Available online at www.sciencedirect.com

ScienceDirect

Pattern Recognition Letters 29 (2008) 1285–1294

Pattern Recognition Letters

www.elsevier.com/locate/patrec

A self-training semi-supervised SVM algorithm and its application in an EEG-based brain computer interface speller system

Yuanqing Li [*], Cuntai Guan, Huiqi Li, Zhengyang Chin

Institute for Inforcomm Research, Heng Mui Keng Terrace, 21, Singapore 119613, Singapore

Received 3 October 2006; received in revised form 16 January 2008
Available online 15 February 2008

Communicated by M.-J. Li

Abstract

In this paper, we first present a self-training semi-supervised support vector machine (SVM) algorithm and its corresponding model selection method, which are designed to train a classifier with small training data. Next, we prove the convergence of this algorithm. Two examples are presented to demonstrate the validity of our algorithm with model selection. Finally, we apply our algorithm to a data set collected from a P300-based brain computer interface (BCI) speller. This algorithm is shown to be able to significantly reduce training effort of the P300-based BCI speller.
© 2008 Elsevier B.V. All rights reserved.

IDT: 10; 90; 170

Keywords: Semi-supervised support vector machine (SVM); Model selection; Convergence; Brain computer interface (BCI); Electroencephalogram (EEG)

1. Introduction

In many classification applications, collecting labeled instances for training is either difficult, expensive or time-consuming. Meanwhile unlabeled data may be relatively easier to obtain. If only a small amount of labeled data and a large amount of unlabeled data are available, semi-supervised learning can often provide us with a satisfactory classifier. In recent years, semi-supervised learning has received considerable attention due to its potential for reducing the effort of labeling data. Existing semi-supervised algorithms include expectation maximization (EM) algorithm, self-training algorithms, co-training algorithms (Nigam and Ghani, 2000; Blum and Mitchell, 1998), entropy minimization (Grandvalet and Bengio, 2004), graph-based methods (Zhou et al., 2004), etc. It is well known that support vector machine is a strong classifier. A class of common semi-supervised learning algorithms based on SVM are transductive support vector machines (TSVMs) (Vapnik, 1998; Joachims, 1999, etc.), in which the labels of test data set leading to the lowest structural risk are chosen. Since the optimization problem of a typical TSVM is non-convex and finding its exact solution is NP-hard, several approximation algorithms have been established (Bennett and Demiriz, 1998; Demiriz and Bennett, 2000; Chapelle and Zien, 2005; Joachims, 2003). However, when the size of test data set is big (e.g. larger than 1000), TSVM type algorithms are still time-consuming (see Example 2 in Section 3). In several other studies (Park and Zhang, 2004; Brefeld and Scheffer, 2004; Kockelkorn et al., 2003; Kiritchenko and Matwin, 2002), a multi-view co-training support vector machine and its variants were presented. For text classification, experiments have clearly shown that the co-training SVM outperforms the

Abbreviations: Semi-supervised SVM, semi-supervised support vector machine; BCI, brain computer interface.

[*] Corresponding author. Tel.: +65 68746761; fax: +65 65679524.
E-mail address: yqli2@i2r.a-star.edu.sg (Y. Li).

co-training naive Bayes (Kiritchenko and Matwin, 2002). Compared with TSVM algorithms, the computational burden of the co-training support vector machine is much lower.

In semi-supervised learning, the self-training EM algorithm with a naive Bayes classifier is commonly used. This algorithm is simple, effective in many cases and converges (Xu and Jordan, 1996). If the naive Bayes classifier in the self-training EM algorithm is replaced by a standard SVM, then a self-training SVM algorithm is obtained. In this paper, we analyze this self-training SVM algorithm, which is similar to the case of the co-training SVM in (Park and Zhang, 2004; Brefeld and Scheffer, 2004) when only one view of data is available. The co-training SVM is an incremental algorithm. However, the self-training SVM in this paper is iterative. Furthermore, it will be proved that the objective function of this self-training SVM (named as structural risk) monotonically decreases during its iterations. Therefore, the convergence and effectiveness of this algorithm are guaranteed. Our data analysis examples show its fast convergence, light computational burden as well as effectiveness.

In SVM, there is a regularization parameter C to be set in advance. This procedure is called model selection. Model selection is very important for SVM type algorithms since this parameter can affect the performance of SVM. Generally, model selection is carried out using cross-validation on training data set. However, if the training data set is small, the result obtained by cross-validation is not reliable. In this paper, we present a semi-supervised learning-based model selection method, in which the test data set are used. Our model selection method is suitable for small training data set.

As an application example, we will illustrate how to use this algorithm in a P300-based brain computer interface (BCI) speller. BCIs provide an alternative means of communication and control for people with severe motor disabilities (Birbaumer et al., 1999), thus research into BCIs has received more attention in recent years, as seen in (Donoghue, 2002; Kubler et al., 2001). Being non-invasive, electroencephalogram (EEG)-based BCI measures specific components of EEG activity, extracts features and translates these features into control signals to devices such as a robot arm or a cursor. For many electroencephalogram (EEG)-based brain computer interfaces (BCIs), a tedious and time-consuming training process is needed to set parameters. In BCI Competition 2005 dataset IVa (BCI2005, 2005), reducing the training process has been explicitly proposed as a task for competitors. Furthermore, an effective BCI system needs to be adaptive to dynamic variations of brain signals, i.e. its parameters need to be adjusted online.

We will use our self-training SVM algorithm to analyze a data set collected from a P300-based BCI speller system and simulate an online scenario. We will illustrate that this algorithm can effectively reduce the training time of the P300-based BCI speller. We also point out that this algorithm can be used to improve the adaptability of the BCI system.

2. Self-training semi-supervised SVM algorithm

In this section, we first present the steps of a self-training semi-supervised SVM algorithm and a model selection method, then analyze the convergence of our algorithm.

2.1. Algorithm

A standard SVM classifier for two-class problem can be defined as

$$\begin{aligned} &\min \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\ &\text{s.t.} \quad y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b) \geqslant 1-\xi_i, \quad \xi_i \geqslant 0, \ i=1,\ldots,N, \end{aligned} \tag{1}$$

where $\mathbf{x}_i \in R^n$ is a feature vector of a training sample, $y_i \in \{-1,1\}$ is the label of $\mathbf{x}_i$, $i=1,\ldots,N$, $C>0$ is a regularization constant.

The steps of a self-training semi-supervised SVM algorithm are as follows:

Algorithm 1. Suppose that we have a small training set F_I containing N_1 samples $\{\mathbf{x}_i, i=1,\ldots,N_1\}$ with given labels $[y_0(1),\ldots,y_0(N_1)]$, and a test set F_T containing N_2 samples $\{\mathbf{x}_{N_1+i},\ i=1,\ldots,N_2\}$ with unknown labels.

Step 1. Using F_I, we train a SVM, and perform classification on F_T. The parameters of the SVM are denoted as $\mathbf{w}^{(1)} \in R^n$, $\xi^{(1)} \in R^{N_1}$ and $b^{(1)} \in R$. The predicted labels are denoted as $[y^{(1)}(1),\ldots,y^{(1)}(N_2)]$. The superscript denotes the current iteration number.

Step 2. The kth iteration ($k=2,\ldots$) follows Steps 2.1–2.3.

Step 2.1. Define a new training set as $F_N = F_I + F_T$, where the labels of F_T are predicted in the $(k-1)$th iteration.

Step 2.2. Using the augmented training set F_N, we train a SVM, and perform classification again on F_T. The parameters of the SVM are denoted as $\mathbf{w}^{(k)} \in R^n$, $\xi^{(k)} \in R^{N_1+N_2}$ and $b^{(k)} \in R$. The predicted labels are denoted as $[y^{(k)}(1),\ldots,y^{(k)}(N_2)]$.

Step 2.3. Calculate the objective function value in (1),

$$f(\mathbf{w}^{(k)},\xi^{(k)}) = \frac{1}{2}\|\mathbf{w}^{(k)}\|^2 + C\sum_{i=1}^{N_1+N_2}\xi_i^{(k)}. \tag{2}$$

Step 3. (Termination step): Given a pre-determined positive constant δ_0: if $|f(\mathbf{w}^{(k)},\xi^{(k)}) - f(\mathbf{w}^{(k-1)},\xi^{(k-1)})| < \delta_0$, the algorithm stops after the kth iteration, and the predicted labels $[y^{(k)}(1),\ldots,y^{(k)}(N_2)]$ of the test set are the final classification results. Otherwise, perform the $(k+1)$th iteration.

2.2. Model selection

In Algorithm 1, we need to set the regularization parameter C of SVM through model selection. In our case, we do not use cross-validation on training data set to search the valued of C because Algorithm 1 is designed to work with small training data set. Cross-validation with small training data set may not be reliable.

Now we present a model selection method to determine C with the help of test data and Fisher ratio. In this method, we search the value of C on a finite set $\{C_1, \ldots, C_L\}$. In the optimization of a SVM, the objective function contains two items, one is the square of the 2-norm of the weight vector ($\|\mathbf{w}\|^2$), which reflects the margin, the other is the residual error. The two items are regularized by C. If we emphasize the margin, we can set C small; otherwise, if we emphasize the residual error, we can set C big. As our experience, we suggest to choose C in the interval $(0, 1]$. More details on how to choose the set $\{C_1, \ldots, C_L\}$ can be seen in references on SVM e.g. (Chang and Lin, 2001).

Given $C \in \{C_1, \ldots, C_L\}$, we run K_0 iterations of Algorithm 1, where K_0 is a preset positive integer; we set $K_0 = 10$ because, as will be seen, Algorithm 1 converges in about 10 iterations generally. After the kth iteration ($k = 1, \ldots, K_0$), we can obtain a SVM score for each data sample in the initial training data set and the test data set. Using the given labels of the training set F_I and the predicted labels $[y^{(k)}(1), \ldots, y^{(k)}(N_2)]$ of the test set F_T we separate the set of scores into two subsets $S_1^{(k)}$ (with labels "+1") and $S_2^{(k)}$ (with labels "−1").

We now calculate the Fisher ratio,

$$R(C,k) = \frac{(m_1^{(k)} - m_2^{(k)})^2}{\sqrt{\sigma_1^{(k)}\sigma_2^{(k)}}}, \tag{3}$$

where $k = 1, \ldots, K_0$. $m_1^{(k)}$ and $\sigma_1^{(k)}$ are the mean value and the variance of $S_1^{(k)}$, respectively. $m_2^{(k)}$ and $\sigma_2^{(k)}$ are the mean value and the variance of $S_2^{(k)}$, respectively.

We consider that $R(C, 1)$ is calculated from the initial small training data set and may not be reliable. Thus we find the maximum of the last $K_0 - 1$ Fisher ratios,

$$R_m(C) = \max\{R(C,2), \ldots, R(C,K_0)\}, \tag{4}$$

where $C \in \{C_1, \ldots, C_L\}$.

Suppose that $R_m(C_0)$ is the maximum of $\{R_m(C) | C \in \{C_1, \ldots, C_L\}\}$. Then C_0 is the selected parameter value for C.

The above method for selecting C is based on the fact that the Fisher ratio generally represents the separability of a corresponding data set, i.e. a large Fisher ratio generally implies high separability of the data set. We choose the C value which leads to the highest Fisher ratio in training data set and test data set (with predicted labels). This is similar to transductive SVM, where the labels of test data set leading to the lowest structural risk are chosen. In Example 1 and Section 3, we will demonstrate the validity of this model selection method.

Generally, for a semi-supervised learning algorithm with parameters, this model selection method may be used to select parameters when the training data set is small. For instance, in the self-training EM algorithm with a naive Bayes classifier, we can directly use this model selection method for parameter setting except that the classifier is naive Bayes classifier instead of the SVM classifier. Another example, using this method we can choose those values of the parameters (including C) for transductive SVM algorithms which lead to the highest Fisher ratio in training data set and test data set (with predicted labels). In this case, the classifier is a transductive SVM instead of the self-training SVM in this paper. But then, the computational burden might be very heavy for large data set as mentioned before.

2.3. Convergence analysis

We now analyze the convergence of Algorithm 1 and have the following theorem.

Theorem 1. *For $f(\mathbf{w}^{(k)}, \xi^{(k)})$ defined in (2), we have,*

$$f(\mathbf{w}^{(k-1)}, \xi^{(k-1)}) \geqslant f(\mathbf{w}^{(k)}, \xi^{(k)}). \tag{5}$$

Proof. According to Step 1 of Algorithm 1, $(\mathbf{w}^{(1)}, \xi^{(1)}, b^{(1)})$ is the solution of following optimization problem with training data set F_I,

$$\begin{aligned} &\min \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N_1}\xi_i \\ &\text{s.t.} \quad y_0(i)(\mathbf{w}^{\mathrm{T}}x_i + b) \geqslant 1 - \xi_i, \quad \xi_i \geqslant 0,\ i = 1, \ldots, N_1, \end{aligned} \tag{6}$$

where $y_0(i)$ are the true initial training data labels, which remain constant in the iterations of Algorithm 1.

Note that the predicted labels $[y^{(1)}(1), \ldots, y^{(1)}(N_2)]$ are defined by the following inequalities:

$$y^{(1)}(i)((\mathbf{w}^{(1)})^{\mathrm{T}}\mathbf{x}_{N_1+i} + b^{(1)}) \geqslant 0, \quad i = 1, \ldots, N_2. \tag{7}$$

We now expand the vector $\xi^{(1)} \in R^{N_1+N_2}$ by defining

$$\xi_{N_1+i}^{(1)} = \begin{cases} 0, & \text{if } y^{(1)}(i)((\mathbf{w}^{(1)})^{\mathrm{T}}\mathbf{x}_{N_1+i} + b^{(1)}) \geqslant 1, \\ 1 - y^{(1)}(i)((\mathbf{w}^{(1)})^{\mathrm{T}}\mathbf{x}_{N_1+i} + b^{(1)}), & \text{otherwise}, \end{cases} \tag{8}$$

where $i = 1, \ldots, N_2$.

Define a label vector $\mathbf{y}^{(1)} = [y_1^{(1)}, \ldots, y_{N_1+N_2}^{(1)}] = [y_0(1), \ldots, y_0(N_1), y^{(1)}(1), \ldots, y^{(1)}(N_2)]$. For the second iteration of Algorithm 1, we can find that $(\mathbf{w}^{(2)}, \xi^{(2)}, b^{(2)})$ is the solution of the following optimization problem with training data set $F_I + F_T$:

$$\begin{aligned} \min \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N_1+N_2}\xi_i \\ \text{s.t.} \quad & y_i^{(1)}(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b) \geqslant 1-\xi_i, \xi_i \geqslant 0, \\ & i = 1,\ldots,N_1+N_2. \end{aligned} \tag{9}$$

From (7) and (8), $(\mathbf{w}^{(1)},\xi^{(1)},b^{(1)})$ is a feasible solution of (9). Since $(\mathbf{w}^{(2)},\xi^{(2)},b^{(2)})$ is an optimal solution of (9), thus we have

$$f(\mathbf{w}^{(1)},\xi^{(1)}) \geqslant f(\mathbf{w}^{(2)},\xi^{(2)}). \tag{10}$$

Similarly, $(\mathbf{w}^{(k-1)},\xi^{(k-1)},b^{(k-1)})$ $(k>2)$ is the solution of following optimization problem,

$$\begin{aligned} \min \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N_1+N_2}\xi_i \\ \text{s.t.} \quad & y_i^{(k-2)}(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b) \geqslant 1-\xi_i, \xi_i \geqslant 0, \\ & i = 1,\ldots,N_1+N_2, \end{aligned} \tag{11}$$

where label vector $\mathbf{y}^{(k-2)} = [y_0(1),\ldots,y_0(N_1),y^{(k-2)}(1),\ldots,y^{(k-2)}(N_2)]$.

$(\mathbf{w}^{(k)},\xi^{(k)},b^{(k)})$ is the solution of following optimization problem:

$$\begin{aligned} \min \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N_1+N_2}\xi_i \\ \text{s.t.} \quad & y_i^{(k-1)}(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b) \geqslant 1-\xi_i, \xi_i \geqslant 0, \\ & i = 1,\ldots,N_1+N_2, \end{aligned} \tag{12}$$

where label vector $\mathbf{y}^{(k-1)} = [y_0(1),\ldots,y_0(N_1),y^{(k-1)}(1),\ldots,y^{(k-1)}(N_2)]$.

We now prove that $(\mathbf{w}^{(k-1)},\xi^{(k-1)},b^{(k-1)})$ is a feasible solution of (12). There are two cases:

1. If $y_i^{(k-2)} = y_i^{(k-1)}$ (e.g. this equality always holds for $i = 1,\ldots,N_1$), then according to the constraint of (11),

$$\begin{aligned} y_i^{(k-1)}((\mathbf{w}^{(k-1)})^{\mathrm{T}}\mathbf{x}_i + b^{(k-1)}) &= y_i^{(k-2)}((\mathbf{w}^{(k-1)})^{\mathrm{T}}\mathbf{x}_i + b^{(k-1)}) \\ &\geqslant 1-\xi_i^{(k-1)}, \quad \xi_i^{(k-1)} \geqslant 0. \end{aligned} \tag{13}$$

2. If $y_i^{(k-2)} \neq y_i^{(k-1)}$ (e.g., this inequality may hold for $i > N_1$), then

$$y_i^{(k-2)}((\mathbf{w}^{(k-1)})^{\mathrm{T}}\mathbf{x}_i + b^{(k-1)}) < 0. \tag{14}$$

This is because from the definition of $y_i^{(k-1)}$, $y_i^{(k-1)}((\mathbf{w}^{(k-1)})^{\mathrm{T}}x_i + b^{(k-1)}) \geqslant 0$.

Thus we have

$$\begin{aligned} y_i^{(k-1)}((\mathbf{w}^{(k-1)})^{\mathrm{T}}\mathbf{x}_i + b^{(k-1)}) &> y_i^{(k-2)}((\mathbf{w}^{(k-1)})^{\mathrm{T}}\mathbf{x}_i + b^{(k-1)}) \\ &\geqslant 1-\xi_i^{(k-1)}. \end{aligned} \tag{15}$$

Until now, we have proved that $(\mathbf{w}^{(k-1)},\xi^{(k-1)},b^{(k-1)})$ satisfies the constraints of (12). That is to say, it is a feasible solution of (12). Noting that $(\mathbf{w}^{(k)},\xi^{(k)},b^{(k)})$ is the optimal solution of (12), hence we have the inequality in (5). The theorem is proven. □

Remarks 1. (i) Since $f(\mathbf{w}^{(k)},\xi^{(k)}) \geqslant 0$, it follows from Theorem 1 that $\{f(\mathbf{w}^{(k)},\xi^{(k)})\}$ is convergent. Thus Algorithm 1 is convergent. This will be demonstrated by our data analysis examples. (ii) The objective function in (1) can be explained as a structural risk (Mika, 2002). According to Theorem 1, the structural risk of the sum of the training set and the test set decreases with every iteration of Algorithm 1. This generally leads to increased classification accuracy rates. (iii) According to our simulations and real data analysis, Algorithm 1 converges very fast (generally in 10 iterations). (iv) Finally, although a minimum of $f(\mathbf{w}^{(k)},\xi^{(k)})$ can be found through the iterations of Algorithm 1, this minimum is only guaranteed to be local.

In (Xu and Jordan, 1996), a standard EM algorithm with a naive Bayesian classifier was analyzed, which is also iterative and self-training. Furthermore, the EM algorithm is convergent since the objective function (expectation of a likelihood) of this algorithm monotonically increases during its iterations. From the above analysis, we can find that our algorithm has a similar working mechanism to the EM algorithm although the objective functions and classifiers of these two algorithms are different. Generally, if the distribution of the data is Gaussian (or close to Gaussian) and the dimension of the data is not very high, the EM algorithm may be used for classification; otherwise, the self-training semi-supervised SVM algorithm in this paper might obtain better results.

3. Simulation results

In this section, we present two data analysis examples. In the first example, we mainly demonstrate the validity of our model selection method using toy data. In the second example, analysis results for several real-world data sets are presented to demonstrate the effectiveness and fast convergence of our algorithm. Additionally, all SVM classifications in the iterations of Algorithm 1 are performed by LIBSVM (Chang and Lin, 2001).

Example 1. We demonstrate the validity and convergence of our semi-supervised SVM algorithm with model selection through 100 independent simulations.

In each simulation, we first randomly generate a data set. Each data sample is a 16 dimensional feature vector. The vectors in the first class are generated by a 16 dimensional Gaussian distribution with identity covariance matrix and zero mean vector. The vectors in the second class are generated as follows: we first construct a mean vector $Me \in R^{16}$ and a diagonal covariance matrix $V \in R^{16\times 16}$. The entries of Me are drawn from a uniform distribution in $[0, 1.3]$, while the diagonal entries of V are drawn from a uniform distribution in $[0, 1]$. Then we use the 16 dimensional Gaussian distribution with covariance matrix V and mean vector Me to generate the vectors in the second class. The initial training data set contains 10 vectors with known labels, while the test set contains 490 data vectors without labels. We also generate an independent test set containing 100 data vectors to further validate our algorithm. Note that the independent test set is not used in the retraining process. There are 600 samples in total, of which 300 samples belong to the first class while the other 300 samples belong to the second class.

We apply Algorithm 1 to the above data set for classification. In this example, the number of iterations is fixed to 10 in order to see the details of iterations.

In each run, we first find a C value from 20 candidates using our model selection method in Section 2.2, where $C \in \{0.01, 0.02, \ldots, 0.2\}$. For the kth iteration of the jth run ($k = 1, \ldots, 10, j = 1, \ldots, 100$), we obtain two classification accuracy rates $\mathrm{rate}_t(C_0, k, j)$ and $\mathrm{rate}_{\mathrm{in}}(C_0, k, j)$ for the test data set and the independent test data set, respectively, where C_0 is the selected parameter for the jth run.

We now calculate the average prediction accuracy rates of $\mathrm{rate}_t(C_0, k, j)$ and $\mathrm{rate}_{\mathrm{in}}(C_0, k, j)$,

$$\overline{\mathrm{rate}}(k) = \frac{1}{200} \sum_{j=1}^{100} (\mathrm{rate}_t(C_0, k, j) + \mathrm{rate}_{\mathrm{in}}(C_0, k, j)), \tag{16}$$

where k represents the kth iteration.

In order to demonstrate the validity of our method for model selection, for each $C_i \in \{0.01, 0.02, \ldots, 0.2\}$, we also calculate the prediction accuracy rates $\mathrm{rate}_t(C_i, k, j)$ and $\mathrm{rate}_{\mathrm{in}}(C_i, k, j)$ for test data set and the independent test data set, respectively. We then calculate the average value of $\mathrm{rate}_t(C_i, k, j)$ and $\mathrm{rate}_{\mathrm{in}}(C_i, k, j)$ over C_i and j as follows:

$$\widetilde{\mathrm{rate}}(k) = \frac{1}{4000} \sum_{i=1}^{20} \sum_{j=1}^{100} (\mathrm{rate}_t(C_i, k, j) + \mathrm{rate}_{\mathrm{in}}(C_i, k, j)). \tag{17}$$

The left subplot of Fig. 1 shows two curves of the average prediction accuracy rates $\overline{\mathrm{rate}}(k)$ and $\widetilde{\mathrm{rate}}(k)$. The curve with "*" depicts $\overline{\mathrm{rate}}(k)$, which is obtained by Algorithm 1 with selection of C. The other one with "o" depicts $\widetilde{\mathrm{rate}}(k)$, which is obtained by Algorithm 1 without selection of C. From this subplot, we first see that the classification accuracy rates $\overline{\mathrm{rate}}(k)$ increase in the iterations of Algorithm 1 with model selection. By comparing the curves of $\overline{\mathrm{rate}}(k)$ and $\widetilde{\mathrm{rate}}(k)$, we also see that the model selection based on our method can significantly improve the performance of Algorithm 1.

Next, we check the convergence of Algorithm 1. For each of the 100 runs, we obtain C_0 through the selection of C. For the selected C_0, we also obtain the values of structural risk in (2). We denote the values of the structural risk as $S(k, j)$, where k and j represent the kth iteration and the jth run, respectively. We average $S(k, j)$ over 100 runs and obtain $\overline{S}(k)$. The average structural risk $\overline{S}(k)$ is shown in the middle subplot in Fig. 1. The decreasing tendency shows the convergence of Algorithm 1. In fact, for each run, the structural risk values also decrease with the iterations of Algorithm 1 (refer to Fig. 2).

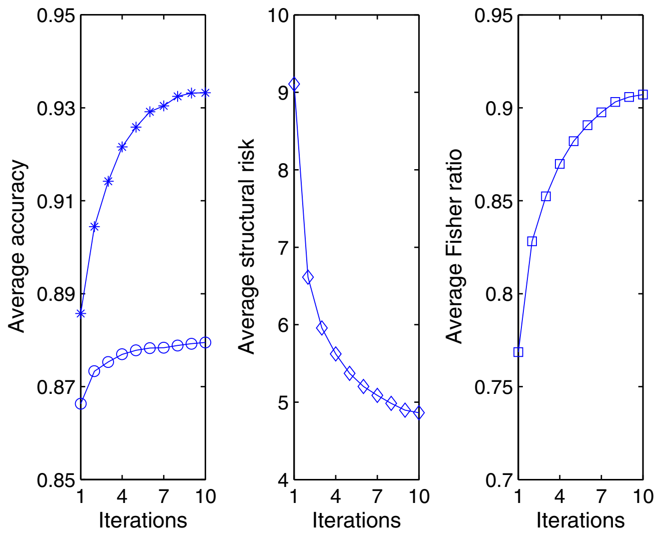

Fig. 1. Analysis results in Example 1. The left subplot shows two curves of average prediction accuracy rates $\overline{\mathrm{rate}}(k)$ ("$*$") and $\widetilde{\mathrm{rate}}(k)$ ("o"). The curves with stars are obtained by Algorithm 1 with model selection, the curves with circles are obtained by Algorithm 1 without model selection. The middle subplot shows the average structural risk, while the right subplot shows the average Fisher ratios.

We also calculate the Fisher ratio values $R(C_0, k, j)$ as in (3). Averaging $R(C_0, k, j)$ over 100 runs, we obtain the average Fisher ratios $\overline{R}(k)$. The average Fisher ratios $\overline{R}(k)$ are shown in the right subplot of Fig. 1. From this subplot, we can see that $\overline{R}(k)$ increase. We have not proved that $R(C, k)$ in (3) increase theoretically. However, we found in our extensive simulations that $R(C, k)$ increase with respect to the iterations of Algorithm 1 in general. This is the reason for using Fisher ratio for model selection in this paper. Although the structural risk decreases with the iterations of Algorithm 1, and small structural risk values may imply higher classification accuracy, it is not suitable for

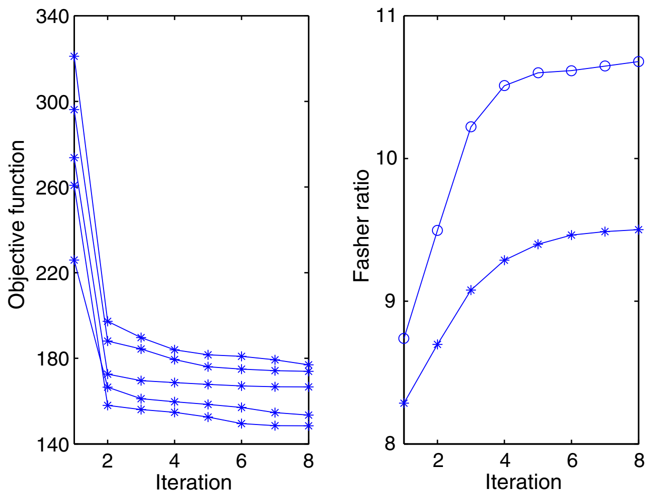

Fig. 2. Left: decreasing tendency of structural risk. Right: the curve with circles depicts the Fisher ratios obtained with the selected C_0, while the curve with stars depicts the average Fisher ratios over all candidates of C.

selecting the parameter C. This is because for the same data set, a small C may lead to a small structural risk value (see (2)).

Example 2. In this example, we demonstrate the validity and convergence of Algorithm 1 by real-world data analysis. We apply Algorithm 1 to five real-world data sets “Cancer”, “Wla”, “diabetes”, “Dimdata” and “german.numer_scale” (Newman et al., 1998). The dimensions and numbers of sample vectors of the five real-world data sets are listed in the second and third columns of Table 1, respectively.

For each data set, we perform a 5-fold cross-validation using Algorithm 1 with model selection. In each fold, the data set is divided into three parts, the first is called the initial training data set (D_0), the second is the test data set which is used in retraining, the third is an independent test set for further validation of Algorithm 1. The size ($|D_0|$) of the initial training data sets for the 5 real data sets are shown in the fourth column of Table 1. We select the size of D_0 according to two principles: (i) the algorithms works; (ii) $|D_0|$ is as small as possible. Thus $|D_0|$ depends on individual data set. For each data set, the ratio of the size of test data set and the independent test set is 4:1. After applying Algorithm 1 to each of the five real-world data sets, we obtain the accuracy rates for each of the 5-folds of the test set and the independent test set. Thus there are a total of 10 accuracy rates, which are then averaged. The average predication accuracy rate for each real-world data set is listed in the fifth column in Table 1 (The average accuracy rate for the independent test sets of each real-world data set is shown in the brackets).

We compare our algorithm with three other well-known semi-supervised learning algorithms, EM algorithm, a transductive SVM algorithm (SVMlight, http://svmlight.joachims.org/) and a Spectral Graph Transducer (SGTlight, http://sgt.joachims.org/), by performing a similar analysis for each of the five data sets. The average of the 10 accuracy rates obtained by EM is listed in the sixth column in Table 1. Using T-test, we compare the 10 accuracy rates obtained by Algorithm 1 and the 10 accuracy rates obtained by EM. The corresponding P value is listed in the seventh column. Similarly, the average of the 10 accuracy rates obtained by SVMlight and the corresponding P value are listed in the eighth and the ninth column, respectively, while the average of the 10 accuracy rates obtained by SGTlight and the corresponding P value are listed in the tenth and the eleventh column, respectively. For the three algorithms, the average accuracy rates for the independent test sets of each real-world data set are shown in the brackets.

From statistical tests, the performance of Algorithm 1 is significantly better than that of EM and SVMlight for the last 4 data sets. For the first data set, there is no significant difference between the performance of Algorithm 1 and that of EM and SVMlight. For the second data set, the performance of SGTlight is significantly better than our algorithm. For the fourth data set, there is no significant difference between the performance of our algorithm and that of SGTlight. For the other three datasets, our algorithm is significantly better than SGTlight in performance.

Here we would like to stress that the performance difference between Algorithm 1 and the other three algorithms may be due to the following reasons: First, performance of classification algorithms may depend on the nature of the classification task (Duda et al., 2001). Although Algorithm 1 and the other three algorithms can handle datasets with small training data, Algorithm 1 may be more suitable for certain data sets. Second, we performed model selection for Algorithm 1 as in Section 2.2. We also performed model selection for SVMlight using cross-validation, however we could not obtain promising results. This is because model selection based on cross-validation with such a small amount of training data for these particular datasets is not reliable. Thus we use the default settings for the parameters of SVMlight. Similarly, we also use the default parameters of SGTlight.

To demonstrate the convergence of Algorithm 1, we show five SVM objective function (structural risk) curves in the left subplot of Fig. 2, which are obtained in the 5-fold cross-validation for data set ‘Dia’. We can see that the five curves converge fast. The decrease of structural risk generally leads to a higher classification accuracy. This also explains the effectiveness of Algorithm 1.

In order to demonstrate that the Fisher ratios generally increase with respect to iterations of Algorithm 1, we present two curves of the Fisher ratios of 8 iterations for the data set ‘Dia’ in the right subplot of Fig. 2. They are obtained in the model selection in the first fold of the cross-validation. The curve with circles corresponds to the selected parameter C_0, while the curve with stars depicts the average Fisher ratios over all C values ($C \in \{0.05, 0.15, \ldots, 0.95\}$). The two curves show the decreasing tendency of Fisher ratios.

At the end of this section, we would like to state the advantage of Algorithm 1 in terms of computational

Table 1
Analysis results (accuracy rates % and p values) for five data sets

Data	Dimen.	Size	$\|D_0\|$	Algorithm 1	EM	P values	SVMlight	P value	SGTlight	P value
Cancer	10	680	4	96.37 (99.1)	*96.38* (98.5)	0.880	95.45 (94.3)	0.441	89.2 (92.5)	0.003
Wla	180	2000	50	87.9 (84.7)	84.7 (82.3)	0.010	77.7 (78.4)	0.000	*91.9* (89.8)	0.003
Dia	8	768	45	*77.8* (75.5)	68 (64.8)	0.027	64.3 (67.2)	0.008	64.2 (61.7)	0.008
Dim	14	2200	100	*91.4* (92.0)	67.3 (65.6)	0.000	69.8 (66.8)	0.000	88.2 (87.6)	0.15
German	24	1000	10	*97.4* (96.7)	91.6 (88.9)	0.0001	79 (83.4)	0.000	68 (61.4)	0.000

complexity. As shown in Figs. 1 and 2, this iterative algorithm can converge fast. According to our simulated and real-world data analysis examples, it generally converges in 10 iterations. This will be further demonstrated in the next section (the left subplot of Fig. 5). From Algorithm 1, the computation burden of each iteration is equivalent to that of a standard SVM. Thus Algorithm 1 has lower computational burden than that of a standard transductive SVM algorithm. For instance, when SVMlight is applied to "Dimdata" data set, the time taken is 5.28×10^5s in our computer (Intel Pentium 2 GHz processor and 2 GB RAM). This is because too many (1826) test data samples are used to retrain the classifier. The time taken for Algorithm 1 with model selection is around 69 s even when all 1826 test data samples are used in retraining the classifier.

4. Application in a BCI system

In this section, we illustrate the application of Algorithm 1 in a P300-based BCI speller. Currently, the electroencephalogram (EEG) is a prevailing brain signal for non-invasive BCIs. Event related potential (ERP), which exploits the electrophysiological responses measured by the EEG to a certain event such as the presentation of an external stimuli, is often used in BCIs. One robust feature is a positive displacement of EEG amplitude (ERP) occurring around 300 ms after onset of an unfrequent and expected stimulus. This is known as the P300 (see Fig. 3). P300 is a common feature in BCI spellers (Donchin, 1981; Farwell and Dochin, 1988; Serby et al., 2005). The main issue in P300-based BCIs is to classify the P300 response from the background noise. A generally tedious and time-consuming training process is required for P300-based BCIs, in order to build a reliable classification model for each subject. It is essential to reduce training effort so that P300-based BCI is more convenient to use.

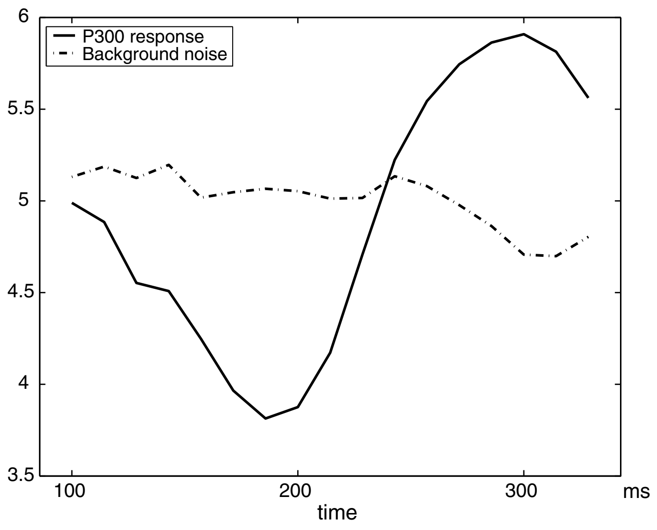

Fig. 3. An example of the average P300 response (μv) to a visual stimulus and average background noise.

In the following data analysis, we will show that Algorithm 1 can be used to reduce training effort in a P300-based speller. The dataset employed in this example was collected from 10 subjects by a P300-based speller paradigm. The experiment details and data collection are described in (Thulasidas et al., 2006). During the experiment, a 6-by-6 matrix that includes alphabets and numbers is presented to the user on a computer screen (Fig. 4). The user focuses on his desired character. Each row or column of the matrix is intensified successively in a random order. Each intensification lasts for 100 ms followed by a 75 ms blank interval. 12 intensifications make up one run which covers all the rows and columns of the matrix. Two of these twelve consecutive intensifications in each run contain the desired symbol, where P300 potential is generated. For each character, 10 runs of twelve intensifications are carried out. Thus when the user focuses on a character, there are $10 \times 12 = 120$ intensifications in which 20 intensifications happen on the column and row containing the character, the other 100 intensifications happen on the other 5 columns and 5 rows without the desired character. The task of the speller is to identify the user's desired character based on the EEG data collected in the 120 intensifications.

The dataset contains training data and testing data collected from 10 subjects. The system is trained separately for each user using a common phrase with 41 characters "THE QUICK BROWN FOX JUMPS OVER LAZY DOG 246138 579". The same phrase is also used in testing data collection with random word order. Using recognizable words or random characters makes no difference in terms of spelling accuracy. This is because the recognition of the desired character is only dependent on three factors: (1) the subject focuses his eyes on the desired character; (2) P300 potential occurs when the row and column of the matrix, which contain the desired character, are intensified; and (3) the algorithm of BCI system can detect the P300 potential correctly.

In our data analysis, the original test data is used as an independent test set, which is not used for retraining in Algorithm 1. The data corresponding to the first three characters of the original training data set is used as an initial training data set. The other 38 characters are used for retraining and testing.

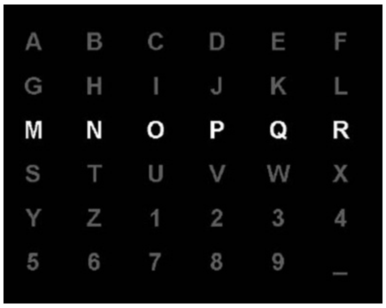

Fig. 4. The stimulus matrix shown on a computer monitor to the user for a P300 BCI speller. One row or one column of the matrix is intensified successively in a random order.

After lowpass filtering is performed on the EEG data, we construct a feature vector for each intensification as follows:

First, 24 EEG channels are used as in the original experiment. For the 24 channels, we extract 24 segments of EEG data, respectively. Each segment is between 150 ms and 500 ms from the start of the intensification, i.e. each segment is 350 ms and contains 87 data points (the sampling rate is 250 Hz). Next, we downsample these segments by a rate of 5. About 17 data points are obtained for each segment. Finally, by concatenating these data points of the 24 segments, a feature vector containing 408 (17×24) data points is constructed for an intensification.

As stated above, 120 intensifications occur for each character. Therefore, a character corresponds to 120 feature vectors. These feature vectors are categorized into two classes. The first class contains 20 feature vectors with their corresponding 20 intensifications occur on the row or column containing the desired character. Physiologically, P300 can be detected in each of these 20 feature vectors. The other 100 feature vectors which do not contain P300 belong to the second class.

For each subject, our initial training data set contains 360 feature vectors corresponding to 3 characters. Among the 360 feature vectors, only 60 feature vectors belong to the first class with P300, the other 300 feature vectors belong to the second class. Our test data set contains $38 \times 120 = 4560$ feature vectors corresponding to 38 characters. Our independent test data set contains $41 \times 120 = 4920$ feature vectors corresponding to 41 characters.

We now apply Algorithm 1 to the above data set. As a result, we obtain a SVM score for each feature vector of the test data set and independent test data set. We use these scores to predict the desired characters. Note that the 120 feature vectors can be equally divided into 12 groups. The first 6 groups correspond to 6 intensified rows, the last 6 groups correspond to 6 intensified columns, respectively. For each group, we sum their corresponding SVM scores and denote the summation as s_i, $i = 1, \ldots, 12$). Suppose that

$$s_{i_1} = \max\{s_1, \ldots, s_6\}, \quad s_{i_2+6} = \max\{s_7, \ldots, s_{12}\}.$$

Then the character contained in the i_1th row and the i_2th column is our predicted one. Furthermore, we calculate the prediction accuracy rates for test data set and independent test data set.

The prediction accuracy rates averaged over 10 subjects obtained by Algorithm 1 are given in the second row of Table 2. The accuracy rates in the third row are given for a standard SVM, which also uses the first 3 characters for training. The accuracy rate in fourth row is given for a standard SVM, which uses all the 41 characters for training. Since none of these characters are used for test, no accuracy rate is given for the test dataset in the fourth row. It can be seen from Table 2 that: Algorithm 1 can obtain a comparable accuracy rate (98.5%) as the standard SVM (99.0%), however the initial training set (3 characters) of Algorithm 1 is much smaller than that of the standard SVM (41 characters). Thus the training data collection time (training time) of the BCI speller can be potentially reduced much while not affecting the accuracy.

Table 2
Accuracy rates (%) for a data set from a P300-based BCI speller

	Test dataset	Ind. test dataset
Algorithm 1 (3 training symbols)	96.1	98.5
Standard SVM (3 training symbols)	79.2	84.4
Standard SVM (41 training symbols)	No accuracy	99.0

In the above offline analysis, we also use the data of 38 characters (without using their true labels) for retraining in Algorithm 1. For the kth iteration of Algorithm 1 with model selection, we calculate the average accuracy rate(k) over 10 subjects for test data sets and independent test data sets. With regards to model selection, we search the C value from the set $S = \{0.01, 0.06, \ldots, 0.46, 0.5\}$. The left subplot of Fig. 5 shows the curves (with "$*$") of average accuracy rates rate(k).

To demonstrate the validity of our model selection method as in Section 2.2, for each $C \in S$, we also calculate the prediction accuracy rates rate(C, k) similarly as above. We average rate(C, k) over all 11 possible candidates of C, and obtain $\overline{\text{rate}}(k)$ representing the average performance of Algorithm 1 without model selection. $\overline{\text{rate}}(k)$ are depicted by the curve with "o" in the left subplot of Fig. 5. From this subplot, we can see that rate(k) is obviously larger than

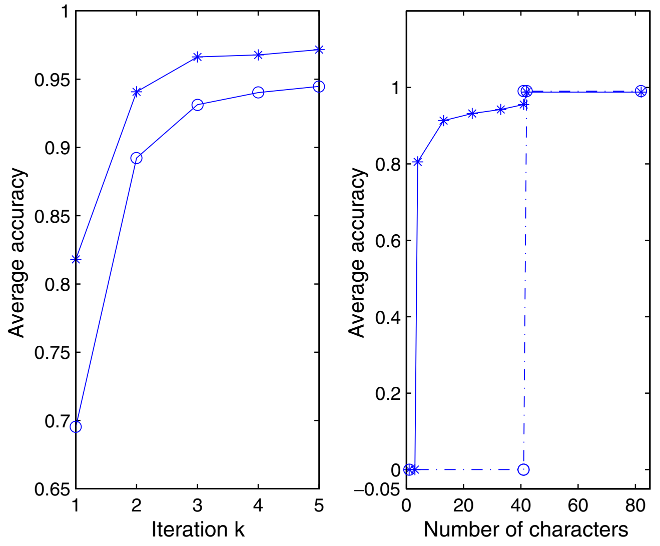

Fig. 5. Left: average accuracy curves obtained after each iteration of Algorithm 1 (the curve with "$*$" is obtained by Algorithm 1 with model selection, while the curve with "o" is obtained by Algorithm 1 without model selection. Right: Solid line with "$*$" is the accuracy curve obtained by an incremental version of Algorithm 1 which can be used in an online scenario. Dashed line with "o" is the accuracy curve obtained by the standard SVM which uses all the first 41 characters for training as in our original experiment. Using new incoming test data, Algorithm 1 can retrain the SVM and thus improve the adaptability of BCI system.

$\overline{\text{rate}}(k)$. This shows our model selection method is an improvement over no model selection.

Additionally, the accuracy curves in the left subplot of Fig. 5 further demonstrate that Algorithm 1 converges fast in about 5 iterations for this example.

Presently, one challenging issue in P300 speller is to considerably reduce this number of runs. State-of-the-art results can now reach almost perfect classification accuracy with as few as 4 runs. See for instance (Rakotomamonjy et al., 2005; Hoffmann et al., 2005; Rivet and Souloumiac, 2007). As shown above, 10 runs are used in this paper as in the experiment setting. We tried to reduce the number of runs in our data analysis. We found that when we used 6 runs, the results were still acceptable (not presented in this paper due to limited page space). But then the algorithm in this paper only can reduce the size of training data set instead of the number of runs. If a small number of runs leads to the BCI speller's not working even with a big training data set, then our algorithm also does not work. Of course, if l runs (l is quite small, e.g. 4) are sufficient for a BCI speller with a big training data set, our algorithm can be used for reducing the size of the training data set.

In the following, we simulate an online scenario to illustrate that an incremental version of Algorithm 1 can be used to improve the adaptability of the P300-based BCI speller in real time. First, we use the data of the first 3 characters to train a SVM. The SVM is used to classify the data of the subsequent 10 characters (4th–13th characters). Using the initial training data set and the data of these 10 characters with predicted labels, we retrain a new SVM using Algorithm 1 and classify the next 10 characters (14th–23rd characters) and so on. We stop retraining the SVM model after all the 38 characters are used. The independent test set are then classified by the finalized SVM model. The curve of average accuracy for the simulated online scenario is depicted by the solid line with "$*$" in the right subplot of Fig. 5. The dashed line with "o" is the average accuracy curve for the standard SVM, which uses all the first 41 characters for training as in our original experiment. Since no classification takes place for these 41 training characters, this is represented by a zero accuracy rate in the right subplot. The prediction accuracy rates of the last 41 characters obtained by Algorithm 1 and the standard SVM are almost equal in the simulated online scenario. However, the difference is: using an incremental version of Algorithm 1, we can start to classify incoming data much earlier than the standard SVM while keeping a satisfactory accuracy rate.

Furthermore, once the use feels that the accuracy of the system is not satisfactory, training of the SVM can be restarted using new incoming data. In this way, we can use Algorithm 1 to improve the adaptability of the BCI system. There is no substantial obstacle to implement the above learning in real time. When the subject is using the BCI speller, another computer (even the same one provided that it has sufficient computation speed) can simultaneously carry out the learning using the recorded data in real time. Once the learning is finished, the BCI system parameters can be updated using the new values.

5. Conclusions

In this paper, we present an iterative self-training semi-supervised SVM algorithm and a corresponding model selection method. This model selection method is based on Fisher ratio and it is suitable when the training data set is small and cross-validation-based model selection method may not work. The validity of this model selection method is demonstrated in our data analysis examples. As an iterative algorithm, the self-training semi-supervised SVM is proven to be convergent, although this convergence is local. By comparing with two semi-supervised learning algorithms, EM and SVMlight, in several real-world datasets, the effectiveness of this algorithm is demonstrated. This algorithm converges fast and has low computational burden. Finally, an application of our algorithm in a P300-based BCI speller is illustrated. This algorithm can be used to reduce training effort and improve adaptability of the P300-based BCI speller in an online scenario.

Acknowledgement

The authors are grateful for the insightful and constructive suggestions from the anonymous reviewers.

References

BCI2005, 2005. <http://ida.first.fraunhofer.de/projects/bci/competition_ii>.

Bennett, K.P., Demiriz, A., 1998. Semi-supervised support vector machines. In: Advances in Neural Information Processing Systems, vol. 12. MIT Press, Cambridge, MA, pp. 368–374.

Birbaumer, N., Ghanayim, N., Hinterberger, T., Iversen, I., Kotchoubey, B., kubler, A., Perelmouter, J., Taub, E., Flor, H., 1999. A spelling device for the paralysed. Nature 398, 297–298.

Blum, A., Mitchell, T., 1998. Combining labeled and unlabeled data with co-training. In: Proc. Conf. on Computational Learning Theory. pp. 92–100.

Brefeld, U., Scheffer, T., 2004. Co-EM support vector learning. In: Proc. 21st Internat. Conf. on Machine Learning, Canada.

Chang, C.C., Lin, C.J., 2001. LIBSVM – A library for support vector machines. <http://www.csie.ntu.edu.tw/cjlin/libsvm/>.

Chapelle, O., Zien, A., 2005. Semi-supervised classification by low density separation. In: Proc. 10th Internat. Workshop on Artificial Intelligence and Statistics, Barbados, pp. 57–64.

Demiriz, A., Bennett, K.P., 2000. Optimization approaches to semi-supervised learning. In: Ferris, C., Mangasarian, O.L., Pang, J.S. (Eds.), Applications and Algorithms of Complementarity. Kluwer Academic Publishers, Boston.

Donchin, E., 1981. Surprise! ... Surprise? Psychophysiology 18, 493–513.

Donoghue, J.P., 2002. Connecting cortex to machines: Recent advances in brain interfaces. Nat. Neurosci. Suppl. 5, 1085–1088.

Duda, R.O., Hart, P.E., Stork, D.G., 2001. In: Pattern Classification, vol. 454. John Wiley & Sons Inc.

Farwell, L.A., Dochin, E., 1988. Talking off the top of your head: Toward a mental prosthesis utilizing event-related brain potential. Electroencephalogr. Clin. Neurophysiol. 70, 510–523.

Grandvalet, Y., Bengio, Y., 2004. Semi-supervised learning by entropy minimization. In: Advances in Neural Information Processing Systems, vol. 16. MIT Press, Cambridge, MA.

Hoffmann, U., Garcia, G., Vesin, J.M., Diserens, K., Ebrahimi, T., 2005. A boosting approach to P300 detection with application to brain-computer interfaces. In: Proc. 2nd Internat. IEEE EMBS Conf. Neural Engineering.

Joachims, T., 1999. Transductive inference for text classification using support vector machines. In: Proc. Internat. Conf. on Machine Learning (ICML), Bled, Slovenia.

Joachims, T., 2003. Transductive learning via spectral graph partitioning. In: Proc. Internat. Conf. on Machine Learning (ICML), Washington, DC, USA.

Kiritchenko, S., Matwin, S., 2002. Email classification with co-training. Technical Report. University of Ottawa, Canada.

Kockelkorn, M., Lneburg, A., Scheffer, T., 2003. Using transduction and multi-view learning to answer emails. In: Proc. 7th European Conf. on Principles and Practice of Knowledge Discovery in Databases, Cavtat-Dubrovnik, Croatia, pp. 266–277.

Kubler, A., Kotchoubey, B., Kaiser, J.J.R., Birbaumer, N., 2001. Brain computer communication: Unlock the locked-in. Psychol. Bull. 127, C358–C375.

Mika, S., 2002. Kernel Fisher Discriminants, Ph.D. Thesis.

Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J., 1998. UCI Repository of machine learning databases, Department of Information and Computer Science. University of California, Irvine, CA. <http://www.ics.uci.edu/mlearn/MLRepository.html>.

Nigam, K., Ghani, R., 2000. Analyzing the effectiveness and applicability of co-training. In: Proc. 9th Internat. Conf. on Information and Knowledge Management, pp. 86–93.

Park, S., Zhang, B., 2004. Co-trained support vector machines for large scale unstructured document classification using unlabeled data and syntactic information. Inform. Process. Manag. 40 (3), 421–439.

Rakotomamonjy, A., Guigue, V., Mallet, G., Alvarado, V., 2005. Ensemble of SVMs for improving brain computer interface P300 speller performances. In: Proc. Internat. Conf. on Artificial Neural Networks, Varsovie.

Rivet, B., Souloumiac, A., 2007. Subspace estimation approach to P300 detection and application to brain-computer interface. In: Proc. IEEE EMBC, Lyon, France, pp. 5071–5074.

Serby, H., Yom-Tov, E., Inbar, G.F., 2005. An improved P300-based brain-computer interface. IEEE Trans. Neural Systems Rehabilit. Eng. 13 (1), 89–98.

Thulasidas, M., Guan, C., WU, J., 2006. Robust classification of EEG signal for brain-computer interface. IEEE Trans. Neural Systems Rehabilit. Eng. 14 (1), 24–29.

Vapnik, V., 1998. Statistical Learning Theory. Springer.

Xu, L., Jordan, M.I., 1996. On convergence properties of the em algorithm for gaussian mixtures. Neural Comput. 1, 129–151.

Zhou, D., Scholkopf, B., Hofmannz, T., 2004. Semi-supervised learning on directed graphs. In: Advances in Neural Information Processing Systems, vol. 16. MIT Press, Cambridge, MA.