERSON 4	73.39	68.29	74.25	**84.70**↑
PERSON 5	95.94	89.59	95.36	**99.75**↑
PERSON 6	73.98	69.77	61.71	**96.43**↑
PERSON 7	57.62	55.15	69.22	**70.75**↑
PERSON 8	91.72	86.05	74.49	**97.80**↑
PERSON 9	81.19	78.88	78.72	**88.75**↑
PERSON 10	**99.12**	93.60	92.00	97.35↓
PERSON 11	76.59	74.67	84.75	**88.75**↑
PERSON 12	55.36	59.71	53.63	**95.05**↑
PERSON 13	79.66	73.46	65.54	**97.60**↑
PERSON 14	92.06	89.11	63.59	**93.12**↑
PERSON 15	79.25	72.24	91.08	**94.20**↑
PERSON 16	**84.08**	79.23	74.74	83.51↓
PERSON 17	93.95	91.03	81.25	**97.60**↑
PERSON 18	82.84	74.88	79.45	**87.20**↑
PERSON 19	**95.97**	89.06	95.88	95.06↓

Table (1) compares the average percentage accuracy of prediction for activity recognition with 19 different individuals. It demonstrates that the transfer learning algorithm performed better than the baseline on 15 individuals and in other cases its accuracy was close to the baseline. Furthermore, it is also worth noting that in most cases, the confusion in the algorithm's prediction was between the following pairs of classes: *In a Moving Vehicle—Standing* and *In a Moving Vehicle—Sitting*. This is expected because in most cases the person was either standing/sitting in a bus or sitting in a car. Table (1) also demonstrates the superior performance of online transfer learning algorithm as compared to the EM algorithm. Finally, note the poor performance of RNNs despite the fact that we fine-tuned the architecture to get the best results. RNNs are in theory very expressive. However, they are also notoriously difficult to train and fine-tune due to their non-convexity and vanishing/exploding gradient issues that arise in backpropagation through time. Indeed, in several cases they even underperform all other methods.

SLEEP STAGE CLASSIFICATION

Sleep disruption can lead to various health issues. Understanding and analyzing sleep patterns, therefore, has great potential to significantly improve the quality of life for both patients and healthy individuals. In both clinical and research settings, the standard tool for quantifying sleep architecture and physiology is polysomnography (PSG), which is the measurement of electroencephalography (EEG), electrooculography (EOG), electromyography (EMG), electrocardiography (ECG), and respiratory function of an individual during sleep. The analysis of sleep architecture is of relevance for the diagnosis of several neurological disorders, e.g., Parkinson's disease (Peeraully et al., 2012), because neurological anomalies often also reflect in variations of a patient's sleep patterns.

Typically, PSG data is divided into 30-second *epochs* and classified into 5 stages of sleep — wake (W), rapid eye movement sleep (REM) or one of 3 non-REM sleep stages (N1, N2, and N3) — based on the visual identification of specific signal features on the EEG, EOG, and EMG channels. Epochs that cannot be distinctly sorted into one of the 5 stages are labeled as *Unknown*. While it is a valuable clinical and research tool, visual classification of EEG data remains time consuming,

requiring up to 2 hours for a highly trained technologist to classify all the epochs within a typical 7-hour PSG recording. Beyond that, inter-scorer agreement rates remain low around 80 (Rosenberg & Van Hout, 2013). High annotation costs and low inter-scorer agreement rates have motivated efforts to develop fully automated approaches for sleep stage classification (Anderer et al., 2005; Jensen et al., 2010; Mal, 2013; Punjabi et al., 2015). However, many of these methods result in generic cross-patient classifiers that fail to reach levels of accuracy and reliability high enough to be adopted in real-world medical settings.

The polysomnograms (PSGs) we used for our evaluation were obtained at a clinical neurophysiology laboratory in Toronto (name anonymized) according to the American Academy of Sleep Medicine guidelines using a Grael HD PSG amplifier (Compumedics, Victoria, Australia). We selected recordings from 142 patients obtained between 2009 and 2015. Out of these 142 recordings, 91 were from healthy subjects and 51 were from patients with Parkinson's disease.

Each recording was manually scored by a single registered PSG technologist. Recordings were first segmented into fixed-sized windows of 30 second epochs. To reduce complexity and processing time in the feature extraction and manual labeling step, we only retained EEG channel C4-A1, which is deemed especially important for sleep stage classification (Sil, 2007). Channel selection and segmentation resulted in a ground truth data set where each instance was represented by a single-channel time series of 7680 floating point numbers corresponding to 30 seconds of C4-A1, sampled at 256 Hz. A vector of 26 scalar features was extracted from each epoch. Bao et al. (2011) and Motamedi-Fakhr et al. (2014) give a detailed listing and explanation of all 26 features.

The online transfer learning algorithm learned an HMM over 50 individuals chosen at random which acted as basis models for prediction on the target individual. We did not use all 140 individuals for the basis models because it resulted in sources getting sparse weights diluting the effect of heterogeneity. We completed the experiments for each individual 10 times in this manner to get a statistical measure of the results.

Fig. (2) shows the scatter plots of accuracy for our online transfer learning technique and the three baseline algorithms - BMM, EM (maximum likelihood) and RNNs - which treat the data as homogeneous for the sleep stage classification dataset. For each plot, a point above the dotted line indicates higher accuracy of online transfer learning technique as compared to the corresponding baseline algorithm for the target patient. The plots show consistent superior performance of our online transfer learning technique as compared to both baseline algorithms - BMM and EM for all target patients. The online transfer learning technique also performs better on a majority of patients (102 out of 142) as compared to an optimized RNN.

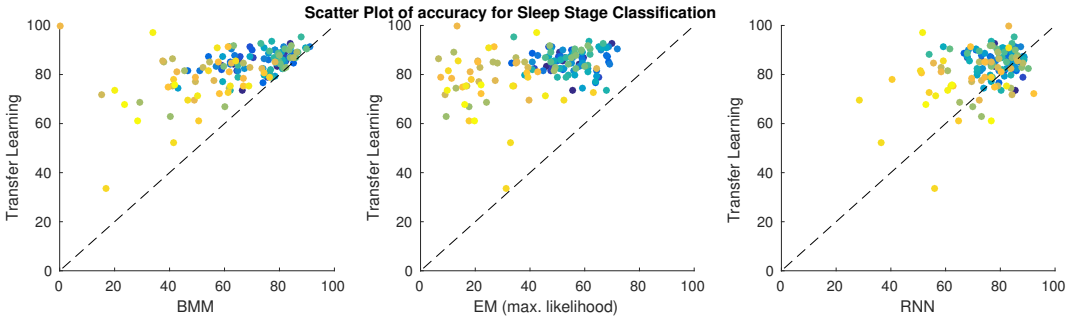


Figure 2: Performance comparison of online transfer learning algorithm with three different baseline algorithms - BMM, EM (max. likelihood) and RNNs on Sleep Stage Classification data using scatter plots of accuracy.

All the results are statistically significant under the Wilcoxon signed rank test with p-value < 0.05. More detailed results for comparison of the online transfer learning technique with the three baseline algorithms is given in appendix (D).

FLOW DIRECTION PREDICTION

Accurate prediction of future traffic plays an important role in proactive network control. Proactive network control means that if we know the future traffic (including directions and traffic volume), then we have more time to find a better policy for the network routing, priority scheduling as well as rate control in order to maximize network throughput while minimizing transmission delay, packet loss rate, etc.

Better understanding the behavior of TCP connections in certain applications can provide important input to automatic application type detection, especially in those scenarios where network traffic is encrypted and DPI (Deep Packet Inspection) is nearly impossible. Different applications can be distinguished by the distinct behavior of their TCP connections, which are well described by the corresponding HMMs.

We performed our experiments with a publicly available dataset of real traffic from academic buildings. The dataset consists of packet traces with TCP flows. For our experiments, we only consider three packet sizes and flow size as the features. The hidden labels are the source of generation of the packet, i.e., *Server* or *Client*. We divided the dataset into 9 domains with each domain consisting of a number of observation sequences. For the online transfer learning algorithm, we learned an HMM for each of 8 sources that acted as basis models for prediction on the 9^{th} source. We compared the performance of the online transfer learning algorithm with EM and the baseline algorithm which treat the data as homogeneous. Table 2 reports the average (of 10 experimental runs) percentage accuracy for each source. The online transfer learning algorithm performs better than both the baseline and the EM algorithm. The results are statistically significant under the Wilcoxon signed rank test with p-value < 0.05. Furthermore, we compare our method to RNNs. It turns out that for the task of traffic direction prediction, RNNs can actually perform well, unlike for instance the activity recognition dataset. The better performance this time may be due to the simpler structure of the data that consists of a single attribute and a binary class. This is in sharp contrast to the activity recognition dataset whose instances contain 48 attributes and can belong to 5 classes, and is thus harder to train.

Table 2: Average percentage accuracy of prediction for flow direction prediction for 9 different domains. The best results among the Baseline, the EM algorithm, RNN and the Transfer Learning algorithm are highlighted in bold font. ↑(or ↓) indicates that transfer learning has significantly better (or worse) accuracy than the best technique among the baseline algorithm, EM and RNN under Wilcoxon signed rank test with pvalue < 0.05.

TARGET DOMAIN	BASELINE	EM	RNN	TRANSFER LEARNING
SOURCE 1	72.00	54.90	**80.00**	71.02 ↓
SOURCE 2	85.33	**89.10**	65.30	86.50↓
SOURCE 3	80.33	81.90	**86.50**	83.33↑
SOURCE 4	86.50	75.80	86.60	**87.17**↑
SOURCE 5	**87.33**	82.80	81.70	86.00↓
SOURCE 6	93.33	78.20	88.90	**93.50**↑
SOURCE 7	95.17	90.70	93.50	**95.33**↑
SOURCE 8	89.83	91.14	91.00	**91.63**↑
SOURCE 9	76.67	75.68	**81.98**	78.83↑

6 CONCLUSION

In many applications, data is produced by a population of individuals that exhibit a certain degree of variability. Traditionally, machine learning techniques ignore this variability and train a single model under the assumption that the population is homogeneous. While several offline transfer learning techniques have already been proposed to account for population heterogeneity, this work describes the first online transfer learning technique (to our knowledge) that incrementally determines which source models best explain a streaming sequence of observations while predicting the corresponding hidden states. We achieved this by adapting the online Bayesian moment matching algorithm originally developed for mixture models to hidden Markov models. Experimental results confirm

the effectiveness of the approach in three real-world applications: activity recognition, sleep stage recognition and flow direction prediction.

In the future, this work could be extended in several directions. Since it is not always clear how many basis models should be used and that the observation sequences of target individuals can necessarily be explained by a weighted combination of basis models, it would be interesting to explore techniques that can automatically determine a good number of basis models and that can generate new basis models on the fly when existing ones are insufficient. Furthermore, since recurrent neural networks (RNNs) have been shown to outperform HMMs with GMM emission distributions in some applications such as speech recognition (Graves et al., 2013), it would be interesting to generalize our online transfer learning technique to RNNs.

REFERENCES

The Visual Scoring of Sleep in Adults. *Journal of Clinical Sleep Medicine*, 3(2):121–131, mar 2007. ISSN 1550-9389.

Performance of an Automated Polysomnography Scoring System Versus Computer-assisted Manual Scoring. *Sleep*, 36(4):573–582, apr 2013. ISSN 1550-9109. doi: 10.5665/sleep.2548.

Samir Al-Stouhi and Chandan K Reddy. Adaptive boosting for transfer learning using dynamic updates. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 60–75. Springer, 2011.

Peter Anderer, Georg Gruber, Silvia Parapatics, Michael Woertz, Tatiana Miazhynskaia, Gerhard Klosch, Bernd Saletu, Josef Zeitlhofer, Manuel J Barbanoj, Heidi Danker-Hopfe, Sari-Leena Himanen, Bob Kemp, Thomas Penzel, Michael Grozinger, Dieter Kunz, Peter Rappelsberger, Alois Schlogl, and Georg Dorffner. An E-health Solution for Automatic Sleep Classification According to Rechtschaffen and Kales: Validation Study of the Somnolyzer 24 x 7 Utilizing the Siesta Database. *Neuropsychobiology*, 51(3):115–133, 2005. ISSN 0302-282X. doi: 10.1159/000085205.

Forrest S Bao, Xin Liu, and Christina Zhang. PyEEG: An Open Source Python Module for EEG/MEG Feature Extraction. *Computational Intelligence and Neuroscience*, 2011:1–7, 2011. ISSN 1687-5265. doi: 10.1155/2011/406391.

Rita Chattopadhyay, Narayanan Chatapuram Krishnan, and Sethuraman Panchanathan. Topology preserving domain adaptation for addressing subject based variability in semg signal. In *AAAI Spring Symposium: Computational Physiology*, pp. 4–9, 2011.

Hai Leong Chieu, Wee Sun Lee, and Leslie P Kaelbling. Activity recognition from physiological data using conditional random fields. 2006.

Diane Cook, Kyle D Feuz, and Narayanan C Krishnan. Transfer learning for activity recognition: A survey. *Knowledge and information systems*, 36(3):537–556, 2013.

Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *Proceedings of the 24th international conference on Machine learning*, pp. 193–200. ACM, 2007.

Morris H. Degroot. *Optimal statistical dcisions*. McGraw-Hill Book Company, New York, St Louis, San Francisco, 1970. ISBN 0-07-016242-5. URL `http://opac.inria.fr/record=b1080767`.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6645–6649. IEEE, 2013.

S Hochreiter and J Schmidhuber. Long short-term memory. *Neural Comp*, 9(8):1735–1780, 1997.

Wei-Shou Hsu and Pascal Poupart. Online bayesian moment matching for topic modeling with unknown number of topics. In *Advances In Neural Information Processing Systems, 2016*, 2016.

Priyank Jaini and Pascal Poupart. Online and distributed learning of gaussian mixture models by bayesian moment matching. *arXiv preprint arXiv:1609.05881*, 2016.

Priyank Jaini, Abdullah Rashwan, Han Zhao, Yue Liu, Ershad Banijamali, Zhitang Chen, and Pascal Poupart. Online algorithms for sum-product networks with continuous variables. In *Proceedings of the Eighth International Conference on Probabilistic Graphical Models*, pp. 228–239, 2016.

Peter S Jensen, Helge B D Sorensen, Helle L Leonthin, and Poul Jennum. Automatic Sleep Scoring in Normals and in Individuals with Neurodegenerative Disorders According to New International Sleep Scoring Criteria. *Journal of Clinical Neurophysiology: Official Publication of the American Electroencephalographic Society*, 27(4):296–302, aug 2010. ISSN 1537-1603. doi: 10.1097/WNP.0b013e3181eaad4b.

Shayan Motamedi-Fakhr, Mohamed Moshrefi-Torbati, Martyn Hill, Catherine M Hill, and Paul R White. Signal Processing Techniques Applied to Human Sleep EEG Signals - A Review. *Biomedical Signal Processing and Control*, 10:21–33, mar 2014. ISSN 17468094. doi: 10.1016/j.bspc.2013.12.003.

Yurii Nesterov. A method of solving a convex programming problem with convergence rate O(1/sqr(k)). *Soviet Mathematics Doklady*, 27:372–376, 1983.

Farheen Omar. Online bayesian learning in probabilistic graphical models using moment matching with applications. 2016.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

Tasneem Peeraully, Ming-Hui Yong, Sudhansu Chokroverty, and Eng-King Tan. Sleep and Parkinson's disease: A review of case-control polysomnography studies. *Movement Disorders*, 27(14): 1729–1737, dec 2012. ISSN 08853185. doi: 10.1002/mds.25197.

Naresh M Punjabi, Naima Shifa, Georg Dorffner, Susheel Patil, Grace Pien, and Rashmi N Aurora. Computer-Assisted Automated Scoring of Polysomnograms Using the Somnolyzer System. *Sleep*, 38(10):1555–1566, 2015. ISSN 1550-9109. doi: 10.5665/sleep.5046.

Parisa Rashidi and Diane J Cook. Transferring learned activities in smart environments. In *Intelligent Environments*, pp. 185–192, 2009.

Abdullah Rashwan, Han Zhao, and Pascal Poupart. Online and distributed bayesian moment matching for sum-product networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 1727–1735, 2016.

Richard S. Rosenberg and Steven Van Hout. The American Academy of Sleep Medicine Inter-scorer Reliability Program: Sleep Stage Scoring. *Journal of Clinical Sleep Medicine*, jan 2013. ISSN 1550-9389. doi: 10.5664/jcsm.2350.

Masa-Aki Sato. Online model selection based on the variational bayes. *Neural Computation*, 13(7): 1649–1681, 2001.

Ling Shao, Fan Zhu, and Xuelong Li. Transfer learning for visual categorization: A survey. *IEEE transactions on neural networks and learning systems*, 26(5):1019–1034, 2015.

I. Sutskever, O. Vinyals, and Q.V. Le. Sequence to sequence learning with neural networks. In *NIPS*, pp. 3104–3112, 2014.

Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of International Conference on Machine Learning (ICML)*, pp. 1139–1147, 2013.

Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, 2016.

T. Tieleman and G. Hinton. Lecture 6.5 - rmsprop, coursera: Neural networks for machine learning. Technical report, 2012.

Chong Wang, John William Paisley, and David M Blei. Online variational inference for the hierarchical dirichlet process. In *AISTATS*, volume 2, pp. 4, 2011.

Frank Wilcoxon. Some rapid approximate statistical procedures. *Annals of the New York Academy of Sciences*, pp. 808–814, 1950.

R.J. Williams and J. Peng. An efficient gradient-based algorithm for online training of recurrent network trajectories. *Neural Computation*, 2(4):490–501, 1990.

Yi Yao and Gianfranco Doretto. Boosting for transfer learning with multiple sources. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 1855–1862. IEEE, 2010.

Zhongtang Zhao, Yiqiang Chen, Junfa Liu, Zhiqi Shen, and Mingjie Liu. Cross-people mobile-phone based activity recognition. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

A NORMAL-WISHART AND DIRICHLET DISTRIBUTION

DIRICHLET DISTRIBUTION

The Dirichlet distribution is a family of multivariate continuous probability distributions over the interval [0,1]. It is the conjugate prior probability distribution for the multinomial distribution. We next show how the combining happens for a Dirichlet as has been highlighted in (3).

$$w_m Dir(\mathbf{w}, \boldsymbol{\alpha}) = \frac{\Gamma(\sum_i \alpha_i)}{\prod_i \Gamma(\alpha_i)} w_m \prod_i w_i^{\alpha_i}$$

$$w_m Dir(\mathbf{w}, \boldsymbol{\alpha}) = \frac{\Gamma(\sum_i \alpha_i)}{\prod_i \Gamma(\alpha_i)} w_m^{\alpha_m+1} \prod_{i \neq m} w_i^{\alpha_i}$$

$$w_m Dir(\mathbf{w}, \boldsymbol{\alpha}) = \frac{\alpha_m}{\sum_i \alpha_i} Dir(\mathbf{w}; \hat{\boldsymbol{\alpha}})$$

where

$$\hat{\alpha}_i = \begin{cases} \alpha_i & \text{if } i \neq m \\ \alpha_i + 1 & \text{if } i = m \end{cases}$$

NORMAL WISHART PRIOR

The Normal-Wishart distribution is a conjugate prior of a multivariate Gaussian distribution with unknown mean and precision matrix (Degroot, 1970). It is the combination of a Wishart distribution over the precision matrix and Gaussian distribution over the mean given the precision matrix.

Let $\boldsymbol{\mu}$ be a d-dimensional vector and $\boldsymbol{\Lambda}$ be a symmetric positive definite $d \times d$ matrix of random variables respectively. Then, a Normal-Wishart distribution over $(\boldsymbol{\mu}, \boldsymbol{\Lambda})$ given parameters $(\boldsymbol{\mu}_0, \kappa, \mathbf{W}, \nu)$ is such that $\boldsymbol{\mu} \sim \mathcal{N}_d\big(\boldsymbol{\mu}; \boldsymbol{\mu}_0, (\kappa\boldsymbol{\Lambda})^{-1}\big)$ where $\kappa > 0$ is real, $\boldsymbol{\mu}_0 \in \mathbb{R}^d$ and $\boldsymbol{\Lambda}$ has a Wishart distribution given as $\boldsymbol{\Lambda} \sim \mathcal{W}(\boldsymbol{\Lambda}; \mathbf{W}, \nu)$ where $\mathbf{W} \in \mathbb{R}^{d \times d}$ is a positive definite matrix and $\nu > d - 1$ is real. The marginal distribution of $\boldsymbol{\mu}$ is a multivariate t-distribution i.e $\boldsymbol{\mu}|\boldsymbol{\Lambda} \sim t_{\nu-d+1}\big(\boldsymbol{\mu}; \boldsymbol{\mu}_0, \frac{\mathbf{W}}{\kappa(\nu-d+1)}\big)$. A Normal-Wishart distribution multiplies with a Gaussian with the same mean and precision matrix to give a new Normal-Wishart distribution.

$$\mathcal{N}_d\big(\mathbf{y}; \boldsymbol{\mu}, (\kappa\boldsymbol{\Lambda})^{-1}\big)\mathcal{NW}(\boldsymbol{\mu}, \boldsymbol{\Lambda}; \boldsymbol{\mu}_0, \kappa, \mathbf{W}, \nu) = c\mathcal{NW}(\boldsymbol{\mu}, \boldsymbol{\Lambda}; \boldsymbol{\mu}_0^*, \kappa^*, \mathbf{W}^*, \nu^*)$$

where

$$\begin{aligned} \boldsymbol{\mu}_0^* &= \frac{\kappa\boldsymbol{\mu}_0 + \mathbf{y}}{\kappa + 1} \\ \kappa^* &= 1 + \kappa \\ \nu^* &= \nu + 1 \\ \mathbf{W}^* &= \mathbf{W} + \frac{\kappa}{\kappa + 1}(\boldsymbol{\mu}_0 - \mathbf{y})(\boldsymbol{\mu}_0 - \mathbf{y})^T \end{aligned}$$

MOMENT MATCHING

In this section we show the system of equations using which the parameters of a product of Dirichlet and Normal-Wishart distribution can be estimated once the set of sufficient moments are known. The set of sufficient moments in this case is $S = \{\boldsymbol{\mu}_j, \boldsymbol{\mu}_j\boldsymbol{\mu}_j^T, \boldsymbol{\Lambda}_j, \Lambda_{j_{km}}^2, w_j, w_j^2\} \mid \forall j \in 1, 2, ..., M\}$ where $\Lambda_{j_{km}}^2$, is the $(k, m)^{th}$ element of the matrix $\boldsymbol{\Lambda}_j$.The expressions for the sufficient moments are :

$$\begin{aligned} \mathbb{E}[w_i] &= \frac{\alpha_i}{\sum_j \alpha_j}; & \mathbb{E}[w_i^2] &= \frac{(\alpha_i)(\alpha_i + 1)}{\Big(\sum_j \alpha_j\Big)\Big(1 + \sum_j \alpha_j\Big)} \\ \mathbb{E}[\boldsymbol{\Lambda}] &= \nu\mathbf{W}; & Var(\boldsymbol{\Lambda}_{ij}) &= \nu(\mathbf{W}_{ij}^2 + \mathbf{W}_{ii}\mathbf{W}_{jj}) \\ \mathbb{E}[\boldsymbol{\mu}] &= \boldsymbol{\delta}; & \mathbb{E}[(\boldsymbol{\mu} - \boldsymbol{\delta})(\boldsymbol{\mu} - \boldsymbol{\delta})^T] &= \frac{\kappa + 1}{\kappa(\nu - d - 1)}\mathbf{W}^{-1} \end{aligned}$$

The parameters of the approximate posterior $\tilde{P}$ can be computed using the equations above in the following manner

$$\alpha_i = \mathbb{E}[w_i]\frac{\mathbb{E}[w_i] - \mathbb{E}[w_i^2]}{\mathbb{E}[w_i^2] - \mathbb{E}[w_i]^2}$$
$$\boldsymbol{\delta} = \mathbb{E}[\boldsymbol{\mu}]$$
$$\mathbf{W}_{ii} = \frac{Var(\boldsymbol{\Lambda}_{ii})}{\mathbb{E}[\boldsymbol{\Lambda}_{ii}]}$$
$$\mathbf{W}_{ij} = \frac{Var(\boldsymbol{\Lambda}_{ij})}{\mathbb{E}[\boldsymbol{\Lambda}_{ij}]}$$
$$\nu = \frac{\mathbb{E}[\boldsymbol{\Lambda}]}{\mathbf{W}}$$
$$\kappa = 1 - \Big((\nu - d - 1)\mathbb{E}[(\boldsymbol{\mu} - \boldsymbol{\delta})(\boldsymbol{\mu} - \boldsymbol{\delta})^T])\mathbf{W}\Big)^{-1}$$

B SOURCE DOMAIN LEARNING USING BMM

The update equation at each time step for a source domain *k* is

$$Pr\Big(\Theta, \Phi, Y_t^k = j | X_t^k, Y_{t-1}^k = i\Big) \propto \overbrace{\Pr(X_t^k | Y_t^k = j)}^{\text{Emission distribution}} \overbrace{\Pr(Y_t^k = j | Y_{t-1}^k = i)}^{\text{Transition Probability}} \overbrace{\Pr(\Theta^k, \Phi^k, Y_{t-1}^k = i | X_{1:t-1}^k)}^{\text{Prior for } t-1}$$
$$\forall j \in \{1, 2, ..., N\}$$

The posterior after inserting all the relevant terms can be written as -

$$Pr\Big(\Theta, \Phi, Y_t^k = j | X_t^k, Y_{t-1}^k = i\Big) \propto \sum_{m=1}^{M} w_{j_m}^k \mathcal{N}(X_t^k | \boldsymbol{\mu}_{j_m}^k, \Sigma_{j_m}^k) \theta_{ji}^k \Big[\prod_{i=1}^{N} Dir(\theta_i^k | \boldsymbol{\alpha}_i^k)\Big]$$
$$\Big[\prod_{j=1}^{N} Dir(\mathbf{w}_j^k; \boldsymbol{\beta}_j^k) \prod_{u=1}^{M} \mathcal{NW}(\boldsymbol{\mu}_{j_u}^k, \boldsymbol{\Lambda}_{j_u}^k; \boldsymbol{\delta}_{j_u}^k, \kappa_{j_u}^k, \mathbf{W}_{j_u}^k, v_{j_u}^k)\Big] \quad \forall j \in \{1, 2, ..., N\}$$

Using (A), we can re-write this as

$$Pr\Big(\Theta, \Phi, Y_t^k = j | X_t^k, Y_{t-1}^k = i\Big) = \frac{1}{Z} \sum_{m=1}^{M} \prod_{u \neq i}^{N} \prod_{u \neq m}^{M} \prod_{i \neq j}^{N} C(i, j, k, m) \Big[Dir(\theta_i^k | \hat{\boldsymbol{\alpha}}_i^k) Dir(\theta_u^k | \boldsymbol{\alpha}_u^k)\Big]$$
$$\Big[Dir(\mathbf{w}_j^k; \hat{\boldsymbol{\beta}}_j^k) Dir(\mathbf{w}_i^k; \boldsymbol{\beta}_i^k)\Big] \Big[\mathcal{NW}(\boldsymbol{\mu}_{j_m}^k, \boldsymbol{\Lambda}_{j_m}^k; \hat{\boldsymbol{\delta}}_m^k, \hat{\kappa}_{j_m}^k, \hat{\mathbf{W}}_{j_m}^k, \hat{v}_{j_m}^k) \mathcal{NW}(\boldsymbol{\mu}_{j_u}^k, \boldsymbol{\Lambda}_{j_u}^k; \boldsymbol{\delta}_{j_u}^k, \kappa_{j_u}^k, \mathbf{W}_{j_u}^k, v_{j_u}^k)\Big] \tag{9}$$

where $Z = \sum_{i,j,k,m} C(i, j, k, m)$ is the normalization constant. Eq (9) is a mixture of product of distributions where each component belongs to the same family as the prior distribution. The set of sufficient moments in this case would be

$$S = \Big\{\theta_i^k, (\theta_i^k)^2, \mathbf{w}_j^k, (\mathbf{w}_j^k)^2, \mu_{j_m}^k, \mu_{j_m}^k (\mu_{j_m}^k)^T, \boldsymbol{\Lambda}_{j_m}^k, \boldsymbol{\Lambda}_{j_m}^k (\boldsymbol{\Lambda}_{j_m}^k)^T | \quad \forall m \in \{1, 2, ..., M\}\Big\}$$

The exact moments can be calculated by

$$\mathbb{E}[z] = \int_{\Theta, \Phi} z Pr\Big(\Theta, \Phi, Y_t^k = j | X_t^k, Y_{t-1}^k = i\Big) d(\Theta) d(\Phi) \quad \forall z \in S$$

Once we know the moments, we can use these moments to estimate the parameters of the approximate distribution using ideas discussed in (3).

C TARGET DOMAIN LEARNING USING BMM

The prior over the weights is

$$\Pr(\boldsymbol{\lambda}, \boldsymbol{\pi}) = Dir(\boldsymbol{\lambda}; \boldsymbol{\gamma}) Dir(\boldsymbol{\pi}; \boldsymbol{\nu})$$

where $\boldsymbol{\gamma}$ and $\boldsymbol{\nu}$ are the hyper-parameters for the Dirichlet distribution. The posterior after each observation is

$$Pr\Big(\boldsymbol{\lambda}, \boldsymbol{\pi}, Y_t = j | X_t\Big) \propto \Pr(X_t | Y_t = j) \sum_{i=1}^{N} \Pr(Y_t = j | Y_{t-1} = i) \Pr(\boldsymbol{\lambda}, \boldsymbol{\pi}, Y_{t-1}) \tag{10}$$

$$\propto \sum_{k=1}^{K} \pi_k \sum_{u=1}^{M} \mathcal{N}(\mu_{j_u}^k, \Sigma_{j_u}^k) \sum_{i=1}^{N} \sum_{m=1}^{K} \lambda_m \theta_{ij}^m Dir(\boldsymbol{\lambda}; \boldsymbol{\gamma}) Dir(\boldsymbol{\pi}; \boldsymbol{\nu}) \tag{11}$$

$$\propto \sum_{k,m}^{K} \sum_{i=1}^{N} \underbrace{\pi_k Dir(\boldsymbol{\pi}; \boldsymbol{\nu})}_{\text{combines}} \underbrace{\lambda_m Dir(\boldsymbol{\lambda}; \boldsymbol{\gamma})}_{\text{combines}} \underbrace{\sum_{u=1}^{M} \mathcal{N}(\mu_{j_u}^k, \Sigma_{j_u}^k) \theta_{ij}^m}_{\text{known}} \tag{12}$$

$$= \frac{1}{Z} \sum_{k,m}^{K} \sum_{i=1}^{N} C(j,k,m)\ Dir(\boldsymbol{\pi}; \hat{\boldsymbol{\nu}}) Dir(\boldsymbol{\lambda}; \hat{\boldsymbol{\gamma}}) \tag{13}$$

where $Z = \sum_{i,j,k,m} C(i,j,k,m)$ is the normalization constant, *K* is the number of source domains and *N* is the number of hidden classes.

Now, we can use the Bayesian Moment Matching algorithm to approximate Eq (8) as a product of two Dirichlets, in the same form as the prior. This posterior will then act as the prior for the next time step. Finally, the values of the weights will be the expected value of each Dirichlet. Let us next see how the combining happens for a Dirichlet.

$$\lambda_m Dir(\boldsymbol{\lambda}, \boldsymbol{\gamma}) = \frac{\gamma_m}{\sum_i \gamma_i} Dir(\boldsymbol{\lambda}; \hat{\boldsymbol{\gamma}}) \tag{14}$$

where

$$\hat{\gamma}_i = \begin{cases} \gamma_i & \text{if } i \neq m \\ \gamma_i + 1 & \text{if } i = m \end{cases}$$

Therefore *C(i,j,k,m)* in Eq (8) is

$$C(i,j,k,m) = \left(\frac{\gamma_m}{\sum_i \gamma_i}\right) \left(\frac{\pi_k}{\sum_i \pi_i}\right) \sum_{u=1}^{M} \mathcal{N}(\mu_{j_u}^k, \Sigma_{j_u}^k) \theta_{ij}^m \tag{15}$$

Next, we outline the moment matching step. The set of sufficient moments is given by

$$S = \{\lambda_i, \lambda_i^2, \pi_i, \pi_i^2 \mid \forall i \in \{1, 2, .., K\}\}$$

$$\mathbb{E}[\lambda_n] = \frac{1}{Z} \sum_{k,m}^{K} \sum_{i=1}^{N} \int \lambda_n C(i,j,k,m) Dir(\boldsymbol{\pi}; \hat{\boldsymbol{\nu}}) Dir(\boldsymbol{\lambda}; \hat{\boldsymbol{\gamma}}) d(\boldsymbol{\lambda}) d(\boldsymbol{\pi}) \tag{16}$$

$$= \frac{1}{Z} \sum_{k,m}^{K} \sum_{i=1}^{N} \int \lambda_n C(i,j,k,m) Dir(\boldsymbol{\lambda}; \hat{\boldsymbol{\gamma}}) d(\boldsymbol{\lambda}) \tag{17}$$

$$= \frac{1}{Z} \sum_{k,m}^{K} \sum_{i=1}^{N} \left(\frac{\hat{\lambda}_n}{\sum_u \hat{\lambda}_u}\right) C(i,j,k,m) \tag{18}$$

Similarly, the second moment can be evaluated as

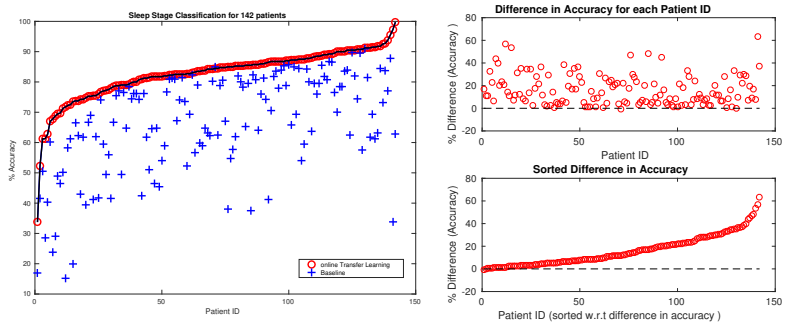
$$\mathbb{E}[\lambda_n^2] = \frac{1}{Z}\sum_{k,m}^{K}\sum_{i=1}^{N}\int \lambda_n^2 C(i,j,k,m) Dir(\boldsymbol{\pi};\hat{\boldsymbol{\nu}}) Dir(\boldsymbol{\lambda};\hat{\boldsymbol{\gamma}}) d(\boldsymbol{\lambda}) d(\boldsymbol{\pi}) \tag{19}$$

$$= \frac{1}{Z}\sum_{k,m}^{K}\sum_{i=1}^{N}\left(\frac{\hat{\lambda}_n(\hat{\lambda}_n+1)}{(\sum_u \hat{\lambda}_u)(1+\sum_u \hat{\lambda}_u)}\right) C(i,j,k,m) \tag{20}$$

We evaluate the moments using the equations above $\forall z \in S$ Once we have the two moments, we can project the posterior into a family of Dirichlet distributions having the same moments. In this way we can perform the learning of the parameters for the target domain.

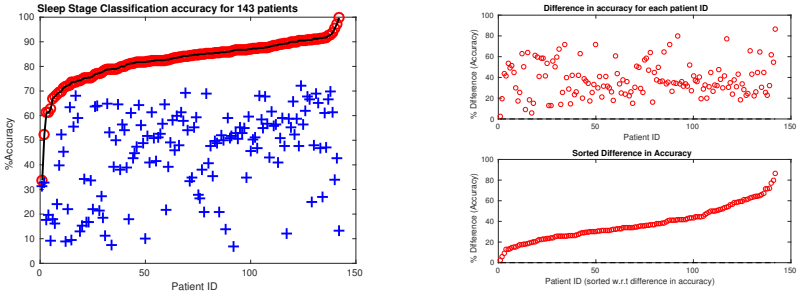
D EXPERIMENT RESULTS : SLEEP STAGE CLASSIFICATION

Fig. 3, 4 and 5 compare the performance of the online transfer learning algorithm with the baseline algorithm, the EM algorithm and recurrent neural networks (RNNs) respectively.



(a) Percentage accuracy (b) Accuracy difference

Figure 3: Performance comparison of online transfer learning algorithm and baseline for the task of sleep stage classification.

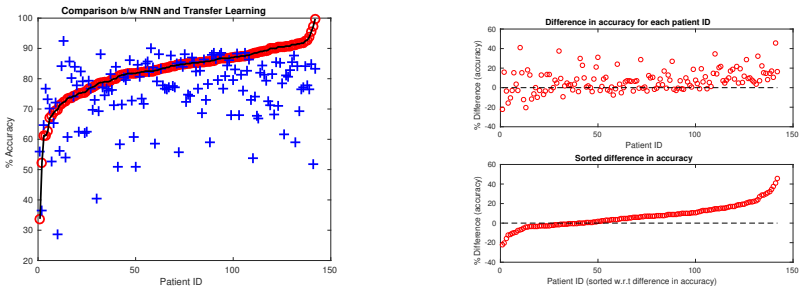


(a) Percentage accuracy (b) Accuracy difference

Figure 4: Performance comparison of online transfer learning algorithm and EM algorithm for the task of sleep stage classification.

Fig. 3a compares the average percentage accuracy for our online transfer learning technique and the baseline algorithm and Fig. 4a compares EM and online transfer learning. The blue $+$ signs represent the accuracy of the baseline algorithm and the red o represent the accuracy of the online transfer learning algorithm. The black line is a reference line that passes through the points plotting the accuracy of the online transfer Learning algorithm. The accuracy is plotted against each individual patient. The blue $+$ signs are always below the black line indicating superior performance of the transfer learning algorithm. Fig. 3b and 4b plot the difference between the accuracy of the baseline algorithm and the transfer learning algorithm. In the top plot, the difference in accuracy is for each

patient corresponding to those shown in Fig. 3a and 4a. In the bottom plot, the difference in accuracy is plotted after sorting. A reference line of 0 is also plotted for the case when there is no difference in performance. The plots suggest that for a majority of patients the transfer learning technique outperforms both the baseline algorithm and EM.



(a) Percentage accuracy

(b) Accuracy difference

Figure 5: Performance comparison of online transfer learning algorithm and Recurrent Neural Networks for the task of sleep stage classification.

In Fig. 5a we compare the performance of the online transfer learning algorithm with RNNs. Fig. 5b plots the difference between the accuracy of RNN and the online transfer learning algorithm. In the top plot, the difference in accuracy is for each patient corresponding to those shown in Fig. 5a. In the bottom plot, the difference in accuracy is plotted after sorting. The figures show that the online transfer learning algorithm outperformed RNNs for a majority of patients (102 out of 142). All the results are statistically significant under the Wilcoxon signed rank test with p-value < 0.05.