Reducing Offline BCI Calibration Effort Using Weighted Adaptation Regularization with Source Domain Selection

Dongrui Wu*, Vernon J. Lawhern†‡, Brent J. Lance†
*Machine Learning Laboratory, GE Global Research, Niskayuna, NY USA
†Translational Neuroscience Branch, U.S. Army Research Laboratory, USA
‡Department of Computer Science, University of Texas at San Antonio, USA
Email: drwu09@gmail.com, vernon.j.lawhern.civ@mail.mil, brent.j.lance.civ@mail.mil

***Abstract*—Single-trial classification of Event-Related Potentials (ERPs) is needed in many real-world brain-computer interface (BCI) applications. However, because of individual differences, the classifier needs to be calibrated by using some labeled subject-specific training samples, which may be inconvenient to obtain. In this paper we propose a weighted adaptation regularization (wAR) approach for offline BCI calibration, which uses data from other subjects to reduce the amount of labeled data required in offline single-trial classification of ERPs. Our proposed model explicitly handles class-imbalance problems which are common in many real-world BCI applications. wAR can improve the classification performance, given the same number of labeled subject-specific training samples; or, equivalently, it can reduce the number of labeled subject-specific training samples, given a desired classification accuracy. To reduce the computational cost of wAR, we also propose a source domain selection (SDS) approach. Our experiments show that wARSDS can achieve comparable performance with wAR but is much less computationally intensive. We expect wARSDS to find broad applications in offline BCI calibration.**

***Index Terms*—Brain-computer interface (BCI), EEG, event-related potentials (ERP), domain adaptation, transfer learning**

I. Introduction

Many real-world brain-computer interface (BCI) applications [11], [13], [21], [22] require single-trial classification of Event-Related Potentials (ERPs) [4], [17]. However, people demonstrate strong individual differences in neural responses to tasks or stimuli, which make it very challenging to develop a generic single-trial ERP classifier whose parameters fit all subjects. Usually, the classifier needs to be calibrated by using some labeled subject-specific training samples. These labeled samples may either be difficult, time-consuming or impractical to obtain. So, there is a critical need to reduce the number of labeled subject-specific training samples required to initially calibrate a BCI system.

Fortunately, although EEG responses from different subjects are generally different, they still share some similarity in the underlying ERP. So, the amount of labeled subject-specific data in calibration could be reduced by making use of information contained in other subjects' data. This is the idea of transfer learning (TL) [14], which has started to find applications in the BCI domain [1], [9], [10], [18].

In [23] we proposed a simple TL approach for single-trial ERP classification which achieved better performance than baseline approaches that did not use TL. Several potential improvements to that approach were also pointed out [23], including using more sophisticated TL algorithms to make use of the unlabeled subject-specific samples in offline calibration, and selecting a subset of auxiliary subjects instead of using all. This paper proposes a new algorithm, weighted adaptation regularization with source domain selection (wARSDS), to implement the above improvements. We show that wARSDS significantly outperforms the TL algorithm in [23], and also the original (unweighted) adaptation regularization algorithm in [12], in offline BCI calibration. An online version of the wARSDS algorithm can be found in [24].

The rest of the paper is organized as follows: Section II introduces the details of the wARSDS algorithm. Section III describes experimental results and performance comparisons of different algorithms. Section IV draws conclusions.

II. Weighted Adaptation Regularization with Source Domain Selection (wARSDS)

This section introduces the wARSDS algorithm, which was modified from the adaptation regularization - regularized least squares (ARRLS) algorithm in [12] to handle class-imbalance problems and multiple source domains, and to also make use of labeled samples in the target domain. wARSDS consists of two parts: source domain selection (SDS) to select the closest source domains, and weighted adaptation regularization (wAR) for each selected source domain. We will introduce wAR first, and then SDS, because SDS relies on the results of wAR.

A. wAR: Problem Definition

Some notations used in wAR are first introduced.

Definition 1: **(Domain)** [12], [14] A domain $\mathcal{D}$ consists of a d-dimensional feature space $\mathcal{X}$ and a marginal probability distribution $P(\mathbf{x})$, i.e., $\mathcal{D} = \{\mathcal{X}, P(\mathbf{x})\}$, where $\mathbf{x} \in \mathcal{X}$.

If two domains $\mathcal{D}_s$ and $\mathcal{D}_t$ are different, then they may have different feature spaces, i.e., $\mathcal{X}_s \neq \mathcal{X}_t$, and/or different marginal probability distributions, i.e., $P_s(\mathbf{x}) \neq P_t(\mathbf{x})$ [12].

978-1-4799-8697-2/15 $31.00 © 2015 IEEE
DOI 10.1109/SMC.2015.557

Definition 2: **(Task)** [12], [14] Given a domain $\mathcal{D}$, a task $\mathcal{T}$ consists of a label space $\mathcal{Y}$ and a prediction function $f(\mathbf{x})$, i.e., $\mathcal{T} = \{\mathcal{Y}, f(\mathbf{x})\}$.

Let $y \in \mathcal{Y}$. Then $f(\mathbf{x}) = Q(y|\mathbf{x})$ can be interpreted as the conditional probability distribution. If two tasks $\mathcal{T}_s$ and $\mathcal{T}_t$ are different, then they may have different label spaces, i.e., $\mathcal{Y}_s \neq \mathcal{Y}_t$, and/or different conditional probability distributions, i.e., $Q_s(y|\mathbf{x}) \neq Q_t(y|\mathbf{x})$ [12].

Definition 3: **(Domain Adaptation**) Given a source domain $\mathcal{D}_s$ with n labeled samples, $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$, and a target domain $\mathcal{D}_t$ with m_l labeled samples $\{(\mathbf{x}_{n+1}, y_{n+1}), ..., (\mathbf{x}_{n+m_l}, y_{n+m_l})\}$ and m_u unlabeled samples $\{\mathbf{x}_{n+m_l+1}, ..., \mathbf{x}_{n+m_l+m_u}\}$, domain adaptation transfer learning aims to learn a target prediction function $f : \mathbf{x}_t \mapsto y_t$ with low expected error on $\mathcal{D}_t$, under the assumptions $\mathcal{X}_s = \mathcal{X}_t$, $\mathcal{Y}_s = \mathcal{Y}_t$, $P_s(\mathbf{x}) \neq P_t(\mathbf{x})$, and $Q_s(y|\mathbf{x}) \neq Q_t(y|\mathbf{x})$.

In our application, EEG epochs from the new subject are in the target domain, while EEG epochs from an existing subject (usually different from the new subject) are in the source domain. There could be more than one source domain, but in wAR we consider each source domain separately. A single data sample would consist of the feature vector for a single EEG epoch from a subject, collected as a response to a specific stimulus. Though the features in source and target domains are computed in the same way, generally their marginal and conditional probability distributions are different, i.e., $P_s(\mathbf{x}) \neq P_t(\mathbf{x})$ and $Q_s(y|\mathbf{x}) \neq Q_t(y|\mathbf{x})$, because the two subjects usually have different neural responses to the same stimulus. As a result, the auxiliary data from a source domain cannot represent the primary data in the target domain accurately, and must be integrated with some labeled data in the target domain to induce the target predictive function.

B. wAR: The Learning Framework

Because

$$f(\mathbf{x}) = Q(y|\mathbf{x}) = \frac{P(\mathbf{x}, y)}{P(\mathbf{x})} = \frac{Q(\mathbf{x}|y)P(y)}{P(\mathbf{x})}, \tag{1}$$

to use the source domain data in the target domain, we need to make sure[1] $P_s(\mathbf{x}_s)$ is close to $P_t(\mathbf{x}_t)$, and $Q_s(\mathbf{x}_s|y_s)$ is also close to $Q_t(\mathbf{x}_t|y_t)$.

Let the classifier be $f = \mathbf{w}^T\phi(\mathbf{x})$, where $\mathbf{w}$ is the classifier parameters, and $\phi : \mathcal{X} \mapsto \mathcal{H}$ is the feature mapping function that projects the original feature vector to a Hilbert space $\mathcal{H}$. The learning framework of wAR is formulated as:

$$\begin{aligned} f = \underset{f \in \mathcal{H}_K}{\operatorname{argmin}} & \sum_{i=1}^{n} w_{s,i}\ell(f(\mathbf{x}_i), y_i) + w_t \sum_{i=n+1}^{n+m_l} w_{t,i}\ell(f(\mathbf{x}_i), y_i) \\ & + \sigma\|f\|_K^2 + \lambda[D_{f,K}(P_s, P_t) + D_{f,K}(Q_s, Q_t)] \end{aligned} \tag{2}$$

where ℓ is the loss function, $K \in R^{(n+m_l+m_u)\times(n+m_l+m_u)}$ is the kernel function induced by ϕ such that $K(\mathbf{x}_i, \mathbf{x}_j) = \langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle$, and σ and λ are non-negative regularization parameters. w_t is the overall weight for target domain samples, which should be larger than 1 so that more emphasis is given to target domain samples than source domain samples. $w_{s,i}$ is the weight for the i^{th} sample in the source domain, and $w_{t,i}$ is the weight for the i^{th} sample in the target domain, i.e.,

$$w_{s,i} = \begin{cases} 1, & \mathbf{x}_i \in \mathcal{D}_{s,1} \\ n_1/(n - n_1), & \mathbf{x}_i \in \mathcal{D}_{s,2} \end{cases} \tag{3}$$

$$w_{t,i} = \begin{cases} 1, & \mathbf{x}_i \in \mathcal{D}_{t,1} \\ m_1/(m_l - m_1), & \mathbf{x}_i \in \mathcal{D}_{t,2} \end{cases} \tag{4}$$

in which $\mathcal{D}_{s,c} = \{\mathbf{x}_i|\mathbf{x}_i \in \mathcal{D}_s \wedge y_i = c\}$ is the set of samples in Class c of the source domain, $\mathcal{D}_{t,c} = \{\mathbf{x}_j|\mathbf{x}_j \in \mathcal{D}_t \wedge y_j = c\}$ is the set of samples in Class c of the target domain, $n_c = |\mathcal{D}_{s,c}|$ and $m_c = |\mathcal{D}_{t,c}|$. The goal of $w_{s,i}$ ($w_{t,i}$) is to balance the number of samples from difference classes in the source (target) domain.

Briefly speaking, the meanings of the four terms in (2) are:

1) The 1st term minimizes the loss on fitting the labeled samples in the source domain.
2) The 2nd term minimizes the loss on fitting the labeled samples in the target domain.
3) The 3rd term minimizes the structural risk of the classifier.
4) The 4th term minimizes the distance between the marginal probability distributions $P_s(\mathbf{x}_s)$ and $P_t(\mathbf{x}_t)$, and the distance between the conditional probability distributions $Q_s(\mathbf{x}_s|y_s)$ and $Q_t(\mathbf{x}_t|y_t)$.

By the Representer Theorem [2], [12], the solution of (2) admits an expression:

$$f(\mathbf{x}) = \sum_{i=1}^{n+m_l+m_u} \alpha_i K(\mathbf{x}_i, \mathbf{x}) = \boldsymbol{\alpha}^T K(X, \mathbf{x}) \tag{5}$$

where $X = [\mathbf{x}_1, ..., \mathbf{x}_{n+m_l+m_u}]^T$, and $\boldsymbol{\alpha} = [\alpha_1, ..., \alpha_{n+m_l+m_u}]^T$ are coefficients to be computed.

Note that the formulation and derivation of wAR closely resemble the ARRLS algorithm proposed by Long et al. [12]; however, there are several major differences:

1) wAR assumes a user is available to label the samples in the target domain, whereas ARRLS assumes all samples in the target domain are unlabeled. As a result, wAR can be iterative, and classification can be updated every time new labeled target domain samples are added.
2) wAR explicitly considers the class imbalance problem in both source and target domains by introducing the weights on samples from different classes.
3) ARRLS also includes manifold regularization [2]. We investigated it but was not able to achieve improved performance in our application, so we excluded it in this paper.

Additionally, with the help of SDS, wARSDS can effectively handle multiple source domains, whereas ARRLS only considers one source domain.

Finally, [12] considered both squared loss and Hinge loss. We only consider the squared loss and 2-class classification in this paper due to space constraints. An analysis with the Hinge loss will be considered in a forthcoming paper.

[1]Strictly speaking, we should also make sure $P_s(y)$ is also close to $P_t(y)$. However, in this paper we assume all subjects conduct similar VEP tasks, so $P_s(y)$ and $P_t(y)$ are intrinsically close. Our future research will consider the general case that $P_s(y)$ and $P_t(y)$ are different.

C. wAR: Loss Functions Minimization

The squared loss for regularized least squares (RLS)

$$\ell(f(\mathbf{x}_i), y_i) = (y_i - f(\mathbf{x}_i))^2 \tag{6}$$

is considered in this paper. Let

$$\mathbf{y} = [y_1, ..., y_{n+m_l+m_u}]^T \tag{7}$$

where $\{y_1, ..., y_n\}$ are known labels in the source domain, $\{y_{n+1}, ..., y_{n+m_l}\}$ are known labels in the target domain, and $\{y_{n+m_l+1}, ..., y_{n+m_l+m_u}\}$ are pseudo labels for the unlabeled target domain samples, i.e., labels estimated using another classifier and known samples in both source and target domains.

Define $E \in R^{(n+m_l+m_u)\times(n+m_l+m_u)}$ as a diagonal matrix with

$$E_{ii} = \begin{cases} w_{s,i}, & i \in [1,\, n] \\ w_t w_{t,i}, & i \in [n+1,\, n+m_l] \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

Substituting (5) and (6) into the first two terms in (2), it follows that

$$\begin{aligned} &\sum_{i=1}^{n} w_{s,i}\ell(f(\mathbf{x}_i), y_i) + w_t \sum_{i=n+1}^{n+m_l} w_{t,i}\ell(f(\mathbf{x}_i), y_i) \\ =&\sum_{i=1}^{n} w_{s,i}(y_i - f(\mathbf{x}_i))^2 + w_t \sum_{i=n+1}^{n+m_l} w_{t,i}(y_i - f(\mathbf{x}_i))^2 \\ =&\sum_{i=1}^{n+m_l+m_u} E_{ii}(y_i - f(\mathbf{x}_i))^2 \\ =&\parallel (\mathbf{y}^T - \boldsymbol{\alpha}^T K)E^{\frac{1}{2}} \parallel^2 \end{aligned} \tag{9}$$

D. wAR: Structural Risk Minimization

As in [12], we define the structural risk as the squared norm of f in $\mathcal{H}_K$, i.e.,

$$\begin{aligned} \|f\|_K^2 =& \sum_{i=1}^{n+m_l+m_u} f(\mathbf{x}_i) \times \sum_{j=1}^{n+m_l+m_u} f(\mathbf{x}_j) \\ =& \sum_{i=1}^{n+m_l+m_u} \sum_{j=1}^{n+m_l+m_u} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \\ =& \boldsymbol{\alpha}^T K \boldsymbol{\alpha} \end{aligned} \tag{10}$$

E. wAR: Marginal Probability Distribution Adaptation

Similar to [12], [15], we compute $D_{f,K}(P_s, P_t)$ using the projected maximum mean discrepancy (MMD):

$$\begin{aligned} D_{f,K}(P_s, P_t) =& \left[\frac{1}{n}\sum_{i=1}^{n} f(\mathbf{x}_i) - \frac{1}{m_l + m_u}\sum_{i=n+1}^{n+m_l+m_u} f(\mathbf{x}_i)\right]^2 \\ =& \boldsymbol{\alpha}^T K M_0 K \boldsymbol{\alpha} \end{aligned} \tag{11}$$

where $M_0 \in R^{(n+m_l+m_u)\times(n+m_l+m_u)}$ is the MMD matrix:

$$(M_0)_{ij} = \begin{cases} \frac{1}{n^2}, & i \in [1,\, n], j \in [1,\, n] \\ \frac{1}{(m_l+m_u)^2}, & i \in [n+1, n+m_l+m_u], \\ & j \in [n+1, n+m_l+m_u] \\ \frac{-1}{n(m_l+m_u)}, & \text{otherwise} \end{cases} \tag{12}$$

F. wAR: Conditional Probability Distribution Adaptation

Similar to the idea proposed in [12], we first need to compute pseudo labels for the unlabeled target domain samples and construct the label vector $\mathbf{y}$ in (7). These pseudo labels can be borrowed directly from the estimates in the previous iteration if wAR is used iteratively, or estimated using another classifier, e.g., a SVM. We then compute the projected MMD w.r.t. each class.

Let $\mathcal{D}_{s,c} = \{\mathbf{x}_i | \mathbf{x}_i \in \mathcal{D}_s \wedge y_i = c\}$ be the set of samples in Class c of the source domain, $\mathcal{D}_{t,c} = \{\mathbf{x}_j | \mathbf{x}_j \in \mathcal{D}_t \wedge y_j = c\}$ be the set of samples in Class c of the target domain, $n_c = |\mathcal{D}_{s,c}|$, and $m_c = |\mathcal{D}_{t,c}|$. Then, the distance between the conditional probability distributions in source and target domains is computed as:

$$\begin{aligned} &D_{f,K}(Q_s, Q_t) \\ =&\sum_{c=1}^{2}\left[\frac{1}{n_c}\sum_{\mathbf{x}_i \in \mathcal{D}_{s,c}} f(\mathbf{x}_i) - \frac{1}{m_c}\sum_{\mathbf{x}_j \in \mathcal{D}_{t,c}} f(\mathbf{x}_j)\right]^2 \end{aligned} \tag{13}$$

Substituting (5) into (13), it follows that

$$\begin{aligned} &D_{f,K}(Q_s, Q_t) \\ =&\sum_{c=1}^{2}\left[\frac{1}{n_c}\sum_{\mathbf{x}_i \in \mathcal{D}_{s,c}} \boldsymbol{\alpha}^T K(X, \mathbf{x}) - \frac{1}{m_c}\sum_{\mathbf{x}_j \in \mathcal{D}_{t,c}} \boldsymbol{\alpha}^T K(X, \mathbf{x})\right]^2 \\ =&\sum_{c=1}^{2} \boldsymbol{\alpha}^T K M_c K \boldsymbol{\alpha} \\ =&\boldsymbol{\alpha}^T K M K \boldsymbol{\alpha} \end{aligned} \tag{14}$$

where

$$M = M_1 + M_2 \tag{15}$$

in which M_1 and M_2 are MMD matrices computed as:

$$(M_c)_{ij} = \begin{cases} 1/n_c^2, & \mathbf{x}_i, \mathbf{x}_j \in \mathcal{D}_{s,c} \\ 1/m_c^2, & \mathbf{x}_i, \mathbf{x}_j \in \mathcal{D}_{t,c} \\ -1/(n_c m_c), & \mathbf{x}_i \in \mathcal{D}_{s,c}, \mathbf{x}_j \in \mathcal{D}_{t,c}, \\ & \text{or } \mathbf{x}_j \in \mathcal{D}_{s,c}, \mathbf{x}_i \in \mathcal{D}_{t,c} \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

G. wAR: The Closed-Form Solution

Substituting (9), (10), (11) and (14) into (2), it follows that

$$\begin{aligned} f =& \underset{f \in \mathcal{H}_K}{\operatorname{argmin}} ||(\mathbf{y}^T - \boldsymbol{\alpha}^T K)E^{\frac{1}{2}}||^2 + \sigma \boldsymbol{\alpha}^T K \boldsymbol{\alpha} \\ &+ \lambda \boldsymbol{\alpha}^T K (M_0 + M) K \boldsymbol{\alpha} \end{aligned} \tag{17}$$

Setting the derivative of the objective function above to 0 leads to

$$\boldsymbol{\alpha} = [(E + \lambda M_0 + \lambda M)K + \sigma I]^{-1} E \mathbf{y} \tag{18}$$

H. Source Domain Selection (SDS)

When there are many source domains, performing wAR for each source domain and then aggregating the results would be very time-consuming; additionally, aggregating results from source domains that are very noisy or very far away from the target domain may also decrease the classification performance. So, there is a need for source domain selection, which selects the closest source domains to reduce the computational cost, and also to (potentially) improve classification performance.

Assume there are Z different source domains. For the z^{th} source domain, we first compute $\mathbf{m}_{z,c}$ $(c = 1, 2)$ the mean vector of each class. Then, we also compute $\mathbf{m}_{t,c}$, the mean vector of each class in the target domain, by making use of the m_l true labels and m_u pseudo-labels. The distance between the two domains is then computed as:

$$d(z,t) = \sum_{c=1}^{2} ||\mathbf{m}_{z,c} - \mathbf{m}_{t,c}|| \tag{19}$$

We next cluster Z, $\{d(z,t)\}_{z=1,...,Z}$, by k-means clustering, and finally choose the cluster that has the smallest centroid, i.e., the source domains that are closest to the target domain. In this way, on average we only need to performing wAR for Z/k source domains, which is corresponding to a 50% computational cost saving if $k = 2$ (the cost for computing $\{d(z,t)\}_{z=1,...,Z}$ and perform k-means clustering is negligible, compared with the cost of performing wAR). A larger k will result in larger savings; however, when k is too large, there may not be enough source domains selected for wAR, and hence the classification performance may decrease. So, there is a trade-off between computational cost saving and classification performance. We used $k = 2$ in this paper.

I. The Complete wARSDS Algorithm

The pseudo code for the complete wARSDS algorithm is described in Algorithm 1. We first use SDS to select the closest source domains, and then perform wAR for each selected source domain separately. The final classification is a weighted average of these individual classifiers, with the weight being the training accuracy of the corresponding wAR.

Algorithm 1: The wARSDS algorithm.

Input: Z source domains, where the z^{th} $(z = 1, ..., Z)$ domain has n_z labeled samples $\{\mathbf{x}_i^z, y_i^z\}_{i=1,...,n_z}$;
m_l labeled target domain samples, $\{\mathbf{x}_j^t, y_j^t\}_{j=1,...,m_l}$;
m_u unlabeled target domain samples, $\{\mathbf{x}_j^t\}_{j=m_l+1,...,m_l+m_u}$;
Parameters σ, λ, and k in k-means clustering.

Output: $\{\bar{y}_j^t\}_{j=m_l+1,...,m_l+m_u}$, estimated labels of the m_u unlabeled target domain samples.

```
// SDS starts
```

if $m_l == 0$ **then**
- Select all Z source domains;
- Go to wAR.

else
- Construct $\{y_j^t\}_{j=m_l+1,...,m_l+m_u}$, pseudo labels for the m_u unlabeled target domain samples, using the estimates from the previous iteration of wAR;
- **for** $z = 1, 2, ..., Z$ **do**
  - Compute $d(z,t)$, the distance between the target domain and the z^{th} source domain, by (19).
- **end**
- Cluster $\{d(z,t)\}_{z=1,...,Z}$ by k-means clustering;
- Retain only the Z' source domains that belong to the cluster with the smallest centroid.

end

```
// SDS ends; wAR starts
```

Choose a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$;

for $z = 1, 2, ..., Z'$ **do**
- Construct feature matrix $\{\mathbf{x}_j\}_{j=1,...,n_z+m_l+m_u}$, where the first n_z rows are the samples from the z^{th} source domain, the next m_l rows are labeled samples from the target domain, and the last m_u rows are unlabeled samples from the target domain;
- Construct the corresponding label vector $\{y_j\}_{j=1,...,n_z+m_l}$;
- Construct $\{y_j\}_{j=n_z+m_l+1,...,n+m_l+m_u}$, pseudo labels for the m_u unlabeled target domain samples, using the estimates from the previous iteration of wARSDS, or build another classifier (e.g., SVM) to estimate the pseudo labels if this is the first iteration;
- Compute kernel matrix K from $\{\mathbf{x}_j\}_{j=1,...,n_z+m_l+m_u}$;
- Construct $\mathbf{y}$ in (7), E in (8), M_0 in (12), and M in (15);
- Compute $\boldsymbol{\alpha}$ by (18);
- Use $\boldsymbol{\alpha}$ to classify the $n_z + m_l$ labeled samples from both domains and record the accuracy, a_z;
- Compute $\{f(\mathbf{x}_j^t)\}_{j=m_l+1,...,m_l+m_u}$ by (5);
- Record $\{y_{z,j}^t\}_{j=m_l+1,...,m_l+m_u}$, where $y_{z,j}^t = sign(f(\mathbf{x}_j^t))$;

end

```
// wAR ends; Aggregation starts
```

Compute $\bar{y}_j^t = sign(\sum_{z=1}^{Z'} a_z y_{z,j}^t)$, $j = m_l + 1, ..., m_l + m_u$;

Return $\{\bar{y}_j^t\}_{j=m_l+1,...,m_l+m_u}$.

III. Experiments and Discussions

Experimental results are presented in this section to demonstrate the performance of wARSDS.

A. Experiment Setup

We used data from a standard Visually Evoked Potential (VEP) oddball task [16], [23]. In this task, image stimuli were presented to subjects at a rate of 0.5 Hz (one image every two seconds). The images presented were either an enemy combatant (target) or a U.S. Soldier (non-target). The subjects were instructed to identify each image as being target or non-target with a unique button press as quickly, but as accurately, as possible. There were a total of 270 images presented to each subject, of which the number of targets ranged from 30

to 55. The experiments were approved by U.S. Army Research Laboratory (ARL) Institutional Review Board [19], [20].

18 subjects participated in the experiments, which lasted on average 15 minutes. Data from four subjects were not used due to data corruption or poor responses. EEG signals were recorded using a 64-channel BioSemi ActiveTwo system with 4 additional EOG channels to record eye movement activity. The EEG data was sampled at 512Hz.

B. Preprocessing and Feature Extraction

We used EEGLAB [6] for EEG signal preprocessing and feature extraction. Of the 64 BioSemi EEG channels, we only used 21 channels (Cz, Fz, P1, P3, P5, P7, P9, PO7, PO3, O1, Oz, POz, Pz, P2, P4, P6, P8, P10, PO8, PO4, O2) mainly in the parietal and occipital areas. We first band-passed the EEG signals to [1, 50] Hz, then downsampled them to 64 Hz, performed average reference, and next epoched them to the $[0, 0.7]$ second interval timelocked to stimulus onset. We removed mean baseline from each channel in each epoch and removed epochs with incorrect button press responses. The final numbers of epochs from the 14 subjects are shown in Table I. Observe that there is significant class imbalance[2] for every subject; that's why we need to use $w_{s,i}$ and $w_{t,i}$ in (2) to balance the two classes in both domains.

Each [0, 0.7] second epoch contains 21×45 raw EEG magnitude samples. To reduce the dimensionality, in each wAR, we performed a simple principal component analysis for the combined concatenated feature vectors from both source and target domains, and took only the scores for the first 20 principal components[3]. We then normalized each feature dimension separately to $[0, 1]$.

C. Evaluation Process and Performance Measure

Although we know the labels of all EEG epochs for all 14 subjects in the experiment, we simulate a different scenario: we have labeled EEG epochs for 13 subjects, but only a small number of epochs for the 14th subject are labeled. Our goal is to iteratively label epochs for the 14th subject so that the remaining unlabeled epochs can be reliably classified. We repeat this procedure 14 times so that each subject has a chance to be the "14th" subject.

Assume there are m_l (m_u) labeled (unlabeled) epochs from the new subject, and they have been arranged in such a way that the first m_l are labeled. Also assume that the true label for the j^{th} epoch from the new subject is y_j^t (1: target; -1: non-target). Using the notations introduced in Algorithm 1, the performance measure is defined as:

$$a = \frac{\sum_{j=1}^{m_l} w_j I_j + \sum_{j=m_l+1}^{m_l+m_u} w_j I_j}{2} \tag{20}$$

[2]In our previous research [23] the non-target samples were downsampled to balance the two classes, and also the performance measure was different. So, the results in this paper should not be compared directly with those in [23]. The problem setting in this paper is more realistic, as class-imbalance is common in many real-world BCI applications.

[3]We tested 20, 30 and 40 principal components, and they showed similar performances.

where I_j is an indicator function on whether the classification is correct or not, i.e.,

$$I_j = \begin{cases} 1, & j \le m_l, \text{ or } j > m_l \text{ and } \bar{y}_j^t = y_j^t \\ 0, & j > m_l \text{ and } \bar{y}_j^t \neq y_j^t \end{cases} \tag{21}$$

and w_j is the weight for the j^{th} epoch to balance the target and non-target epochs, i.e.,

$$w_j = \begin{cases} 1/m_t, & y_j^t = 1 \\ 1/m_{nt}, & y_j^t = -1 \end{cases} \tag{22}$$

in which m_t is the number of target epochs from the new subject and m_{nt} is the number of non-target epochs.

D. Algorithms

We compared the performances of wARSDS with six other algorithms:

1) Baseline 1 (BL1), which assumes we know all labels of the samples from the new subject, and uses SVM with different combinations of parameters ($c = 2^{\{-1,0,\ldots,5\}}$, $\gamma = 2^{\{-4,-3,\ldots,2\}}$) to find the highest 5-fold cross-validation classification accuracy. This usually represents an upper bound of the classification performance we can get, by using the data from the new subject only.
2) Baseline 2 (BL2), which is a simple iterative procedure: in each iteration we randomly select five unlabeled training samples from the new subject, label and add them to the labeled training dataset, and then train an SVM classifier by 5-fold cross-validation. We iterate until the maximum number of iterations is reached.
3) The TL algorithm introduced in [23], which simply combines the labeled samples from the new subject with samples from each existing subject and train an SVM classifier. The final classification is a weighted average of all individual classifiers, with weight being the cross-validation accuracy of the corresponding classifier.
4) TLSDS, which is the above TL algorithm with SDS.
5) ARRLS algorithm proposed in [12] but without manifold regularization, which is also the wAR algorithm developed in the previous section, by setting $w_t = w_{s,i} = w_{t,i} = 1$.
6) wAR, which uses all existing subjects, instead of performing SDS.

Weighted libSVM [5] with RBF kernel was used as the classifier in BL1, BL2, TL and TLSDS. We chose $w_t = 2$ in wAR and wARSDS to give the labeled target domain samples more emphasis, and $\sigma = 0.1$ and $\lambda = 10$, following the practice in [12].

E. Experimental Results

Because random samples were selected for labeling in each iteration, each algorithm was repeated 30 times so that statistically meaningful results could be obtained. The performances of the seven algorithms, which are averaged across the 30 runs for each subject, are shown in Fig. 1, where each subfigure represents a different "14th" subject. The average performance

TABLE I
NUMBER OF EPOCHS FOR EACH SUBJECT AFTER PREPROCESSING. THE NUMBERS OF TARGET EPOCHS ARE GIVEN IN THE PARENTHESES.

Subject	1	2	3	4	5	6	7	8	9	10	11	12	13	14
No. Epochs	241 (26)	260 (24)	257 (24)	261 (29)	259 (29)	264 (30)	261 (29)	252 (22)	261 (26)	259 (29)	267 (32)	259 (24)	261 (25)	269 (33)

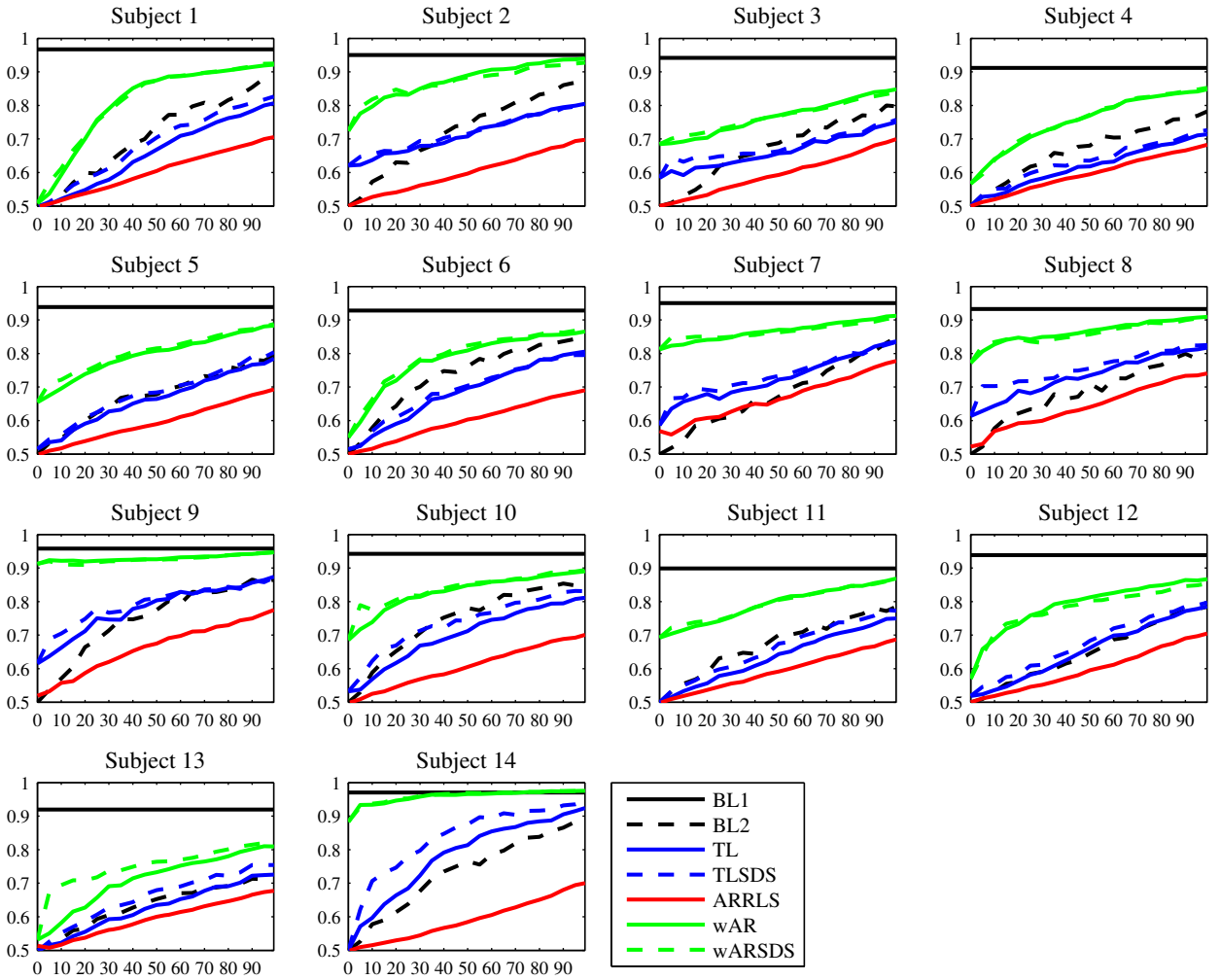

Fig. 1. Performance of the seven algorithms for each individual subject, averaged over 30 runs. Horizontal axis: m_l, number of labeled subject-specific training samples; vertical axis: classification accuracy computed by (20).

of the seven algorithms across the 14 subjects is shown in Fig. 2. Observe that:

1) Generally the performances of all algorithms (except BL1, which is not iterative) increase as more subject-specific training samples are labeled and added, which is intuitive.
2) BL2 cannot build a model when there are no labeled samples at all from the new subject (observe that the first point on the BL2 curve in Fig. 1 is always .5, representing random guess), but all TL based algorithms can, because they can borrow information from other subjects. Moreover, without any labeled samples from the new subject, wAR and wARSDS can build a model with an average classification accuracy of 68%, which is much better than random guess.
3) Generally TL outperforms BL2 when m_l is small, but its performance may be worse when m_l is large. This is because when m_l is small, BL2 cannot be trained extensively to obtain a reliable model, whereas it is beneficial for TL to borrow training samples from other subjects. However, as m_l increases, the performance of BL2 increases rapidly, because BL2 is trained solely on these m_l samples from the new subject. On the other hand, TL is trained by combining these m_l samples with a lot more samples from other subjects, so the impact of m_l on TL is not as large as that on BL2. Because BL2's

performance improves faster than TL as m_l increases, eventually BL2 outperforms TL.

4) TLSDS always outperforms TL. This is because TL uses a very simple way to combine the samples from the new and existing subjects, and hence an existing subject whose ERPs are significantly different from the new subject's would have a negative impact on the final classification performance. SDS removes (some of) such subjects, and hence benefits the performance. Additionally, with the help of SDS, on average TLSDS outperforms BL2 when m_l is small, and has comparable performance as BL2 when m_l is large.
5) ARRLS performs the worst, because all other algorithms explicitly handle class-imbalance using weights, whereas ARRLS does not.
6) wAR and wARSDS significantly outperform BL2, TL, TLSDS and ARRLS. This is because a sophisticated domain adaptation algorithm is used in wAR and wARSDS, which explicitly considers class imbalance, and is optimized not only for high classification accuracy, but also for small structural risk and close similarity of the features. Interestingly, for certain subjects, e.g., Subjects 2, 9 and 14 in Fig. 1, the performances of wAR and wARSDS even approach or exceed BL1, with only 100 random samples (about 35% of the total samples; recall that BL1 was trained using 80% of the total samples). This shows that wAR and wARSDS can indeed transfer useful information, which may not be contained in the samples from the new subject, from other subjects.
7) wARSDS and wAR have very similar performance (on average wARSDS slightly outperforms wAR when m_l is small), but the computational cost of wARSDS is only about 50% of wAR, which is a large saving, especially when the number of existing subjects is very large.

We also performed comprehensive statistical tests to check if the performance differences among the algorithms are statistically significant. To assess overall performance differences among all six algorithms (BL1 was not included because it is not iterative), a measure called the area-under-performance-curve (AUPC) was calculated. The AUPC is the area under the curve of the accuracies obtained at each of the 30 runs, and is normalized to $[0, 1]$. Larger AUPC values indicate better overall classification performance.

First, we used Friedman's test, a two-way non-parametric ANOVA where column effects are tested for significant differences after adjusting for possible row effects. We treated the algorithm type (BL2, TL, TLSDS, ARRLS, wAR, wARSDS) as the column effects, with subjects as the row effects. Each combination of algorithm and subject had 30 values corresponding to 30 runs performed. Friedman's test showed statistically significant differences among the six algorithms ($p = .0000$).

Then, non-parametric multiple comparison tests using Dunn's procedure [7], [8] were used to determine if the difference between any pair of algorithms is statistically significant, with a p-value correction using the FDR method by Benjamini and Hochberg [3]. The results showed that the performances of wAR and wARSDS are statistically significantly different from BL2, TL, TLSDS and ARRLS ($p = .0000$ in all cases). There is no statistically significant performance difference between wAR and wARSDS ($p = .2518$).

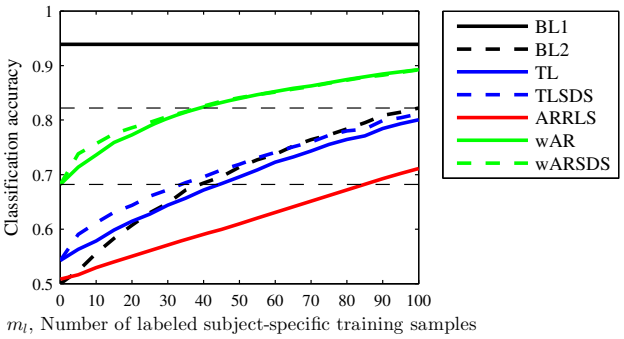

Fig. 2. Average performance of the seven algorithms across the 14 subjects.

In summary, we have demonstrated that given the same number of labeled subject-specific training samples, wAR and wARSDS can significantly improve offline calibration performance. In other words, given a desired classification accuracy, wAR and wARSDS can reduce the number of labeled subject-specific training samples. For example, in Fig. 2, the average classification accuracy of BL2 is 82.22%, given 100 labeled subject-specific training samples. However, to achieve that performance, on average wAR and wARSDS only need 40 samples, which corresponds to 60% saving of labeling effort. Moreover, Fig. 2 also shows that, without using any labeled subject-specific samples, wAR and wARSDS can achieve similar performance to BL2 which uses 35 labeled subject-specific samples.

IV. Conclusions

In this paper we have proposed a wAR approach for offline BCI calibration, which uses data from other subjects to reduce the amount of labeled data required to perform accurate offline single-trial classification of ERPs. It also explicitly considers the class-imbalance problem, which is very common in real-world BCI applications. wAR can indeed improve the classification performance, given the same number of labeled subject-specific training samples; or, equivalently, it can reduce the number of labeled subject-specific training samples, given a desired classification accuracy. Moreover, we also proposed wARSDS, which can achieve comparable performance with wAR but is much less computationally intensive. We expect wARSDS to find broad applications in offline BCI calibration.

Acknowledgement

The authors would like to thank Scott Kerick, Jean Vettel, Anthony Ries, and David W. Hairston at the US Army Research Laboratory (ARL) for designing the experiment and collecting the data.

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-10-2-0022. The views and the conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S Government.

REFERENCES

[1] M. Ahn, H. Cho, and S. C. Jun, "Calibration time reduction through source imaging in brain computer interface (BCI)," *Communications in Computer and Information Science*, vol. 174, pp. 269–273, 2011.

[2] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *Journal of Machine Learning Research*, vol. 7, pp. 2399–2434, 2006.

[3] Y. Benjamini and Y. Hochberg, "Controlling the false discovery rate: A practical and powerful approach to multiple testing," *Journal of the Royal Statistical Society, Series B (Methodological)*, vol. 57, pp. 289–300, 1995.

[4] N. Bigdely-Shamlo, A. Vankov, R. Ramirez, and S. Makeig, "Brain activity-based image classification from rapid serial visual presentation," *IEEE Trans. on Neural Systems and Rehabilitation Engineering*, vol. 16, no. 5, pp. 432–441, 2008.

[5] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

[6] A. Delorme and S. Makeig, "EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis," *Journal of Neuroscience Methods*, vol. 134, pp. 9–21, 2004.

[7] O. Dunn, "Multiple comparisons among means," *Journal of the American Statistical Association*, vol. 56, pp. 62–64, 1961.

[8] O. Dunn, "Multiple comparisons using rank sums," *Technometrics*, vol. 6, pp. 214–252, 1964.

[9] P.-J. Kindermans and B. Schrauwen, "Dynamic stopping in a calibrationless P300 speller," in *Proc. 5th Int'l. Brain-Computer Interface Meeting*, Pacific Grove, CA, June 2013.

[10] P.-J. Kindermans, H. Verschore, D. Verstraeten, and B. Schrauwen, "A P300 bci for the masses: Prior information enables instant unsupervised spelling," in *Proc. Neural Information Processing Systems (NIPS)*, Lake Tahoe, NV, December 2012.

[11] B. J. Lance, S. E. Kerick, A. J. Ries, K. S. Oie, and K. McDowell, "Brain-computer interface technologies in the coming decades," *Proc. of the IEEE*, vol. 100, no. 3, pp. 1585–1599, 2012.

[12] M. Long, J. Wang, G. Ding, S. J. Pan, and P. S. Yu, "Adaptation regularization: A general framework for transfer learning," *IEEE Trans. on Knowledge and Data Engineering*, vol. 26, no. 5, pp. 1076–1089, 2014.

[13] K. McDowell, C.-T. Lin, K. Oie, T.-P. Jung, S. Gordon, K. Whitaker, S.-Y. Li, S.-W. Lu, and W. Hairston, "Real-world neuroimaging technologies," *IEEE Access*, vol. 1, pp. 131–149, 2013.

[14] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[15] B. Quanz and J. Huan, "Large margin transductive transfer learning," in *Proc. 18th ACM Conf. on Information and Knowledge Management (CIKM)*, Hong Kong, November 2009.

[16] A. J. Ries, J. Touryan, J. Vettel, K. McDowell, and W. D. Hairston, "A comparison of electroencephalography signals acquired from conventional and mobile systems," *Journal of Neuroscience and Neuroengineering*, vol. 3, no. 1, pp. 10–20, 2014.

[17] P. Sajda, E. Pohlmeyer, J. Wang, L. Parra, C. Christoforou, J. Dmochowski, B. Hanna, C. Bahlmann, M. Singh, and S.-F. Chang, "In a blink of an eye and a switch of a transistor: Cortically coupled computer vision," *Proc. of the IEEE*, vol. 98, no. 3, pp. 462–478, 2010.

[18] W. Samek, F. Meinecke, and K.-R. Muller, "Transferring subspaces between subjects in brain-computer interfacing," *IEEE Trans. on Biomedical Engineering*, vol. 60, no. 8, pp. 2289–2298, 2013.

[19] US Department of Defense Office of the Secretary of Defense, "Code of federal regulations protection of human subjects," *Government Printing Office*, vol. 32 CFR 19, 1999.

[20] US Department of the Army, "Use of volunteers as subjects of research," *Government Printing Office*, vol. AR 70-25, 1990.

[21] J. van Erp, F. Lotte, and M. Tangermann, "Brain-computer interfaces: Beyond medical applications," *Computer*, vol. 45, no. 4, pp. 26–34, 2012.

[22] J. Wolpaw and E. W. Wolpaw, Eds., *Brain-Computer Interfaces: Principles and Practice*. Oxford, UK: Oxford University Press, 2012.

[23] D. Wu, B. J. Lance, and V. J. Lawhern, "Active transfer learning for reducing calibration data in single-trial classification of visually-evoked potentials," in *Proc. IEEE Int'l. Conf. on Systems, Man, and Cybernetics*, San Diego, CA, October 2014.

[24] D. Wu, V. J. Lawhern, and B. J. Lance, "Reducing BCI calibration effort in RSVP tasks using online weighted adaptation regularization with source domain selection," in *Proc. Int'l. Conf. on Affective Computing and Intelligent Interaction*, Xi'an, China, September 2015.