Mach Learn (2008) 71: 33–53
DOI 10.1007/s10994-007-5039-1

Joint feature re-extraction and classification using an iterative semi-supervised support vector machine algorithm

Yuanqing Li · Cuntai Guan

Received: 27 November 2006 / Accepted: 30 October 2007 / Published online: 17 November 2007

Abstract The focus of this paper is on joint feature re-extraction and classification in cases when the training data set is small. An iterative semi-supervised support vector machine (SVM) algorithm is proposed, where each iteration consists both feature re-extraction and classification, and the feature re-extraction is based on the classification results from the previous iteration. Feature extraction is first discussed in the framework of Rayleigh coefficient maximization. The effectiveness of common spatial pattern (CSP) feature, which is commonly used in Electroencephalogram (EEG) data analysis and EEG-based brain computer interfaces (BCIs), can be explained by Rayleigh coefficient maximization. Two other features are also defined using the Rayleigh coefficient. These features are effective for discriminating two classes with different means or different variances. If we extract features based on Rayleigh coefficient maximization, a large training data set with labels is required in general; otherwise, the extracted features are not reliable. Thus we present an iterative semi-supervised SVM algorithm embedded with feature re-extraction. This iterative algorithm can be used to extract these three features reliably and perform classification simultaneously in cases where the training data set is small. Each iteration is composed of two main steps: (i) the training data set is updated/augmented using unlabeled test data with their predicted labels; features are re-extracted based on the augmented training data set. (ii) The re-extracted features are classified by a standard SVM. Regarding parameter setting and model selection of our algorithm, we also propose a semi-supervised learning-based method using the Rayleigh coefficient, in which both training data and test data are used. This method is suitable when cross-validation model selection may not work for small training data set. Finally, the results of data analysis are presented to demonstrate the validity of our approach.

Editor: Olivier Chapelle.

Y. Li (✉) · C. Guan
Institute for Infocomm Research, Singapore 119613, Singapore
e-mail: yqli2@i2r.a-star.edu.sg

Y. Li (✉)
e-mail: auyqli@yahoo.com

Keywords Supporter vector machine (SVM) · Semi-supervised learning · Rayleigh coefficient · Feature extraction · Model selection

1 Introduction

In many classification applications, collection labeled instances for training a model is either difficult, expensive, or time-consuming. However unlabeled data may be relatively easy 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 collecting labeled training data. Existing semi-supervised algorithms include expectation maximization (EM) algorithm, self-training algorithms (Nigam and Ghani 2000), 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. By extending vector machines with unlabeled data, transductive support vector machines (TSVMs) were developed in (Vapnik 1998). Since the optimization problem of a TSVM is non-convex and finding the exact TSVM solution is NP-hard, several approximations have been established (Joachims, 1999, 2002; Bennett and Demiriz 1998; Demiriz and Bennett 2000; Fung and Mangasarian 2001; Chapelle and Zien 2005). In several other studies (Park and Zhang 2004; Brefeld and Scheffer 2004; Kockelkorn et al. 2003; Kiritchenko and Matwin), 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 co-training naive Bayes (Kiritchenko and Matwin).

In pattern recognition and machine learning, it is often useful to apply a transformation to the raw data (i.e. feature extraction) before classification. This transformation can be linear, e.g. linear discriminant analysis (LDA), principal component analysis (PCA). It can also be nonlinear, e.g., kernel transformation. In many cases, determining the transformation for feature extraction needs a large amount of labeled data. One example is common spatial pattern (CSP) feature extraction in Electroencephalogram (EEG) data analysis and EEG-based brain computer interfaces (BCIs). The CSP feature of EEG signals, which corresponds to event-related de-synchronization (ERD) and event-related synchronization (ERS) evoked by motor imagery or movements, is very effective in discriminating several motor imageries (Pfutscheller et al. 1997; Blanchard and Blankertz 2004; Ramoser et al. 2000; Wolpaw et al. 2002). However, reliable CSP feature extraction relies on a time-consuming training process to gather labeled data, which is needed for determining a spatial filter matrix, also known as the CSP transformation matrix. Another example is in Fisher discriminant analysis (FDA) and linear discriminant analysis (LDA), in which we need to determine a transformation vector or matrix using labeled data. If the number of labeled data is not large enough, the extracted features are not reliable. This, in turn, will lead to a low accuracy rate in subsequent classification. Therefore extracting reliable features for small training data set is an important issue, which seems to have not been sufficiently addressed so far.

Extracting reliable features and performing classification simultaneously, when the training data set is small, is the main focus of this paper. We will first discuss feature extraction through Rayleigh coefficient maximization. One can find that CSP features can be explained under this framework. Furthermore, using the Rayleigh coefficient, we present two other features, one is for discriminating two classes with different means, the other is for discriminating two classes with different variances. In the extraction of these three features, a large

training data set is generally needed for determining a transformation matrix. If the training data set is small and these features are extracted directly, the features will not be reliable. In this case, standard semi-supervised learning methods will not work well (Zhou et al. 2003). We will propose a solution to this problem, using an iterative semi-supervised SVM algorithm for joint feature extraction and classification for a small training data set. As shown in (Chapelle et al. 2006), semi-supervised learning relies on certain assumptions, e.g. cluster assumption: If points are in the same cluster, they are likely to be of the same class. Since SVM is used as a classifier in this paper, our semi-supervised algorithm also relies on the cluster assumption.

In each iteration of the proposed algorithm, the training data set is updated using the test data (whose labels have been predicted in the previous iteration). In this way, we practically have more data for training. Based on the updated training data set, features are then re-extracted and classified by a standard SVM. The improvement in prediction accuracy in one iteration leads to higher quality of features in the next iteration, and the latter leads to a further improvement in the subsequent prediction accuracy and so on. This is the main difference between our method and the conventional semi-supervised algorithms.

There are several existing algorithms for classification with feature selection e.g. in (Krishnapuram et al. 2004; Weston et al. 2002). In comparison, our method is different from others: the features in our algorithm are iteratively re-extracted and classified, and the feature re-extraction is based on both training data set and test data set with predicted labels in each iteration. In (Li and Guan 2006), an extended EM algorithm for joint feature extraction and classification is proposed. Comparing the iterative semi-supervised SVM algorithm in this paper with that in (Li and Guan 2006), there are several major differences: (i) The classifier used in this paper is SVM rather than naive Bayes classifier as in (Li and Guan 2006); (ii) Our simulations show that the Rayleigh coefficient increases in the iterations of the algorithm of this paper. The effectiveness of the iterative semi-supervised SVM algorithm can be explained under the framework of the Rayleigh coefficient. (iii) A semi-supervised learning-based method is presented in this paper for parameter setting and model selection in small training data case. In this method, both training data and test data (without labels) are used. (iv) The feature used in (Li and Guan 2006) is limited to CSP features and the corresponding application is limited to EEG data analysis and BCIs. In this paper, we discuss feature extraction based on Rayleigh coefficient maximization. Besides the CSP feature, we also define two other features. Therefore, the range of applications of our iterative semi-supervised SVM algorithm can be extended beyond EEG data analysis.

The remainder of this paper is organized as follows. We discuss feature extraction based on Rayleigh coefficient maximization in Sect. 2. An iterative semi-supervised SVM algorithm is proposed in Sect. 3. We also present a method for parameter setting and model selection in small training data case. Three examples of data analysis are presented in Sect. 4 to demonstrate the validity of our algorithm. Conclusions in Sect. 5 review the approach in this paper.

2 Feature extraction based on Rayleigh coefficient maximization

In this section, we first present a brief description of Rayleigh coefficient maximization, then discuss the extraction of three features under this framework.

2.1 Rayleigh coefficient maximization

Many algorithms for feature extraction can be deduced by maximizing the following Rayleigh coefficient (Mika 2002),

$$\max J(\mathbf{q}) = \frac{\mathbf{q}^T \mathbf{S}_I \mathbf{q}}{\mathbf{q}^T \mathbf{S}_N \mathbf{q}}, \tag{1}$$

where $\mathbf{S}_I$ and $\mathbf{S}_N$ are symmetric $m \times m$ matrices designed such that they can measure the desired information and the undesired noise along the direction of $\mathbf{q}$.

For example, in Fisher discriminant analysis, $\mathbf{S}_I$ and $\mathbf{S}_N$ are the between class scatter and the within class scatter matrices respectively.

The ratio in (1) is maximized when one covers the desired information as much as possible while avoiding the undesired. The solution of (1) can be obtained by solving the following generalized eigenproblem,

$$\mathbf{S}_I \mathbf{q} = \lambda \mathbf{S}_N \mathbf{q} \tag{2}$$

where λ is a generalized eigenvalue, and $\mathbf{q}$ is a generalized eigenvector corresponding to λ. Note that there are m generalized eigenvectors, which can be obtained by jointly diagonalizing two matrices $\mathbf{S}_I$ and $\mathbf{S}_N$.

2.2 CSP feature extraction

CSP features of EEG signals are very effective in discriminating different motor imageries and are commonly used in BCIs and EEG data analysis. In the following, we will briefly illustrate CSP feature extraction and explain how CSP feature extraction can be deduced under the framework of the Rayleigh coefficient. More details on CSP feature extraction can be found in (Blanchard and Blankertz 2004) etc.

Suppose that $\{(\mathbf{A}(j), y(j)), j = 1, \ldots, N\}$ is a training data set for two classes, where $\mathbf{A}(j) \in R^{m \times L}$, label $y(j) = 1$ or -1. In EEG data, m and L represent the numbers of channels and time samples respectively. Let us define the second order correlation matrices $\Gamma^{(1)}$ and $\Gamma^{(2)}$,

$$\Gamma^{(1)} = \sum_{j \in CL_1} \frac{\mathbf{A}(j)\mathbf{A}^T(j)}{\operatorname{trace}(\mathbf{A}(j)\mathbf{A}^T(j))}, \qquad \Gamma^{(2)} = \sum_{j \in CL_2} \frac{\mathbf{A}(j)\mathbf{A}^T(j)}{\operatorname{trace}(\mathbf{A}(j)\mathbf{A}^T(j))}, \tag{3}$$

where CL_1 and CL_2 represent the two classes of training data.

First, we calculate a transformation matrix $\mathbf{Q}$ to jointly diagonalize matrices $\Gamma^{(1)}$ and $\Gamma^{(2)}$ (Vapnik 1998). Noting that $\Gamma^{(1)}$ and $\Gamma^{(2)}$ are two symmetric matrices, we can find an orthogonal matrix $\mathbf{U}$ with its first row being nonnegative such that

$$\mathbf{U}^T(\Gamma^{(1)} + \Gamma^{(2)})\mathbf{U} = \Lambda, \tag{4}$$

where $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_m)$, $\lambda_1, \ldots, \lambda_m$ are the positive eigenvalues of $\Gamma^{(1)} + \Gamma^{(2)}$ arranged in descending order.

Let

$$\bar{\mathbf{U}} = \mathbf{U}\Lambda^{-\frac{1}{2}}, \tag{5}$$

then we have

$$\bar{\mathbf{U}}^T(\Gamma^{(1)} + \Gamma^{(2)})\bar{\mathbf{U}} = \mathbf{I}, \tag{6}$$

where $\mathbf{I}$ is an identity matrix.

Define

$$\bar{\Gamma}^{(1)} = \bar{\mathbf{U}}^T \Gamma^{(1)} \bar{\mathbf{U}} = \Lambda^{-\frac{1}{2}} \mathbf{U}^T \Gamma^{(1)} \mathbf{U} \Lambda^{-\frac{1}{2}}. \tag{7}$$

Obviously, $\bar{\Gamma}^{(1)}$ is still a symmetric matrix. Diagonalize $\bar{\Gamma}^{(1)}$ as follows,

$$\mathbf{V}^T \bar{\Gamma}^{(1)} \mathbf{V} = \mathrm{diag}(\tilde{\lambda}_1, \ldots, \tilde{\lambda}_m), \tag{8}$$

where $\mathbf{V}$ is a orthogonal eigenvector matrix of $\bar{\Gamma}^{(1)}$ with its first row being nonnegative, and $\tilde{\lambda}_1, \ldots, \tilde{\lambda}_m$ are the eigenvalues of $\bar{\Gamma}^{(1)}$, which are sorted in a descending order.

Set

$$\mathbf{Q} = \mathbf{U} \Lambda^{-\frac{1}{2}} \mathbf{V}. \tag{9}$$

Then $\mathbf{Q}$ jointly diagonalize matrices $\Gamma^{(1)}$ and $\Gamma^{(2)}$.

Next, we construct a matrix $\bar{\mathbf{Q}}$, which is composed of the first l_1 and the last l_2 columns of $\mathbf{Q}$ (Blanchard and Blankertz 2004). For data matrix $\mathbf{X}_j \in R^{m \times L}$, the CSP feature vector is defined as

$$\mathbf{x}_{csp}(j) = \mathrm{diag}(\bar{\mathbf{Q}}^T \mathbf{X}_j \mathbf{X}_j^T \bar{\mathbf{Q}}) = [\|\mathbf{f}_1\|_2^2, \ldots, \|\mathbf{f}_{l_1+l_2}\|_2^2]^T, \tag{10}$$

where $\mathbf{f}_1, \ldots, \mathbf{f}_{l_1+l_2}$ denote the $l_1 + l_2$ rows of the transformed data matrix $\bar{\mathbf{Q}}^T \mathbf{X}_j$.

Remark 1 The joint diagonalization matrix $\mathbf{Q}$ defined in (9) is generally unique. This can be explained as follows. From the theory of linear algebra, if a symmetric $m \times m$ matrix has m different eigenvalues, then any two eigenvectors corresponding to two different eigenvalues are orthogonal, and each eigenspace corresponding to a eigenvalue is one dimensional. Thus the eigenvector corresponding to an eigenvalue is unique up to a scale. Since $\Gamma^{(1)} + \Gamma^{(2)}$ is symmetric and it has m different eigenvalues generally, the normalized and orthogonal matrix $\mathbf{U}$ in (4), with its first row set to nonnegative, is unique. Furthermore, the normalized and orthogonal matrix $\mathbf{V}$ in (8), with its first row set to nonnegative, is also unique. This is because the symmetric matrix $\bar{\Gamma}^{(1)}$ in (7) generally has m different eigenvalues: Considering that $\mathbf{U}$ is orthogonal, $\mathbf{U}^T \Gamma^{(1)} \mathbf{U}$ has the same eigenvalues as those of $\Gamma^{(1)}$. If $\Gamma^{(1)}$ has m nonzero eigenvalues, then $\mathbf{U}^T \Gamma^{(1)} \mathbf{U}$ also has m nonzero eigenvalues. By multiplying the diagonal matrix $\Lambda^{-\frac{1}{2}}$ (with m different diagonal entries) on both sides of $\mathbf{U}^T \Gamma^{(1)} \mathbf{U}$, we observe that this will result in m different eigenvalues in $\bar{\Gamma}^{(1)} = \Lambda^{-\frac{1}{2}} \mathbf{U}^T \Gamma^{(1)} \mathbf{U} \Lambda^{-\frac{1}{2}}$. When $\mathbf{U}$ and $\mathbf{V}$ are uniquely determined, $\mathbf{Q}$ is unique.

From (10), the CSP feature vector can be regarded as a variance vector, of which each entry, $\|\mathbf{f}_i\|_2^2$, is the variance of one row in the transformed data matrix $\bar{\mathbf{Q}}^T \mathbf{X}_j$. Therefore, we can see that CSP features can be used for discriminating the two classes with different variances.

In the following, we show the above CSP feature extraction can be implemented by maximizing a Rayleigh coefficient and hence solving a generalized eigenvalue problem.

We define $\mathbf{S}_I$ and $\mathbf{S}_N$ in (1) as

$$\mathbf{S}_N = \Gamma^{(1)} + \Gamma^{(2)}, \qquad \mathbf{S}_I = \Gamma^{(1)} - \Gamma^{(2)}. \tag{11}$$

Note that for $\mathbf{S}_I$ and $\mathbf{S}_N$ defined in (11), the Rayleigh coefficient in (1) might be negative. In this case, we will find the maximal absolute value of the Rayleigh coefficient.

Through Rayleigh coefficient maximization, a matrix which jointly diagonalizes matrices $\mathbf{S}_I$ and $\mathbf{S}_N$ is obtained. It is not difficult to prove that this matrix also jointly diagonalizes matrices $\Gamma^{(1)}$ and $\Gamma^{(2)}$. That is to say, the transformation matrix $\mathbf{Q}$ in CSP feature extraction can be obtained by Rayleigh coefficient maximization.

2.3 Fisher discriminant feature extraction

In FDA for two class problems, $\mathbf{S}_I$ and $\mathbf{S}_N$ in (1) are defined as

$$\mathbf{S}_I = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T, \qquad \mathbf{S}_N = \sum_{i=1,2} \sum_{\mathbf{x} \in C_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^T, \tag{12}$$

where $\mathbf{m}_1$ and $\mathbf{m}_2$ are the mean vectors of two classes, C_1 and C_2 represent two classes. Using $\mathbf{S}_I$ and $\mathbf{S}_N$ in (12), we can define a Rayleigh coefficient as in (1).

Since the rank of $\mathbf{S}_I$ in (12) is one, there is only one generalized eigenvector (up to a scale) which maximizes the Rayleigh coefficient in (1). This eigenvector corresponds to the nonzero generalized eigenvalue. The other $m - 1$ independent generalized eigenvectors correspond to eigenvalue zero. Therefore, through the standard Fisher discriminant (FD) approach, we cannot obtain a transformation matrix such that all its column vectors maximize the corresponding Rayleigh coefficient. In the following, we use a regularization approach for feature extraction through defining

$$\mathbf{S}_I = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T + \alpha \mathbf{I}, \qquad \mathbf{S}_N = \sum_{i=1,2} \sum_{\mathbf{x} \in C_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^T, \tag{13}$$

where α is a predetermined positive parameter. In Appendix 1, we will explain the principle for setting α: α should be set small such that the largest generalized eigenvalue and corresponding generalized eigenvector are close to those obtained in normal FDA with $\alpha = 0$. In this way, the advantage of normal FDA in maximizing the Rayleigh coefficient can be retained. In this paper, α is always set to 0.05.

In (Mika et al. 1999), a regularization was imposed on the $\mathbf{S}_N$ to deal with the ill-posed case (e.g. $\mathbf{S}_N$ may be close to singular and not positive definite). In this paper, we impose regularization on $\mathbf{S}_I$ rather than $\mathbf{S}_N$. As will be seen, $\mathbf{S}_N$ is calculated using training data and large amounts of test data in our algorithm. It is positive definite and nonsingular. Our main objective is to extract multi-dimensional features. The benefit of multi-dimensional feature extraction can be seen in Appendix 1 (Part B).

Through Rayleigh coefficient maximization, we obtain a matrix $\mathbf{Q}$ which jointly diagonalizes the two matrices $\mathbf{S}_I$ and $\mathbf{S}_N$ defined in (13). We then construct a transformation matrix $\bar{\mathbf{Q}}$, which is composed by the first l_1 columns of $\mathbf{Q}$, for feature extraction.

Remark 2 The above FD feature extraction is also suitable for the case where data samples are matrices. In this case, $\mathbf{m}_1$ and $\mathbf{m}_2$ are mean matrices of two classes. $(\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T$ may be of full rank. If it is the case, α in (13) can be set to zero.

After the transformation matrix $\bar{\mathbf{Q}}$ is determined, we define two features here.

(i) For data vector $\mathbf{X}_j$, we define feature vector $\mathbf{x}_{fd1}(j)$ as

$$\mathbf{x}_{fd1}(j) = \bar{\mathbf{Q}}^T \mathbf{X}_j. \tag{14}$$

(ii) For a data vector or a data matrix $\mathbf{X}_j$, we define feature vector $\mathbf{x}_{fd2}(j)$ as

$$\mathbf{x}_{fd2}(j) = \mathrm{diag}(\bar{\mathbf{Q}}^T \mathbf{X}_j \mathbf{X}_j^T \bar{\mathbf{Q}}). \tag{15}$$

Hereafter, features $\mathbf{x}_{fd1}$ and $\mathbf{x}_{fd2}$ are called feature FD1 and feature FD2 respectively. Similar to the CSP feature, feature FD2 is also suitable for discriminating the two classes with different variances. Additionally, FD_2 feature extraction is similar with a matrix style FDA (Kong et al. 2005) when the data is two dimensional.

Through regularization in (13), we can extract multi-dimensional features as above. Furthermore, we can find that $\mathbf{S}_I$ in (13) has m nonzero eigenvalues, and $\mathbf{S}_N$ has m different eigenvalues generally. From Remark 1, the joint diagonalization matrix $\mathbf{Q}$ and the extracted features can be uniquely determined. This is another benefit brought by regularization. In fact, the first column of $\mathbf{Q}$ can be close to that of FDA when the parameter α in (13) is sufficiently small (see Appendix 1). The first column of $\mathbf{Q}$ maximizes the numerator of (1) i.e. inter-class scatter, and simultaneously minimizes the denominator of (1) i.e. within-class scatter. The other columns mainly minimize the denominator of (1). This is because all the last $m-1$ eigenvalues of $\mathbf{S}_I$ in (13) are equal to α.

We now compare the CSP feature with the above two features from two aspects. (i) From the definitions of the above three features, only feature FD1 in (14) is for discriminating two classes with different means. the CSP feature and feature FD2 in (15) discriminate two classes with different variances. (ii) Although feature FD2 and the CSP feature have similar definitions (see (10) and (15)), their transformation matrices (all denoted as $\bar{\mathbf{Q}}$) are obtained by maximizing different Rayleigh coefficients. In Table 1, we summarize these three features in this paper.

3 An iterative semi-supervised SVM algorithm for joint feature extraction and classification

In this section, we will propose an iterative semi-supervised SVM algorithm for joint feature extraction and classification. Two important problems, the effectiveness of the algorithm, parameter setting and model selection, are also discussed.

3.1 Algorithm

From the discussion in Sect. 2, a training data set is needed to determine a transformation matrix $\bar{\mathbf{Q}}$ for feature extraction. This transformation matrix is obtained by jointly diagonalizing two matrices $\mathbf{S}_I$ and $\mathbf{S}_N$, where the definitions of $\mathbf{S}_I$ and $\mathbf{S}_N$ depend on the features used here (see Table 1). Since the entries of these two matrices are statistically estimated, it is desired to have sufficient training data for good estimation. However, the process for collecting training (labeled) data is time consuming in general. It is very important to reduce the training effort (for collecting labeled data) in many real-world applications. In the following, we present an iterative semi-supervised SVM algorithm, which jointly performs feature extraction and classification for small training data cases.

In each iteration of our algorithm, the following standard SVM is used for classification.

$$\begin{aligned} \min \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\ \text{subject to } & y_i(\mathbf{w}^T\mathbf{x}(i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0,\ i = 1, \ldots, N, \end{aligned} \tag{16}$$

Table 1 Summary of features

	CSP feature	FD1 feature	FD2 feature
$\mathbf{S}_I$	$\Gamma^{(1)} - \Gamma^{(2)}$	$(\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T + \alpha\mathbf{I}$	$(\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T + \alpha\mathbf{I}$
$\mathbf{S}_N$	$\Gamma^{(1)} + \Gamma^{(2)}$	$\sum_{i=1,2}\sum_{\mathbf{x}\in C_i}(\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^T$	$\sum_{i=1,2}\sum_{\mathbf{x}\in C_i}(\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^T$
Data type	Matrix or vector	vector	Matrix or vector
Definition	$\mathrm{diag}(\bar{\mathbf{Q}}^T\mathbf{X}_j\mathbf{X}_j^T\bar{\mathbf{Q}})$	$\bar{\mathbf{Q}}^T\mathbf{X}_j$	$\mathrm{diag}(\bar{\mathbf{Q}}^T\mathbf{X}_j\mathbf{X}_j^T\bar{\mathbf{Q}})$
Discriminate means/variances	variances	means	variances

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

In the kth iteration of the following algorithm, we first re-extract n dimensional feature vectors denoted as $\mathbf{x}_k$, then apply SVM to these feature vectors for classification. Note that n is an hyperparameter. The feature refers to one of the three types of features, which have been presented in Sect. 2.

Algorithm 1

Step 1 (Initialization) Two raw data sets are given: D_I with labels and D_T without labels. We use D_I for initial training, and use D_T for testing. Suppose that D_I and D_T contain N_1 and N_2 data samples respectively. For the first iteration, we train a transformation matrix $\bar{\mathbf{Q}}_1$ based on D_I and its labels. Using $\bar{\mathbf{Q}}_1$, we extract features for both D_I and D_T. The features of D_I are denoted as $\mathbf{x}_1(i)$, $i = 1, \ldots, N_1$, while the features of D_T are denoted as $\mathbf{x}_1(i)$, $i = N_1 + 1, \ldots, N_1 + N_2$. With the features $\{\mathbf{x}_1(i), i = 1, \ldots, N_1\}$ and their corresponding labels, we train a SVM which is described in (16). We then apply this SVM to the features $\{\mathbf{x}_1(i), i = N_1 + 1, \ldots, N_1 + N_2\}$ to perform classification for D_T. We obtain the predicted labels of D_T and denote them as $[y_1(1), \ldots, y_1(N_2)]$ (where the subscript 1 of $\bar{\mathbf{Q}}_1$ and y represents the first iteration).

Step 2 The kth iteration ($k = 2, 3, \ldots$) follows Steps 2.1–2.4.

Step 2.1 (Update the training data set) We form a new training data set D_k using the samples of D_I (with given labels) and the samples of D_T (with predicted labels), i.e. $D_k = D_I + D_T$. The predicted labels of D_T are: $[y_{k-1}(1), \ldots, y_{k-1}(N_2)]$ obtained in the $(k-1)$th iteration. In fact, the samples of D_k do not change, only the labels of D_T, $[y_{k-1}(1), \ldots, y_{k-1}(N_2)]$, are updated in this step.

Step 2.2 (Feature re-extraction) Next, the features $\{\mathbf{x}_{k-1}(i), i = 1, \ldots, N_1 + N_2\}$ extracted from D_I and D_T in the previous iteration are updated. Based on the training data set D_k, we regenerate a transformation matrix denoted as $\bar{\mathbf{Q}}_k$ for feature extraction. Using $\bar{\mathbf{Q}}_k$, we re-extract features for both D_I and D_T. The features of D_I are denoted as $\mathbf{x}_k(i)$, $i = 1, \ldots, N_1$, while the features of D_T are denoted as $\mathbf{x}_k(i)$, $i = N_1 + 1, \ldots, N_1 + N_2$ (the subscript k refers to the kth iteration).

Step 2.3 (Classification) A new SVM is trained using the re-extracted feature vectors $\{\mathbf{x}_k(i), i = 1, \ldots, N_1 + N_2\}$ and their corresponding labels. Note that the labels of $\{\mathbf{x}_k(i), i = N_1 + 1, \ldots, N_1 + N_2\}$ used here are $[y_{k-1}(1), \ldots, y_{k-1}(N_2)]$. We then perform classification for D_T using its

feature vectors $\{\mathbf{x}_k(i), i = N_1 + 1, \ldots, N_1 + N_2\}$, and obtain the new predicted labels denoted as $[y_k(1), \ldots, y_k(N_2)]$.

Step 2.4 Calculate the ratio which reflects the difference of the labels of the test data set predicted in the kth and $(k-1)$th iteration,

$$r(k) = \frac{\sum_{i=1}^{N_2} |y_k(i) - y_{k-1}(i)|}{2N_2}. \tag{17}$$

Step 3 (Termination) If $|r(k)| < \alpha_0$ or if $k = K_0$, then the algorithm terminates after the kth iteration, where α_0 is a pre-determined positive constant, K_0 is a pre-determined positive integer. $[y_k(1), \ldots, y_k(N_2)]$ for the test set D_T are the final classification results. Otherwise, go to Step 2 to perform the $(k+1)$th iteration.

In Step 3 of Algorithm 1, we use two parameters α_0 and K_0 to terminate the iterations. Generally, we can set α_0 small in order to make the labels predicted in the last two consecutive iterations as consistent as possible. If α_0 is too small to terminate the iterations, then the algorithm will end after K_0 iterations. From our extensive simulations, we observe that this algorithm typically converges in 10 iterations. Thus we suggest that K_0 can be set less than 15. In this paper, α_0 and K_0 are set to 0.005 and 10 respectively.

If the feature used in Algorithm 1 is a CSP feature, then the transformation matrices $\bar{\mathbf{Q}}_k$ is trained with the two matrices $\mathbf{S}_I$ and $\mathbf{S}_N$ defined in (11). For the other two types of features, the transformation matrices $\bar{\mathbf{Q}}_k$ is trained with the two matrices $\mathbf{S}_I$ and $\mathbf{S}_N$ defined in (13).

As in general semi-supervised learning algorithms, Algorithm 1 uses its own predictions to improve its own performance. This may lead to a problem of "data-incest" where errors are re-enforced along with the correct classifications. One way to deal with this problem is to make a "smarter" use of the predicted labels other than simply using them again (Tong and Koller 2001; Zhu et al. 2003). This may further improve the quality of features and classification accuracy. Thus combining feature selection in a more subtle way with a smarter use of the predicted labels could be a direction of future research.

3.2 Discussions

In this subsection, we first discuss parameter setting and model selection, then present some explanation on the effectiveness of our algorithm.

A. Parameter setting and model selection

In Algorithm 1, we need to set two important parameters, the dimension of the feature vectors, n (the number of columns of $\bar{\mathbf{Q}}_k$) and the regularization parameter C of SVM. We do not use cross-validation on training data set to search C and n due to two reasons: First, Algorithm 1 is designed to work with small training data set. Cross-validation in small training data set may not be reliable; Second, features extracted from a small training data set also may not be reliable for parameter setting and model selection.

Now we present a semi-supervised learning-based method for determining C and n by the help of test data (without labels) and the Rayleigh coefficient. In this method, we search a combination of C and n values on a finite grid with $C \in \{C_1, \ldots, C_{K_1}\}$ and $n \in \{n_1, \ldots, n_{K_2}\}$ as in LDS algorithm (Chapelle and Zien 2005).

Under a given combination of C and n, we run Algorithm 1. In the kth iteration, we calculate the Rayleigh coefficient,

$$R(C,n,k)=\begin{cases}\dfrac{(\mathbf{q}_1^{(k)})^T\mathbf{S}_I^{(k)}\mathbf{q}_1^{(k)}}{(\mathbf{q}_1^{(k)})^T\mathbf{S}_N^{(k)}\mathbf{q}_1^{(k)}}, & \text{for FD1 and FD2 features,}\\ \dfrac{(\mathbf{q}_1^{(k)})^T\mathbf{S}_I^{(k)}\mathbf{q}_1^{(k)}}{(\mathbf{q}_1^{(k)})^T\mathbf{S}_N^{(k)}\mathbf{q}_1^{(k)}}+\left|\dfrac{(\mathbf{q}_m^{(k)})^T\mathbf{S}_I^{(k)}\mathbf{q}_m^{(k)}}{(\mathbf{q}_m^{(k)})^T\mathbf{S}_N^{(k)}\mathbf{q}_m^{(k)}}\right|, & \text{for CSP feature,}\end{cases} \tag{18}$$

where $\mathbf{q}_1^{(k)}$ and $\mathbf{q}_m^{(k)}$ are the 1st and the mth column of the transformation matrix $\bar{\mathbf{Q}}_k$, which correspond to the biggest and the smallest generalized eigenvalues respectively. Note that for CSP feature extraction, the smallest generalized eigenvalue is negative. $\mathbf{S}_I^{(k)}$ and $\mathbf{S}_N^{(k)}$ are determined using the new training data set D_k containing test data with predicted labels (see Algorithm 1). The definitions of $\mathbf{S}_I^{(k)}$ and $\mathbf{S}_N^{(k)}$ depend on the type of feature used here (see Table 1).

During the procedure for selecting C and n, we always let Algorithm 1 end after K_0 iterations. Thus for different combinations of C and n, we obtain the Rayleigh coefficients $R(C,n,1),\ldots,R(C,n,K_0)$. Considering that $R(C,n,1)$ is calculated with the small training data set and may not be reliable, we find the maximum of the last K_0-1 Rayleigh coefficients,[1]

$$R_m(C,n)=\max\{R(C,n,2),\ldots,R(C,n,K_0)\}. \tag{19}$$

Suppose that C_0 and n_0 is the combination of C and n such that the corresponding $R_m(C,n)$ is the maximum on the grid. Then C_0 and n_0 are the selected parameter values.

The above method for selecting C and n is mainly based on the fact: The Rayleigh coefficient generally represents the separability of a corresponding data set, i. e., bigger Rayleigh coefficient generally implies higher separability of the data set. We choose the combination of C and n which leads to the highest Rayleigh coefficient in training data set and test data set (with predicted labels). This is a little similar as the case of transductive SVM, in which the labels of test data set leading to the lowest structural risk are chosen. In Examples 1 and 3 of this paper, we will demonstrate the validity of this method.

B. Effectiveness

We explain the effectiveness of our approach as follows:

First, feature re-extraction in Algorithm 1 can improve the separability of features. Reliable FDA feature extraction needs a large amount of labeled training data. If the labeled data set is small and thus insufficient, we cannot directly use FDA for feature extraction. Algorithm 1 provides us a way to use FDA in small training data case. That is, in each iteration of Algorithm 1, we first augment the training data set using test data with predicted labels from the previous iteration, then perform FDA feature extraction. We apply Algorithm 1 to a BCI data set (see Example 3 in Sect. 4 for data description) to extract CSP feature vectors. Figure 1 shows the scatter plots of the first two dimensions of these CSP feature vectors of both training and test data. The features extracted in the first iteration are shown in the left subplot, while the features extracted after the 7th iteration are shown in the right subplot. By comparison, we can see that feature re-extraction of Algorithm 1 can improve the separability of features of two classes. This is mainly because Rayleigh coefficient maximization generally improves the separability of features (see Appendix 2).

Second, from our simulations, we found that for given C and n, the Rayleigh coefficients $R(C,n,k)$ (defined in (18)) increases in most cases, while the terminating criterion $r(k)$

[1]Note: $R_m(C,n)$ in (19) also can be defined as the mean of $R(C,n,2),\ldots,R(C,n,K_0)$.

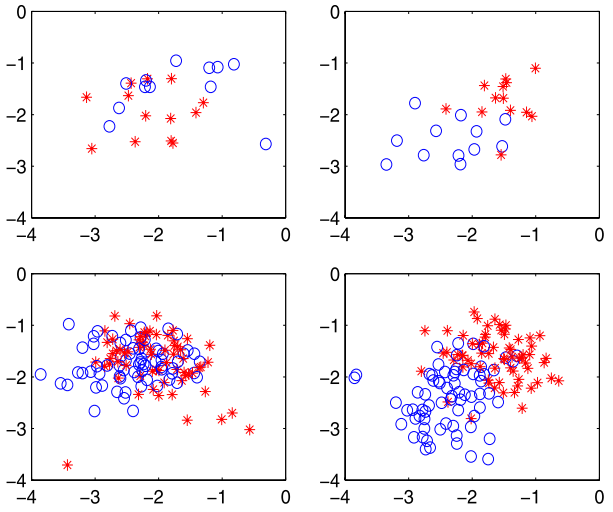

Fig. 1 An illustrative example on feature re-extraction in Algorithm 1. *Left*: Features extracted in the first iteration; *Right*: Features extracted in the 7th iteration; *Top*: Features from the training data set; *Bottom*: Features from the test data set. Stars and circles represent the two classes respectively in the four subplots

defined in (17) decreases with respect to k. Furthermore, $r(k)$ converges fast (generally in less than 10 iterations). Thus, Algorithm 1 shows satisfactory convergence in most cases (see Tables 2 and 3 in Example 1). Furthermore, a significant increase in the Rayleigh coefficient after each iteration in Algorithm 1 leads to higher prediction accuracy rates, and vice versa.

Third, as mentioned before, the parameters C and n are also selected based on the Rayleigh coefficient. The selected combination of C and n leads to the biggest Rayleigh coefficient, which implies better features and higher classification accuracy (see Examples 1 and 3).

Remark 3 (i) In Algorithm 1, we can use EM in place of SVM (Li and Guan 2006). We will try other classifiers such as Hidden Markov Models (HMM) in the future. The corresponding iterative algorithm has the same working mechanism as Algorithm 1. (ii) How to determine whether a classifier can be used in place of SVM in Algorithm 1? The principle for judgment is: the Rayleigh coefficient increases in the iterations of the corresponding algorithm.

4 Experimental results

In this section, three examples are shown to demonstrate the validity of Algorithm 1. In Example 1, we mainly illustrate the validity and convergence of Algorithm 1. The main objective of Example 2 is to compare our algorithm with baseline algorithms. In Example 3, we show that Algorithm 1 with CSP feature re-extraction can be used in BCI systems. In these three examples, the regularization parameter C of SVM and the dimension number n of feature vectors, i.e. the number of columns in $\bar{\mathbf{Q}}_k$ in Algorithm 1, are determined by the method in Sect. 3.2. As will be seen in Examples 1 and 3, this method can significantly improve the performance of Algorithm 1.

The data sets used range from toy data (Example 1), real-world data commonly used for testing classification algorithms (Example 2), to the data from an EEG based BCI experiment.

Additionally, all SVM classifications in the iterations of Algorithm 1 are performed by LIBSVM (Chang and Lin).

Example 1 We demonstrate the validity and convergence of our semi-supervised SVM algorithm with parameter setting and model selection through 20 independent simulations. The feature used in this example is feature FD2 defined in (15).

In each simulation, we first randomly generate a data set. Each data sample is a 16 dimensional vector. The vectors in the first class are generated by a 16 dimensional Gaussian distribution with identity covariance matrix and 0 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, 1.5], 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 15 vectors with known labels, while the test set contains 485 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 retraining. There are 600 samples in total, of which 200 samples belong to the first class while the other 400 samples belong to the second class.

We apply Algorithm 1 to the above data set for feature re-extraction and 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 search a combination C and n from 80 combinations using our method in Sect. 3.2, where $C \in \{0.2, 0.4, 0.6, 0.8, 1\}$, $n \in \{1, 2, \ldots, 16\}$. For the kth iteration of the jth run ($k = 1, \ldots, 10$, $j = 1, \ldots, 20$), we obtain two classification accuracy rates for the test data set and the independent test data set. We denote these classification accuracy rates as $\text{rate}_t^{(1)}(C_0, n_0, k, j)$ and $\text{rate}_{in}^{(1)}(C_0, n_0, k, j)$ respectively, where C_0 and n_0 are selected parameters for the jth run, the superscript 1 refers to Algorithm 1 (the selected C_0 and n_0 in the first 5 runs can be seen in the first column of Table 2 given at the end of this example).

We first compare the accuracy rates $\text{rate}_t^{(1)}(C_0, n_0, 1, j)$ and $\text{rate}_t^{(1)}(C_0, n_0, 10, j)$ by T-test ($j = 1, \ldots, 20$), which are obtained in the 1st iteration and the 10th iteration respectively. The p value is 0.0002. We also compare $\text{rate}_{in}^{(1)}(C_0, n_0, 1, j)$ and $\text{rate}_{in}^{(1)}(C_0, n_0, 10, j)$. The p value is 0.0003. This shows that the iterations of Algorithm 1 can improve classification accuracy significantly.

For further comparison, we apply two alternate algorithms, Algorithm 2 and Algorithm 3, on the same data set for 20 runs. Algorithm 2 is obtained by omitting feature re-extraction from Algorithm 1. Algorithm 3 is a self-training EM algorithm with a naive Bayes classifier. We first perform FDA to extract features, then use the two alternate algorithms for classification. Note that the parameters C and n are the same as in Algorithm 1, and the features are not updated during the iterations of Algorithms 2 and 3. Similarly as above, we obtain the prediction accuracy rates $\text{rate}_t^{(q)}(C_0, n_0, k, j)$ and $\text{rate}_{in}^{(q)}(C_0, n_0, k, j)$ for test data set and independent test data set respectively, where $q = 2, 3$ implies Algorithms 2 and 3.

We now perform the comparison of Algorithms 1 and 2. Using T-test, we compare $\text{rate}_t^{(1)}(C_0, n_0, 10, j)$ with $\text{rate}_t^{(2)}(C_0, n_0, 10, j)$ for test data set, and compare $\text{rate}_{in}^{(1)}(C_0, n_0, 10, j)$ and $\text{rate}_{in}^{(2)}(C_0, n_0, 10, j)$ for independent test data set. The corresponding P values of T-test are 0.0029 and 0.0023 respectively. Similarly, we perform the comparison of Algorithms 1 and 3. The P values of T-test are almost zero. The comparison of the results obtained by Algorithm 1 and the other two algorithms further illustrates that feature re-extraction indeed helps improve the classification performance.

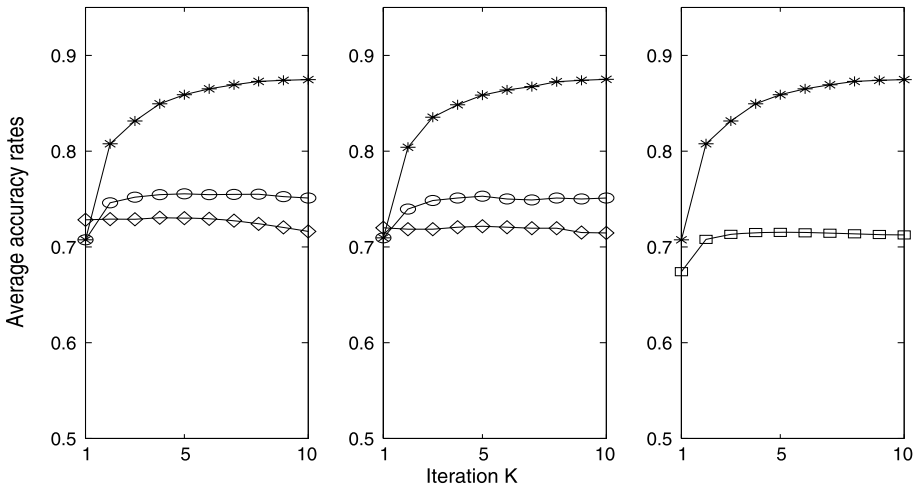

Fig. 2 Analysis results in Example 1. The *left and middle subplots* show the curves of prediction accuracy rates averaged over 20 runs in each iteration, for the test data sets and the independent test data sets respectively. In these two subplots, the curves with stars are obtained by Algorithm 1 with feature re-extraction, the *curves with circles* are obtained by Algorithm 2 without feature re-extraction, while *the curves with diamonds* are obtained by Algorithm 3, a self-training EM without feature re-extraction. The *right subplot* shows two curves of average prediction accuracy rates from Algorithm 1. The *curve with stars* is from parameter setting and model selection, while the other one with squares is not

We average $\text{rate}_t^{(q)}(C_0, n_0, k, j)$ and $\text{rate}_{in}^{(q)}(C_0, n_0, k, j)$ over 20 runs, and obtain the average prediction accuracy rates $\bar{\text{rate}}_t^{(q)}(k)$ and $\bar{\text{rate}}_{in}^{(q)}(k)$ respectively ($q = 1, 2, 3$),

$$\bar{\text{rate}}_t^{(q)}(k) = \frac{1}{20}\sum_{j=1}^{20} \text{rate}_t^{(q)}(C_0, n_0, k, j), \tag{20}$$

$$\bar{\text{rate}}_{in}^{(q)}(k) = \frac{1}{20}\sum_{j=1}^{20} \text{rate}_{in}^{(q)}(C_0, n_0, k, j). \tag{21}$$

The average accuracy rates $\bar{\text{rate}}_t^{(q)}(k)$ and $\bar{\text{rate}}_{in}^{(q)}(k)$ ($q = 1, 2, 3$, representing Algorithms 1, 2, 3) are shown in the left and middle subplot of Fig. 2 respectively. From the these two subplots of Fig. 2, we can see Algorithm 1 performs FDA feature re-extraction and classification effectively when training data set is small. It can also be seen that the performance of Algorithm 3 is relatively poor. Since the training data set is insufficient, the features directly extracted by FDA are not reliable. This leads to relatively poor classification results in Algorithm 3. Based on this comparison, we deduce that feature re-extraction plays an important role in Algorithm 1 (also refer to Fig. 1).

In order to demonstrate the validity of our method for parameter setting and model selection, for each combination (C, n), we also calculate the prediction accuracy rates $\text{rate}_t^{(1)}(C, n, k, j)$ for test data set. We average $\text{rate}_t^{(1)}(C, n, k, j)$ over all 80 possible combinations of C and n, and over 20 runs,

$$\tilde{\text{rate}}(k) = \frac{1}{1600}\sum_{i=1}^{5}\sum_{q=1}^{16}\sum_{j=1}^{20} \text{rate}_t^{(1)}(C_i, n_q, k, j). \tag{22}$$

Table 2 Terminating criterion $r(C_0, n_0, k)$ of Algorithm 1 for the first 5 runs in Example 1

	Iterations	2	3	4	5	6	7	8	9	10
Run 1	$r(0.4, 10, k)$	0.286	0.085	0.047	0.029	0.016	0.029	0.022	0.008	0.002
Run 2	$r(0.8, 9, k)$	0.315	0.087	0.074	0.029	0.027	0.014	0.010	0.004	0.004
Run 3	$r(0.2, 11, k)$	0.166	0.051	0.029	0.010	0.002	0	0	0	0
Run 4	$r(0.2, 9, k)$	0.325	0.056	0.016	0.008	0.004	0.002	0.002	0.002	0.002
Run 5	$r(1, 9, k)$	0.437	0.099	0.039	0.014	0.008	0.002	0.002	0.002	0.002

Table 3 The Rayleigh coefficient $R(C_0, n_0, k)$ for the first 5 runs in Example 1

	Iterations	2	3	4	5	6	7	8	9	10
Run 1	$R(0.4, 10, k)$	0.832	2.607	3.848	4.716	5.656	6.681	9.811	13.795	14.056
Run 2	$R(0.8, 9, k)$	0.909	2.035	2.730	3.824	4.450	5.163	5.348	5.826	5.972
Run 3	$R(0.2, 11, k)$	1.562	4.824	7.996	10.839	11.061	11.955	11.955	11.955	11.955
Run 4	$R(0.2, 9, k)$	0.937	4.337	5.132	5.561	5.488	5.747	5.488	5.747	5.488
Run 5	$R(1, 9, k)$	0.634	3.324	4.798	7.118	7.260	8.522	8.953	9.206	9.286

The average prediction accuracy rates $\tilde{\text{rate}}(k)$ reflect the performance of Algorithm 1 without selection of C and n.

The right subplot of Fig. 2 shows two curves of the average prediction accuracy rates $\bar{\text{rate}}(k)$ and $\tilde{\text{rate}}(k)$. The curve with stars depicts $\bar{rate}_t^{(1)}(k)$, which is obtained by Algorithm 1 with selection of C and n. The other one with squares depicts $\tilde{\text{rate}}(k)$, which is obtained by Algorithm 1 without selection of C and n. From this subplot, we can see parameter setting and 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 20 runs, we obtain C_0 and n_0 through selection of C and n. For the selected C_0 and n_0, we also obtain the terminating criterion $r(C_0, n_0, k)$ in (17) and the Rayleigh coefficient $R(C_0, n_0, k)$ in (18) by running Algorithm 1. In the following Tables 2 and 3, we list $r(C_0, n_0, k)$ and $R(C_0, n_0, k)$ of the first 5 runs. From the two tables, we can see that Algorithm 1 converges fast, and that the Rayleigh coefficients increase in the iterations of Algorithm 1. Note that in each run, only 9 values of $r(C_0, n_0, k)$ are available among the 10 iterations (see definition of $r(k)$ in Algorithm 1). Additionally, the Rayleigh coefficient $R(C_0, n_0, 1)$ (obtained in the first iteration) is not listed in Table 3. This is because Rayleigh coefficient maximization is based on a small training data set in the first iteration of Algorithm 1. In this case, the matrix $\mathbf{S}_N$ in (13) may be close to singular. The Rayleigh coefficient obtained in the first iteration may be too large to be compared with those obtained in subsequent iterations.

Example 2 In this example, we test Algorithm 1 on 5 real-world data sets "Cancer," "Isonosphere," "diabetes," "Dimdata" and "german.numer_scale" (Newman et al. 1998). The dimensions and numbers of sample vectors of the four real-world data sets are listed in the second and third columns of Table 4 respectively.

The feature used here is feature FD1 defined in (14). For each data set, we perform a 5-fold cross-validation. In each fold, the data set is divided into three parts. The first part is called the initial training data set. The second part is the test data set which is used in

Table 4 Analysis results (accuracy rates % and p values) for four data sets

Data	Dimension	Size	Size of training set	Alg. 1	SVM	P value	EM	P values	SVMlight	P value
Cancer	10	680	10	94.8	70.4	0.004	95.4	0.56	92.8	0.37
Isono.	34	350	50	86.6	83.5	0.26	89.1	0.77	73.3	0.03
Dia	8	768	40	80.5	73.4	0.002	68	0.000	64.3	0.000
Dim	14	2200	20	96.1	81.0	0.000	67.3	0.000	69.8	0.000
German	24	1000	10	99	77.8	0.002	91.6	0.017	79	0.002

retraining. The third part is the independent test set for further validation (this data set is not used for retraining). For all four data sets, the sizes of the initial training data sets are shown in the fourth column of Table 4. Under the condition that our algorithm works, we choose the size of the initial training data sets as small as possible (also much smaller than the sizes of training data sets given by Newman et al. 1998). The ratio of the sizes of test data set and independent test set is 4 : 1. We apply Algorithm 1 to each of the 5 real-world data sets. We obtain the accuracy rates for the test set and the independent test set in each fold of the cross-validation. Thus there are a total of 10 accuracy rates, which are then averaged. The average predication accuracy rate for each data set is listed in the fifth column in Table 4.

For the purpose of comparison, we first use a standard SVM algorithm to replace Algorithm 1 and perform a similar analysis for each of the 5 data sets. 10 accuracy rates are also obtained and then averaged. The average predication accuracy rate is listed in the sixth column in Table 4. Using T-test, we further compare the 10 accuracy rates obtained by Algorithm 1 and the 10 accuracy rates obtained by the standard SVM algorithm. The corresponding p value is listed in the seventh column. Next, we compare Algorithm 1 with two semi-supervised algorithms similarly as above, one is self-training EM, the other is a transductive SVM (SVMlight, http://svmlight.joachims.org/). The average predication accuracy rates obtained by these two algorithms are listed in the eighth and tenth columns respectively, while the corresponding p values are shown in the ninth and eleventh columns in Table 4. Note that when using above standard SVM or SVMlight for classification, we first perform the model selection using leave-one-out (an extreme case of cross validation) on training data set. We use leave-one-out rather than cross-validation because the training data set is small.

From statistical tests, the accuracy performance of Algorithm 1 is significantly better (p values < 0.05) than that of standard SVM algorithm and SVMlight algorithm for 4 of the 5 data sets, and it is significantly better than that of EM for 3 of the 5 data sets. For the first data set "Cancer," Algorithm 1 does not show obvious advantage compared with EM algorithm and SVmlight algorithm. For the second data set "Isonosphere," Algorithm 1 does not show obvious advantage compared with standard SVM algorithm and EM algorithm. If the quality of raw data is high, then feature re-extraction (even feature extraction) will not add too much value.

Additionally, our algorithm is faster than SVMlight algorithm when the data set is large. For instance, when SVMlight algorithm is applied to "Dimdata" data set, the time taken is 5.28×10^5 s in our computer (Intel Pentium 2 GHz processor and 2 GB RAM). This is because too many (1826) test data samples are used to train the classifier. The time taken for Algorithm 1 is around 48s even when all 1826 test data samples are used in retraining the classifier.

Example 3 In this example, we evaluate Algorithm 1 with CSP feature re-extraction using the following data set: Data set IVa in BCI Competition 2005, provided by K.R. Muller, B. Blankertz and G. Curio. This data set is provided for researchers to evaluate their algorithm performance when only a small amount of labeled training data is available. The details of the experiment and data set can be found from the website: http://ida.first.fraunhofer.de /projects/bci/competition. In each trial of the experiment, visual cues indicated for 3.5 s one of the following 2 motor imageries, which the subject should perform: (R) right hand, (F) right foot. The presentation of target cues were intermitted by periods of random length, 1.75 to 2.25 s, in which the subject could relax. In this paper, we present our analysis results for the first 2 subjects "aa" and "al," as the results for the other three subjects are similar.

Before the feature extraction and classification, proper preprocessing is necessary to ensure good performance. In this paper, the preprocessing methods include Common Average Reference (CAR) for spatial filtering, frequency filtering, in which a IIR type filter is used to filter the data in mu band (12–14 Hz). For each trial, we use data of length 3.5 seconds for analysis. During this period, the cue was visible on the screen. The selected frequency bands for the two subjects in our study are 12–14 Hz.

As an example, we first describe our analysis procedure and results for subject "aa." The same procedure applies to subject "al." We use 200 trials to test our algorithm. We divide the 200 trials into 8 folds according to their sequential order for 8-fold cross-validation. One of the 8 folds are used as the initial training set (25 trials) and the rest of the 7 folds are used as the test set (175 trials). During each iteration of Algorithm 1, we extract CSP features from initial training data set and test set, and predict the labels of the test set. The prediction accuracy rates $\mathrm{rate}(C_0, n_0, k, j)$ are then calculated, where k represents the kth iteration, $j (= 1, \ldots, 8)$ represents the jth fold which is used as the initial training set, and C_0 and n_0 are selected parameters. Note that we search a combination of C and n on a grid with $C \in \{0.2, 0.4, 0.6, 0.8, 1\}$ and $n \in \{6, 8, \ldots, 20\}$. For the purpose of comparison, we also calculate the prediction accuracy rates $\mathrm{rate}(C, n, k, j)$ for each (C, n).

We average $\mathrm{rate}(C_0, n_0, k, j)$ over all 8 folds and obtain the average prediction accuracy rates $\bar{\mathrm{rate}}(k)$ for each iteration as follows,

$$\bar{\mathrm{rate}}(k) = \frac{1}{8} \sum_{j=1}^{8} \mathrm{rate}(C_0, n_0, k, j). \tag{23}$$

Furthermore, we average $\mathrm{rate}(C, n, k, j)$ over all 40 combinations of C and n, and 8 folds as follows,

$$\tilde{\mathrm{rate}}(k) = \frac{1}{320} \sum_{i=1}^{5} \sum_{q=1}^{8} \sum_{j=1}^{8} \mathrm{rate}(C_i, n_q, k, j). \tag{24}$$

In each iteration of Algorithm 1, we also calculate the average Rayleigh coefficient,

$$\bar{R}(k) = \frac{1}{8} \sum_{j=1}^{8} R(C_0, n_0, k, j), \tag{25}$$

where $R(C_0, n_0, k, j)$ is defined as in (18), C_0 and n_0 are selected parameters, k and j represent the kth iteration and the jth fold respectively.

The results are shown in the first row of Fig. 3. In the left subplot, average accuracy $\bar{\mathrm{rate}}(k)$ and $\tilde{\mathrm{rate}}(k)$ are depicted. The obvious difference between $\bar{\mathrm{rate}}(k)$ and $\tilde{\mathrm{rate}}(k)$ demonstrates the validity of our method for parameter setting and model selection. In the right

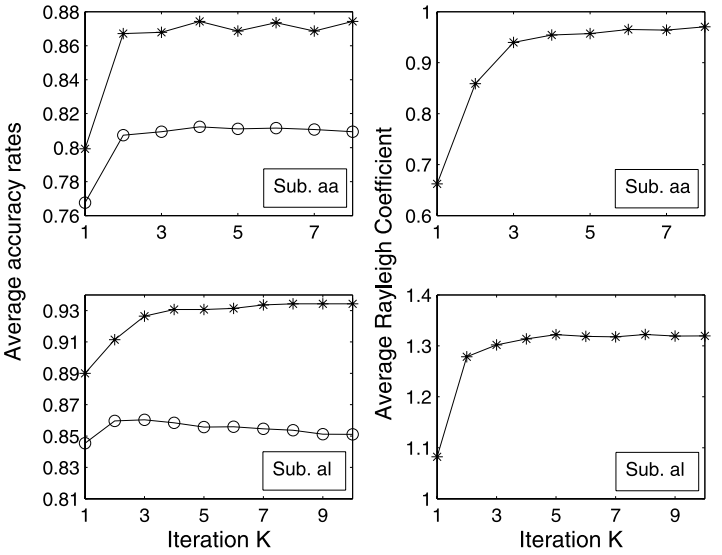

Fig. 3 Data analysis results in Example 3. The *1st and 2nd rows* are for subjects "aa" and "al" respectively. The *first column* shows the curves of average prediction accuracy rates for the test set. The *curves with stars* are from Algorithm 1 with parameter selection; while the *curves with circles* are from Algorithm 1 without parameter selection. The *2nd column* depicts the curves of the Rayleigh coefficient averaged over 8 folds in cross-validation

subplot, the curve of average Rayleigh coefficients $\bar{R}(k)$ obtained in the cross-validation is shown. This curve demonstrates the convergence of our algorithm.

The results for subject "al" are shown in the second row of Fig. 3. These results are similar to the results of subject "aa."

From Example 3, we can see that semi-supervised learning algorithms can be used to reduce the user's training effort in BCI systems. Another promising application of semi-supervised learning in BCI systems is in online adaptation, where huge amounts of unlabeled data is available for retraining/adjusting the system parameters.

From these three examples of this paper, three main advantages of our approach can be seen: (i) feature re-extraction in Algorithm 1 can improve the quality of features (also see Fig. 1) and the classification accuracy; (ii) Algorithm 1 converges fast (generally in less than 10 iterations); (iii) our method for parameter setting and model selection is effective even when training data set is extremely small and cross-validation method for parameter setting and model selection may not work.

5 Conclusions

In this paper, we first presented feature extraction under the framework of Rayleigh coefficient maximization. Three features including the CSP feature were discussed. All three features can be estimated with a transformation matrix determined by maximizing the Rayleigh coefficients. To solve the small training set problem, we proposed an iterative semi-supervised SVM algorithm with feature re-extraction. Using this algorithm, we can extract reliable features and achieve good classification performance in cases when the training data set is small. Furthermore, we also proposed a semi-supervised learning-based method using

the Rayleigh coefficient for parameter setting and model selection. This method utilizes both training data and test data and can be suitable for small training data set. Finally, we have demonstrated the validity of our approach using three examples of data analysis.

Future studies include the extension of our algorithm to Kernel-type and multi-class cases to expand its applicability.

Acknowledgements The authors are grateful for the insightful and constructive suggestions from anonymous reviewers and the action editor, Dr. Olivier Chapelle and acknowledge their contribution to this paper. The authors are also grateful for Mr. Chin Zheng Yang for his contribution in the presentation of this paper.

Appendix 1: On the regularization in FDA

In (13), we add a regularization item to extract muti-dimensional features. We now discuss the relationship between the generalized eigenvalues/eigenvectors after regularization and the generalized eigenvalues/eigenvectors obtained by normal FDA (i.e. $\alpha = 0$ in (13)). We first list a set of generalized eigenvalues and generalized eigenvectors obtained with different α values to show the relationship. Next, we illustrate the principle for setting the value of α.

A: We generate two 4×20 matrices **A** and **B** using two different Gaussian distributions (with means 0 and 0.5 respectively, and unit variance). **A** and **B** represent two classes of data, of which one column is a data sample. For a given α, we calculate $\mathbf{S}_I$ and $\mathbf{S}_N$ in (13). Through joint diagonalizing $\mathbf{S}_I$ and $\mathbf{S}_N$, we obtain 4 generalized eigenvalues $\tilde{\lambda}_i$ and 4 generalized eigenvectors $\mathbf{q}_i$ $(i = 1, \ldots, 4)$. In Table 5 and Table 6, we list these generalized eigenvalues and generalized eigenvectors calculated with 5 different α values. Note that $\alpha = 0$ implies normal FDA.

From Table 5, when α is close to zero, the generalized eigenvalues (highlighted in bold) are close to those obtained by normal FDA. The explanation for this is as follows: We take αI in (13) as a disturbance. From the joint diagonalization procedure in Sect. 2.1, the generalized eigenvalues will tend to the normal FDA generalized eigenvalues when αI tends to a zero matrix.

In Table 6, similar phenomena can be found only for the first generalized eigenvector $\mathbf{w}_1$ (highlighted in bold), which corresponds to the biggest generalized eigenvalues. For the other three generalized eigenvectors $\mathbf{w}_2$, $\mathbf{w}_3$ and $\mathbf{w}_4$, no clear relationship can be seen between the case with regularization and the case with normal FDA. The main reason is that the three generalized eigenvectors corresponding to the zero eigenvalue in normal FDA cannot be fixed during joint diagonalization procedure (any nonzero linear combination of the three generalized eigenvectors is still a generalized eigenvector).

B. In Example 1 of Sect. 4, we have 20 runs of simulations where α was fixed to 0.05. We now consider 10 different values of α and run 20 simulations for each α as in Example 1. The obtained predication accuracy rates are denoted as rate(α, j), where

Table 5 Generalized eigenvalues $\tilde{\lambda}_i$ calculated with 5 different α values

α	0	0.01	0.5	1	2
$\tilde{\lambda}_1$	**0.4822**	**0.4877**	0.7777	1.1457	2.0737
$\tilde{\lambda}_2$	**−0.0000**	**0.0096**	0.4641	0.8681	1.5110
$\tilde{\lambda}_3$	**−0.0000**	**0.0055**	0.2696	0.5320	1.0401
$\tilde{\lambda}_4$	**−0.0000**	**0.0038**	0.1874	0.3695	0.7236

Table 6 Generalized eigenvectors $\mathbf{w}_i$ calculated with 5 different α values

α	0	0.01	0.5	1	2
$\mathbf{w}_1$	$\begin{bmatrix} \mathbf{-0.1334} \\ \mathbf{-0.3526} \\ \mathbf{-0.4485} \\ \mathbf{-0.4524} \end{bmatrix}$	$\begin{bmatrix} \mathbf{-0.1308} \\ \mathbf{-0.3521} \\ \mathbf{-0.4498} \\ \mathbf{-0.4537} \end{bmatrix}$	$\begin{bmatrix} 0.0259 \\ -0.2644 \\ -0.5259 \\ -0.5478 \end{bmatrix}$	$\begin{bmatrix} 0.2330 \\ -0.0221 \\ -0.5737 \\ -0.6684 \end{bmatrix}$	$\begin{bmatrix} 0.3878 \\ 0.2559 \\ -0.5193 \\ -0.6987 \end{bmatrix}$
$\mathbf{w}_2$	$\begin{bmatrix} -0.3957 \\ 0.4931 \\ -0.0876 \\ 0.0262 \end{bmatrix}$	$\begin{bmatrix} 0.4709 \\ 0.5649 \\ -0.4498 \\ -0.5679 \end{bmatrix}$	$\begin{bmatrix} 0.4594 \\ 0.6467 \\ -0.1883 \\ -0.4697 \end{bmatrix}$	$\begin{bmatrix} 0.3736 \\ 0.7211 \\ 0.0445 \\ -0.2504 \end{bmatrix}$	$\begin{bmatrix} 0.1704 \\ 0.6967 \\ 0.3089 \\ -0.0169 \end{bmatrix}$
$\mathbf{w}_3$	$\begin{bmatrix} -0.3398 \\ 0.2784 \\ 0.4535 \\ -0.3875 \end{bmatrix}$	$\begin{bmatrix} 0.3564 \\ -0.2049 \\ 0.4982 \\ 0.3596 \end{bmatrix}$	$\begin{bmatrix} 0.3631 \\ -0.1400 \\ -0.4764 \\ 0.3915 \end{bmatrix}$	$\begin{bmatrix} 0.3639 \\ -0.0766 \\ -0.4521 \\ 0.4191 \end{bmatrix}$	$\begin{bmatrix} 0.3532 \\ 0.0270 \\ -0.4067 \\ 0.4570 \end{bmatrix}$
$\mathbf{w}_4$	$\begin{bmatrix} 0.5058 \\ 0.5332 \\ -0.3249 \\ -0.5556 \end{bmatrix}$	$\begin{bmatrix} 0.3476 \\ -0.3864 \\ 0.2442 \\ -0.2268 \end{bmatrix}$	$\begin{bmatrix} 0.3784 \\ -0.3558 \\ 0.2530 \\ -0.1873 \end{bmatrix}$	$\begin{bmatrix} 0.4006 \\ -0.3287 \\ 0.2600 \\ -0.1547 \end{bmatrix}$	$\begin{bmatrix} 0.4272 \\ -0.2874 \\ 0.2692 \\ -0.1083 \end{bmatrix}$

Table 7 Average classification accuracy rates (%) obtained with 10 different α values

α	0	0.0001	0.001	0.01	0.02	0.05	0.1	0.5	1	2
$\bar{\text{rate}}(\alpha)$	82.04	89.74	90.03	89.32	89.04	87.50	84.21	71.58	71.42	71.52

$\alpha \in \{0, 0.0001, 0.001, 0.01, 0.02, 0.05, 0.1, 0.5, 1, 2\}$, $j = 1, \ldots, 20$. Denote the mean of $\text{rate}(\alpha, 1:20)$ as $\bar{\text{rate}}(\alpha)$, which is shown in Table 7.

From Table 7 and Part A of this appendix, α should be set small such that the largest generalized eigenvalue and corresponding generalized eigenvector are close to those obtained in normal FDA. In this way, the advantage of normal FDA will be kept in our algorithm. Additionally, when α is sufficiently small (e.g. $\alpha = 0.0001, 0.001, 0.01, 0.02$), the prediction accuracy rates obtained with different α values do not have a significant difference.

However, from Table 7, we can find that the result obtained with $\alpha = 0$ is significantly less than those obtained with small α values. This is because when $\alpha = 0$, we perform normal FDA feature extraction, and the extracted features are one dimensional.

Appendix 2: A simulation on Rayleigh coefficient and separability of data

In this appendix, we will show that bigger Rayleigh coefficient implies higher separability of data through a simulation.

First we generate 3 data sets denoted as $\{\mathbf{x}_1(i) \in R^8, i = 1, \ldots, 200\}$, $\{\mathbf{x}_2(i) \in R^8, i = 1, \ldots, 200\}$, $\{\mathbf{x}_3(i) \in R^8, i = 1, \ldots, 200\}$, each of which has two classes. The data vectors $\mathbf{x}_1(i)$, $\mathbf{x}_2(i)$, $\mathbf{x}_3(i)$, $i = 1, \ldots, 100$, are generated from the normal distribution $N(\mathbf{0}, \mathbf{I})$, where $\mathbf{0}$ is a 8 dimensional zero mean vector, $\mathbf{I} \in R^{8\times 8}$ is a unit covariance matrix. The

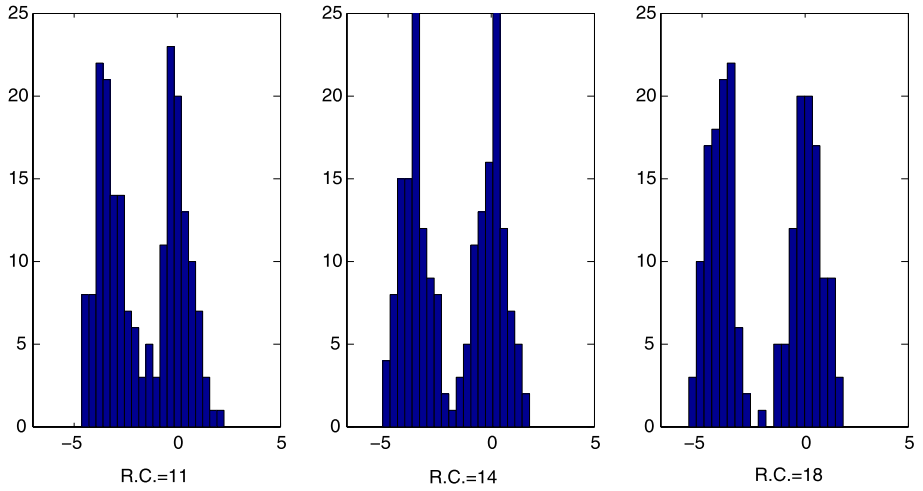

Fig. 4 Three distribution histograms for three feature sets with different Rayleigh coefficients (R.C.) show different degrees of separability

data vectors $\mathbf{x}_1(i)$, $i = 101, \ldots, 200$ are generated from the normal distribution $N(\mathbf{m}_1, \mathbf{I})$, where $\mathbf{m}_1 = [0.67, 2.11, 1.57, 2.80, 2.14, 0.68, 1.35, 0.52]^T$. The data vectors $\mathbf{x}_2(i)$ and $\mathbf{x}_3(i)$, $i = 101, \ldots, 200$, are generated from the normal distribution $N(\mathbf{m}_2, \mathbf{I})$, where $\mathbf{m}_2 = [2.91, 1.07, 0.15, 2.27, 2.68, 0.86, 0.75, 2.80]^T$. Note that $\mathbf{m}_1$ and $\mathbf{m}_2$ are also arbitrarily selected.

For the above 3 data sets, we define the Rayleigh coefficient according to (1) and (12). Through Rayleigh coefficient maximization, we obtain 3 generalized eigenvectors denoted as $\mathbf{w}_1$, $\mathbf{w}_2$ and $\mathbf{w}_3$ for the 3 data sets respectively. Using these eigenvectors, we can obtain 3 new feature sets: $\{\mathbf{w}_1^T\mathbf{x}_1(i) \in R^8, i = 1, \ldots, 200\}$, $\{\mathbf{w}_2^T\mathbf{x}_2(i) \in R^8, i = 1, \ldots, 200\}$, $\{\mathbf{w}_3^T\mathbf{x}_3(i) \in R^8, i = 1, \ldots, 200\}$. For the 3 feature sets, the three subplots of Fig. 4 show their distribution histograms and the Rayleigh coefficients (denoted as R.C.) respectively. From Fig. 4, we can see that a higher Rayleigh coefficient implies higher separability of a data set.

References

Bennett, K. P., & Demiriz, A. (1998). Semi-supervised support vector machines. In M. S. Kearns, S. A. Solla, & D. A. Cohn (Eds.), *Advances in neural information processing systems* (Vol. 12, pp. 368–374). Cambridge: MIT Press.

Blanchard, G., & Blankertz, B. (2004). BCI competition 2003-data set IIa: spatial patterns of self-controlled brain rhythm modulations. *IEEE Transactions on Biomedical Engineering*, *51*(6), 1062–1066.

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the conference on computational learning theory* (pp. 92–100).

Brefeld, U., & Scheffer, T. (2004). Co-EM support vector learning. In *Proceedings of the 21st international conference on machine learning*, Canada.

Chang, C. C., & Lin, C. J. (2003). 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 *Proceedings of the tenth international workshop on artificial intelligence and statistics* (pp. 57–64). Barbados.

Chapelle, O., Scholkopf, B., & Zien, A. (2006). *Semi-supervised learning*. Cambridge: MIT Press.

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

Fung, G., & Mangasarian, O. (2001). Semi-supervised support vector machines for unlabeled data classification. In *Optimization methods and software* (pp. 1–14). Boston: Kluwer Academic.

Grandvalet, Y., & Bengio, Y. (2004). Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems* (Vol. 16). Cambridge: MIT Press.

Kong, H., Teoh, E., Wang, J., & Kambhamettu, C. 2005. Generalized 2D fisher discriminant analysis. In *Proceedings of the 16th British machine vision conference*, Oxford, UK.

Joachims, T. (1999). Transductive Inference for text classification using support vector machines. In *Proceedings of the international conference on machine learning (ICML)*.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the ACM conference on knowledge discovery and data mining (KDD)*. New York: ACM.

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 Proceedings of the 7th European conference on principles and practice of knowledge discovery in databases (pp. 266–277), Cavtat-Dubrovnik, Croatia.

Krishnapuram, B., Hartemink, A. J., Carin, L., & Figueiredo, M. A. T. (2004). A Bayesian approach to joint feature selection and classifier design. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(9), 1105–1111.

Li, Y., & Guan, C. (2006). An extended EM algorithm for joint feature extraction and classification in brain computer interfaces. *Neural Computation*, *18*, 2730–2761.

Mika, S. (2002). *Kernel Fisher discriminants*, PhD thesis.

Mika, S., Ratsch, G., Weston, Scholkopf, B., & Mller, K. R. (1999). Fisher discriminant analysis with kernels. In Y.-H. Hu, J. Larsen, E. Wilson, & S. Douglas (Eds.) *Neural networks for signal processing IX* (pp. 41–48). New York: IEEE.

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 *Proceedings of 9th international conference 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. *Information Processing & Management*, *40*(3), 421–439.

Pfutscheller, G., Neuper, C., Flotzinger, D., & Pregenzer, M. (1997). EEG-based discrimination between imagination of right and left hand movement? *Electroencephalography and Clinical Neurophysiology*, *103*, 642–651.

Ramoser, H., Muller-Gerking, J., & Pfurtscheller, G. (2000). Optimal spatial filtering of single trial EEG during imagined hand movement. *IEEE Transactions on Rehabilitation Engineering*, *8*(4), 441–446.

Tong, S. & Koller, D. (2001). Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, *2*, 45–66.

Vapnik, V. (1998). *Statistical learning theory*. Berlin: Springer.

Weston, J., Perez-Cruz, F., Bousquet, O., Chapelle, O., Elisseeff, A., & Scholkopf, B. (2002). Feature selection and transduction for prediction of molecular bioactivity for drug design. *Bioinformatics*, *1*(1), 1–8.

Wolpaw, J. R., Birbaumer, N., McFarland, D. J., Pfurtscheller, G., & Vaughan, T. M. (2002). Brain-computer interfaces for communication and control. *Clinical Neurophysiology*, *113*, 767–791.

Zhou, D., Bousquet, O., Lal, T. N., Weston, J., & Scholkopf, B. (2003). Learning with local and global consistency. In *Advances in neural information processing systems* (Vol. 15). Cambridge: MIT Press.

Zhou, D., Scholkopf, B., & Hofmannz, T. (2004). Semi-supervised learning on directed graphs. In *Advances in neural information processing systems* (Vol. 16). Cambridge: MIT Press.

Zhu, X., Lafferty, J., & Ghahramani, Z. (2003). Combining active learning and semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the ICML-2003 workshop on the continuum from labeled to unlabeled data*, Washington DC.