Information Sciences 364–365 (2016) 156–166

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A semi-supervised random vector functional-link network based on the transductive framework

Simone Scardapane*, Danilo Comminiello, Michele Scarpiniti, Aurelio Uncini

Department of Information Engineering, Electronics and Telecommunications (DIET), "Sapienza" University of Rome, Via Eudossiana 18, Rome 00184, Italy

ARTICLE INFO

Article history:
Received 15 May 2015
Revised 14 July 2015
Accepted 29 July 2015
Available online 22 August 2015

Keywords:
Random vector functional-link
Semi-supervised learning
Transductive learning
Manifold regularization

ABSTRACT

Semi-supervised learning (SSL) is the problem of learning a function with only a partially labeled training set. It has considerable practical interest in applications where labeled data is costly to obtain, while unlabeled data is abundant. One approach to SSL in the case of binary classification is inspired by work on transductive learning (TL) by Vapnik. It has been applied prevalently using support vector machines (SVM) as the base learning algorithm, giving rise to the so-called transductive SVM (TR-SVM). The resulting optimization problem, however, is highly non-convex and complex to solve. In this paper, we propose an alternative semi-supervised training algorithm based on the TL theory, namely semi-supervised random vector functional-link (RVFL) network, which is able to obtain state-of-the-art performance, while resulting in a standard convex optimization problem. In particular we show that, thanks to the characteristics of RVFLs networks, the resulting optimization problem can be safely approximated with a standard quadratic programming problem solvable in polynomial time. A wide range of experiments validate our proposal. As a comparison, we also propose a semi-supervised algorithm for RVFLs based on the theory of manifold regularization.

1. Introduction

Supervised learning (SL) is the task of inferring a function starting from a set of labeled examples of it [19]. Semi-supervised learning (SSL) is an extension of SL, in which the user is provided with an additional set of *unlabeled* data [12]. The central problem of SSL is to incorporate this additional set into the learning problem, so as to maximize accuracy over unseen patterns. Under this respect, SSL algorithms differentiate themselves mostly on the assumptions that are made to this end [19]. SSL is useful whenever labeled data is scarce (or costly to obtain), while unlabeled data is common and easy to collect. Examples abound in real world applications, including bioinformatics [23], image processing [32], computer vision [18] and text classification [22].

One common approach to SSL is manifold regularization (MR) [5,17,28,38]. In MR, it is assumed that the input data, while being observed in an Euclidean space $\mathbb{R}^d$ (called the ambient space), practically lies on an embedded manifold $\mathcal{M}$, called the intrinsic space, whose dimension might be much lower than d. Since the geometry of $\mathcal{M}$ only depends on the input data, it can be estimated using both labeled and unlabeled data together. Originally, Belkin and Niyogi [4] proposed this idea for dimensionality reduction in the ambient space. Later, it was shown how information on $\mathcal{M}$ can be used to incorporate an efficient regularization term in the learning problem [5], forcing the output of the resulting function to be close whenever two points are similar in their

* Corresponding author. Tel.: +39 06 44585495; fax: +39 06 4873300.
E-mail addresses: simone.scardapane@uniroma1.it, simonescardapane@gmail.com (S. Scardapane), danilo.comminiello@uniroma1.it (D. Comminiello), michele.scarpiniti@uniroma1.it (M. Scarpiniti), aurelio.uncini@uniroma1.it (A. Uncini).

http://dx.doi.org/10.1016/j.ins.2015.07.060
0020-0255/

intrinsic space. From this general framework, it is possible to derive a wide range of MR kernel-based algorithms, including the Laplacian Support Vector Machine (LAP-SVM) [28], and the Laplacian Regularized Least-Square (LAP-RLS) [5]. The main drawback of the MR approach is that, in order to derive the regularization term, it is necessary to estimate a matrix connected to $\mathcal{M}$, called the Laplacian, which in turn depends on the pairwise similarities between each pair of points. This is not straightforward in general, since its computation requires setting a number of free parameters, each of which heavily influences the resulting classification accuracy (see e.g. [36]). In fact, model selection is a crucial problem in SSL, where labeled data may not be sufficient to obtain a large validation set [31].

In the case of binary classification, an alternative approach to SSL is constituted by the transductive learning (TL) theory introduced by V. Vapnik [39,41]. TL is similar to SSL, in that we are provided with an additional set of unlabeled data. The main difference is that in TL we do not wish to infer a classifier over the full input space, but we only require the unknown labels of the additional dataset. Based on the principle of structural risk minimization, Vapnik proposed a transductive version of the support vector machine (denoted as TR-SVM) [22,39]. In TR-SVM, the unknown labels are introduced as additional variables in the optimization problem, and we minimize the training error over both labeled and unlabeled data. Despite being formulated in the TL framework, TR-SVM can actually be used without modifications for SSL, where it is known as semi-supervised SVM (S^3VM) [6,13].[1] Compared to MR approaches, the TR-SVM only requires an additional free parameter to be specified, but its optimization problem involves both real and binary variables, making it highly non-convex. Over the last years, multiple algorithms based on different relaxations have been proposed for its solution [13,14,27]. It is known, however, that the performance of each of these algorithms is highly domain-dependent, and each of them presents complex implementation details [13].

The basic idea underlying the TR-SVM, i.e. considering the unknown labels as variables in the learning problem, has rarely been applied to models laying outside the realm of the standard SVM formulation (two exceptions being applications to least-square SVM [2], and kernel ridge regression [11]). In this paper, we propose to apply it to a specific class of neural networks, known as random vector functional-link (RVFL) networks [29], or random-weights neural networks (RWNN) [3,35], to derive a novel semi-supervised algorithm for binary classification. An RVFL network is composed of a fixed layer of non-linearities, followed by an adaptable linear layer [20,29,34]. The main interest of RVFL in this context is that, under a least-square training criterion, the solution to its optimization problem can be expressed in closed form. Due to this, we show that the resulting semi-supervised algorithm can be formulated as a standard unconstrained 0-1 optimization problem [25]. While this is an NP-hard problem in general, we show that it can safely be approximated with a box-constrained quadratic (BCQ) problem, solvable in polynomial time [8]. The result is a semi-supervised training algorithm for RVFL networks, that we denote as transductive RVFL (TR-RVFL).[2] Through an extensive set of experimental evaluations, we show that it performs similarly, or better, than a conventional state-of-the-art algorithm based on the MR theory. However, it is easier to tune, and in many situations it can be trained faster than its counterpart.

The main novelty of this paper is the formulation of the optimization problem of TR-RVFL as a BCQ problem, which is solvable in polynomial time. Additionally, we also present an MR-based training algorithm for RVFL networks, and we use it for comparison purposes. In fact, to the best of the authors' knowledge, no work exists related to SSL with the use of RVFL networks. The only related work is [10], where the authors consider the use of functional link networks for semi-supervised clustering.

The rest of the paper is organized as follows. In Section 2, we present the basic concepts used in the rest of the paper, including the formulation of the SSL problem, and the theory of RVFL networks. For completeness, in Section 3, we derive an MR-based training algorithm for RVFL networks, which is used as a comparison in the experimental section. Section 4, which is the main innovative part of the paper, presents the TR-RVFL algorithm. We perform an extensive range of experiments in Section 5, before presenting our conclusive remarks in Section 6.

2. Preliminaries

In this Section, we present the basic theoretical tools used in the rest of the paper. In particular, we provide an overview of SSL in Section 2.1, followed by a derivation of a least-square training algorithm for RVFL networks in the standard supervised case in Section 2.2.

Before this, we provide a brief note on the notation which is adopted. In the following, vectors are denoted with lowercase bold letters, such as $\mathbf{a}$, while matrices are denoted by uppercase bold letters, such as $\mathbf{A}$. All vectors are considered column vectors. The notation A_{ij} denotes the (i,j)th entry of matrix $\mathbf{A}$. $\|\cdot\|_2$ is the standard Euclidean norm. We will also make use of the weighted Euclidean norm, defined for a generic vector $\mathbf{a}$ and matrix $\mathbf{B}$ as:

$$\|\mathbf{a}\|_{\mathbf{B}}^2 = \mathbf{a}^T\mathbf{B}\mathbf{a}\,. \tag{1}$$

Finally, we use $\mathbf{A} \succeq 0$ to denote a positive semi-definite (PSD) matrix, i.e. a matrix for which $\mathbf{x}^T\mathbf{A}\mathbf{x} \geq 0$ for any vector $\mathbf{x}$ of suitable dimensionality.

2.1. Approaches to semi-supervised learning

We consider learning a binary classifier in the supervised and semi-supervised case. Thus, we are interested in finding a classification function $f : \mathbb{R}^d \to \{-1, +1\}$, given a set of examples of this relationship. In the SL setting [19], we have only access

[1] See also [12, Chapters 24 and 25] for additional comments on the link between transduction and SSL.

[2] We use the term 'transductive' to denote its relation to the TL framework. However, as for TR-SVM, the resulting algorithm can be used without modifications in an SSL context.

to a training set of N examples, denoted by $D = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$, where N is the number of examples and $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \{-1, +1\}$, $i = 1, \ldots, N$. Each $\mathbf{x}_i$ is also known as a *pattern*. In the SSL setting [12], instead, we suppose to have available an additional dataset $U = \{\mathbf{x}_i\}_{i=N+1}^{N+Q}$ of Q unlabeled samples. The task is to use the knowledge of U to increase the generalization capability of our inferred model as much as possible.

In the SL case, a large family of learning algorithms can be formulated as the minimization of the following functional form [19]:

$$\min_{f \in \mathcal{H}} J_{\text{SL}}(f) = \lambda \sum_{i=1}^{N} l(y_i, f(\mathbf{x}_i)) + r(f)\,, \tag{2}$$

where $l(y, d)$ is a loss function quantifying the cost incurred for predicting d instead of the true y, $r(f)$ is a term imposing some regularizing constraint on f, $\lambda \in \mathbb{R}$ is a scalar term, called *regularization factor*, and the minimization is performed over a suitable hypothesis space $\mathcal{H}$. In this framework, the question is then how to modify Eq. (2) to take into account the information contained in U. As we discussed in Section 1, two important solutions are those originating from the MR setting [5], and the TL theory [13].

In the former case, we suppose that our patterns, while being observed over $\mathbb{R}^d$, lie in reality over a lower-dimensional manifold $\mathcal{M}$. If this hypothesis is correct, we would like to impose an additional regularization term $r_{\mathcal{M}}(f)$, acting on the intrinsic geometry of $\mathcal{M}$, giving rise to a modified cost function:

$$J_{\text{MR}}(f) = J_{\text{SL}}(f) + \lambda_u r_{\mathcal{M}}(f)\,, \tag{3}$$

where the new term is weighted by a second scalar $\lambda_u > 0$. The geometry of $\mathcal{M}$ is unknown, but it can be estimated from sufficient data, including the patterns contained in U. In particular, starting from labeled and unlabeled data, we can build an approximation of $\mathcal{M}$ in the form of an adjacency graph. This can be represented by a weighting matrix $\mathbf{W} \in \mathbb{R}^{(N+Q)\times(N+Q)}$, such that W_{ij} is the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ on the graph itself. As an example, we can set $W_{ij} = \left\|\mathbf{x}_i - \mathbf{x}_j\right\|_2^2$ whenever $\mathbf{x}_i$ is one of the k-nearest neighbors of $\mathbf{x}_j$, with k set *a priori*, and 0 otherwise. From this, we can consider the following regularization term, forcing $f(\cdot)$ to be equal whenever two points are close in the adjacency graph [5]:

$$r_{\mathcal{M}}(f) = \sum_{i,j=1}^{N+Q} W_{ij}\big(f(\mathbf{x}_i) - f(\mathbf{x}_j)\big)^2\,. \tag{4}$$

It is straightforward to show from Eq. (4) that $r_{\mathcal{M}}(f) = \mathbf{f}^T\mathbf{L}\mathbf{f}$, where $\mathbf{f} = \big[f(\mathbf{x}_1), \ldots, f(\mathbf{x}_{N+Q})\big]^T$. $\mathbf{L}$ is the graph Laplacian associated to our adjacency graph, defined as $\mathbf{L} = \mathbf{D} - \mathbf{W}$, where $\mathbf{D}$ is a diagonal matrix with $D_{ii} = \sum_{i=1}^{N+Q} W_{ij}$. This approach has three main drawbacks. From the usability point of view, we need to define a methodology for constructing the adjacency graph, i.e., we must specify when two points must be considered as neighbors, how to compute the distance between them, whether to normalize the resulting matrix, and so on. Computationally, its construction can be expensive, since it requires to scan every combination of patterns for determining the respective neighbors. Finally, the manifold assumption may not correspond to the reality, and in some situations it may degrade the final result.

A simpler solution is given by the TL theory [39], where the unknown labels of U are added as additional variables in problem (2). Practically, the approach results in the following optimization problem [13]:

$$\min_{f \in \mathcal{H},\, \hat{\mathbf{y}} \in \{-1,+1\}^Q} J_{\text{TR}}(f, \hat{\mathbf{y}}) = J_{\text{SL}}(f) + \lambda_u \sum_{i=N+1}^{N+Q} l(\hat{y}_i, f(\mathbf{x}_i))\,, \tag{5}$$

where $\hat{\mathbf{y}} = \big[\hat{y}_{N+1}, \ldots, \hat{y}_{N+Q}\big]^T$ represents the vector of unknown labels. This approach has a strong theoretical justification in the *structural risk minimization* theory [39], and it can be seen to implement the so-called *cluster assumption*, i.e., similar points in input should possess similar outputs. This hypothesis is conceptually simpler with respect to the manifold assumption, but the resulting optimization problem in Eq. (5) is extremely complex to solve, being in general a non-convex, mixed integer optimization problem. In the case of TR-SVM, several algorithms have been proposed to solve it, based on different relaxing assumptions [13].

2.2. Random-vector functional links

In this section we focus on the SL problem without unlabeled data. An RVFL network is a model of the form [29]:[3]

$$f(\mathbf{x}) = \sum_{i=1}^{B} \beta_i h(\mathbf{x}; \mathbf{w}_i) = \boldsymbol{\beta}^T \mathbf{h}(\mathbf{x})\,, \tag{6}$$

i.e., a linear combination of B non-linear *functional links* $h(\cdot)$, each of which is parameterized by a fixed vector $\mathbf{w}_i$ [15]. To obtain the class of $\mathbf{x}$ from the RVFL output in Eq. (6), we simply take its sign:

$$\text{Class of } \mathbf{x} = \text{sign}(f(\mathbf{x}))\,. \tag{7}$$

[3] The original formulation in [29] considers adaptable connections between the input and the output, which is a straightforward extension.

Once B is chosen, parameters $\mathbf{w}_1, \ldots, \mathbf{w}_B$ are randomly assigned from a fixed probability distribution, and the optimal weights $\boldsymbol{\beta}$ are obtained. Convergence properties of RVFL networks are analyzed in [20]. We define the *output vector* $\mathbf{y} = [y_1, \ldots, y_N]^T$ and the *hidden matrix* $\mathbf{H} \in \mathbb{R}^{N \times B} = \left[\mathbf{h}(\mathbf{x}_1), \ldots, \mathbf{h}(\mathbf{x}_N)\right]^T$. The optimal $\boldsymbol{\beta}$ in the least-square sense is then given by the solution to the following optimization problem:

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^B} J_{\text{SL}}(\boldsymbol{\beta}) = \frac{\lambda}{2} \left\| \mathbf{H}\boldsymbol{\beta} - \mathbf{y} \right\|_2^2 + \frac{1}{2} \left\| \boldsymbol{\beta} \right\|_2^2, \tag{8}$$

where the factor $\frac{1}{2}$ is added for successive simplifications. Solving for $\boldsymbol{\beta}$ we obtain the well-known solution:

$$\boldsymbol{\beta}^* = \left(\mathbf{H}^T\mathbf{H} + \frac{1}{\lambda}\mathbf{I} \right)^{-1} \mathbf{H}^T\mathbf{y}, \tag{9}$$

where $\mathbf{I}$ is the identity matrix of appropriate size. If d is larger than the number of labeled points, we can exploit the algebraic identity $\left(\mathbf{H}^T\mathbf{H} + \frac{1}{\lambda}\mathbf{I}\right)\mathbf{H}^T = \mathbf{H}\left(\mathbf{H}\mathbf{H}^T + \frac{1}{\lambda}\mathbf{I}\right)$ to obtain an alternative, computationally cheaper formulation:

$$\boldsymbol{\beta}^* = \mathbf{H}^T \left(\mathbf{H}\mathbf{H}^T + \frac{1}{\lambda}\mathbf{I} \right)^{-1} \mathbf{y}. \tag{10}$$

3. Manifold regularized RVFL networks

For completeness, in this section we derive an MR-based semi-supervised algorithm for RVFL networks, which we will denote as LAP-RVFL. Apart from being a novel addition to the family of MR algorithms, we will use it in our experimental section as a state-of-the-art benchmark for the TR-RVFL introduced in the next section. Before proceeding, we define $\tilde{\mathbf{H}} \in \mathbb{R}^{(N+Q) \times B}$ as the hidden matrix computed over all available points:

$$\tilde{\mathbf{H}} = \begin{pmatrix} \mathbf{h}^T(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}^T(\mathbf{x}_N) \\ \mathbf{h}^T(\mathbf{x}_{N+1}) \\ \vdots \\ \mathbf{h}^T(\mathbf{x}_{N+Q}) \end{pmatrix}. \tag{11}$$

Since $f(\mathbf{x}) = \boldsymbol{\beta}^T\mathbf{h}(\mathbf{x})$, Eq. (4) in this case is given by:

$$r_{\mathcal{M}}(\boldsymbol{\beta}) = \left(\tilde{\mathbf{H}}\boldsymbol{\beta}\right)^T \mathbf{L} \left(\tilde{\mathbf{H}}\boldsymbol{\beta}\right). \tag{12}$$

Putting together Eqs. (3) and (8), and the previous expression, the optimization problem of the MR-based RVFL (LAP-RVFL) is given by:

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^B} J_{\text{MR}}(\boldsymbol{\beta}) = \frac{\lambda}{2} \left\| \tilde{\mathbf{H}}\boldsymbol{\beta} - \tilde{\mathbf{y}} \right\|_{\mathbf{C}}^2 + \frac{1}{2} \left\| \boldsymbol{\beta} \right\|_2^2 + \frac{\lambda_u}{2} \left\| \tilde{\mathbf{H}}\boldsymbol{\beta} \right\|_{\mathbf{L}}^2, \tag{13}$$

where $\tilde{\mathbf{y}} = \left[\mathbf{y}^T\ \hat{\mathbf{y}}^T\right]^T$, and $\mathbf{C}$ is an $(N+Q) \times (N+Q)$ diagonal matrix with the first N entries equal to λ and the last Q entries equal to 0. As before, the solution of Eq. (13) can be obtained by equating the gradient to 0:

$$\frac{\partial J_{\text{MR}}}{\partial \boldsymbol{\beta}} = \tilde{\mathbf{H}}^T\mathbf{C}\tilde{\mathbf{H}}\boldsymbol{\beta} - \tilde{\mathbf{H}}^T\mathbf{C}\mathbf{y} + \boldsymbol{\beta} + \lambda_u\tilde{\mathbf{H}}^T\mathbf{L}\tilde{\mathbf{H}}\boldsymbol{\beta} = 0. \tag{14}$$

From which we obtain the solution:

$$\boldsymbol{\beta}^*_{\text{MR}} = \left(\tilde{\mathbf{H}}^T\mathbf{C}\tilde{\mathbf{H}} + \mathbf{I} + \lambda_u\tilde{\mathbf{H}}^T\mathbf{L}\tilde{\mathbf{H}}\right)^{-1} \tilde{\mathbf{H}}^T\mathbf{C}\mathbf{y}. \tag{15}$$

If d is larger than $N+Q$, we can rewrite the previous formula using a computationally cheaper expression:

$$\boldsymbol{\beta}^*_{\text{MR}} = \tilde{\mathbf{H}}^T \left(\mathbf{C}\tilde{\mathbf{H}}\tilde{\mathbf{H}}^T + \mathbf{I} + \lambda_u\mathbf{L}\tilde{\mathbf{H}}\tilde{\mathbf{H}}^T\right)^{-1} \mathbf{C}\mathbf{y}. \tag{16}$$

4. Transductive RVFL networks

4.1. Formulation of the problem

An alternative semi-supervised approach to the training of RVFL networks is obtained by following the TR approach. Putting together the general formulation in Eq. (5) with Eq. (9) we obtain the following training problem:

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^B,\ \hat{\mathbf{y}} \in \{-1,+1\}^Q} J_{\text{TR}}(\boldsymbol{\beta}, \hat{\mathbf{y}}) = \frac{1}{2} \left\| \tilde{\mathbf{H}}\boldsymbol{\beta} - \tilde{\mathbf{y}} \right\|_{\boldsymbol{\Lambda}}^2 + \frac{1}{2} \left\| \boldsymbol{\beta} \right\|_2^2, \tag{17}$$

where $\tilde{\mathbf{y}} = \left[\mathbf{y}^T\ \hat{\mathbf{y}}^T\right]^T$ and $\boldsymbol{\Lambda} \in \mathbb{R}^{(N+Q)\times(N+Q)}$ is defined as:

$$\boldsymbol{\Lambda} = \text{diag}\left(\underbrace{\lambda, \ldots, \lambda}_{\text{N times}}, \underbrace{\lambda_u, \ldots, \lambda_u}_{\text{Q times}}\right). \tag{18}$$

It is easy to show that, differently from the TR-SVM case, the only free variables in (17) are the unknown entries of $\hat{\mathbf{y}}$. In particular, for every possible choice of $\hat{\mathbf{y}}$, we obtain a slightly modified version of the standard RVFL, whose solution with respect to $\boldsymbol{\beta}$ can be found as before by setting the gradient of Eq. (17) to 0, thus obtaining:

$$\boldsymbol{\beta}^*_{\text{TR}} = \left(\tilde{\mathbf{H}}^T\boldsymbol{\Lambda}\tilde{\mathbf{H}} + \mathbf{I}\right)^{-1}\tilde{\mathbf{H}}^T\boldsymbol{\Lambda}\tilde{\mathbf{y}}. \tag{19}$$

Back-substituting Eq. (19) in Eq. (17) we obtain a fully *integer programming* (IP) problem over the only variable $\hat{\mathbf{y}}$. Now we proceed to show that the problem admits a particularly convenient form. Define the matrix:

$$\mathbf{P} = \left(\tilde{\mathbf{H}}^T\boldsymbol{\Lambda}\tilde{\mathbf{H}} + \mathbf{I}\right)^{-1}\tilde{\mathbf{H}}^T\boldsymbol{\Lambda}. \tag{20}$$

If the number of labeled and unlabeled data is lower than the number of hidden neurons, as before, the matrix can alternatively be defined as:

$$\mathbf{P} = \tilde{\mathbf{H}}^T\left(\boldsymbol{\Lambda}\tilde{\mathbf{H}}\tilde{\mathbf{H}}^T + \mathbf{I}\right)^{-1}\boldsymbol{\Lambda}. \tag{21}$$

Combining the definition of $\mathbf{P}$ with Eq. (17) we have:

$$\min_{\hat{\mathbf{y}}\in\{-1,+1\}^Q} J_{\text{TR}}(\hat{\mathbf{y}}) = \frac{1}{2}\left\|\tilde{\mathbf{H}}\mathbf{P}\tilde{\mathbf{y}} - \tilde{\mathbf{y}}\right\|^2_{\boldsymbol{\Lambda}} + \frac{1}{2}\|\mathbf{P}\tilde{\mathbf{y}}\|^2_2. \tag{22}$$

A simple rearrangement shows that the previous problem is equivalent to:

$$\min_{\hat{\mathbf{y}}\in\{-1,+1\}^Q} J_{\text{TR}}(\hat{\mathbf{y}}) = \frac{1}{2}\|\tilde{\mathbf{y}}\|^2_{\mathbf{M}}, \tag{23}$$

where we defined the following matrix:

$$\mathbf{M} = \left(\tilde{\mathbf{H}}\mathbf{P} - \mathbf{I}\right)^T\boldsymbol{\Lambda}\left(\tilde{\mathbf{H}}\mathbf{P} - \mathbf{I}\right) + \mathbf{P}^T\mathbf{P}. \tag{24}$$

Finally, we need to express the objective function in term of $\hat{\mathbf{y}}$ only. Consider the block partitioning:

$$\mathbf{M} = \left[\begin{array}{c|c} \mathbf{M}_1 & \mathbf{M}_2 \\ \hline \mathbf{M}_3 & \mathbf{M}_4 \end{array}\right], \tag{25}$$

where $\mathbf{M}_1 \in \mathbb{R}^{N\times N}$ and the other blocks' sizes follow. Back-substituting this block partitioning in Eq. (23) we can write:

$$\begin{aligned}\tilde{\mathbf{y}}^T\mathbf{M}\tilde{\mathbf{y}} &= \left[\mathbf{y}^T\ \hat{\mathbf{y}}^T\right]\left[\begin{array}{c|c} \mathbf{M}_1 & \mathbf{M}_2 \\ \hline \mathbf{M}_3 & \mathbf{M}_4 \end{array}\right]\left[\begin{array}{c} \mathbf{y} \\ \hline \hat{\mathbf{y}} \end{array}\right] \\ &= \mathbf{y}^T\mathbf{M}_1\mathbf{y} + \hat{\mathbf{y}}^T\left(\mathbf{M}_3 + \mathbf{M}_2^T\right)\mathbf{y} + \hat{\mathbf{y}}^T\mathbf{M}_4\hat{\mathbf{y}} \\ &= \mathbf{y}^T\mathbf{M}_1\mathbf{y} + \hat{\mathbf{y}}^T\mathbf{M}_4\hat{\mathbf{y}} + \hat{\mathbf{y}}^T\mathbf{t},\end{aligned} \tag{26}$$

where we defined $\mathbf{t} = \left(\mathbf{M}_3 + \mathbf{M}_2^T\right)\mathbf{y}$. From this we obtain the final optimization problem, which is an unconstrained *0-1 quadratic programming problem* [25]:

$$\min_{\hat{\mathbf{y}}\in\{-1,+1\}^Q} J_{\text{TR}}(\hat{\mathbf{y}}) = \frac{1}{2}\left(\hat{\mathbf{y}}^T\mathbf{M}_4\hat{\mathbf{y}} + \hat{\mathbf{y}}^T\mathbf{t}\right). \tag{27}$$

where we ignored the term $\mathbf{y}^T\mathbf{M}_1\mathbf{y}$ which does not depend on $\hat{\mathbf{y}}$. This problem can be shown to be NP-hard in general [25]. Although several methods exist to obtain its optimal solution [7,24,26,30], they are not suitable for problems where the number of variables is larger than a few hundred elements. In the following section, we propose a simple relaxation of it, together with a criterion for checking whether the obtained solution is also optimal for the original problem.

4.2. A convex approximation

To relax the optimization problem in Eq. (27), we first make the following observation:

Lemma 1. *The matrix* $\mathbf{M}_4$ *in Eq. (27) is PSD.*

Proof. First, note that $\mathbf{P}^T\mathbf{P} \succeq 0$, since for any vector $\mathbf{a}$ we have $\mathbf{a}^T\mathbf{P}^T\mathbf{P}\mathbf{a} = \|\mathbf{P}\mathbf{a}\|^2_2 \geq 0$. A similar argument holds for the first term in the rhs of Eq. (24). Since the sum of two PSD matrices is PSD, we have that $\mathbf{M} \succeq 0$. From this we can conclude straightforwardly

the lemma:

$$\begin{bmatrix}\mathbf{0}^T & \mathbf{a}^T\end{bmatrix}\left[\begin{array}{c|c}\mathbf{M}_1 & \mathbf{M}_2 \\ \mathbf{M}_3 & \mathbf{M}_4\end{array}\right]\begin{bmatrix}\mathbf{0} \\ \mathbf{a}\end{bmatrix} = \mathbf{a}^T\mathbf{M}_4\mathbf{a} \geq 0\,, \tag{28}$$

□

Due to this, it makes sense to consider a convex approximation where the binary constraint is relaxed to the interval $[-1, +1]$ [25]:

$$\begin{aligned}\min_{\hat{\mathbf{y}}\in\mathbb{R}^Q} \quad & \frac{1}{2}\left(\hat{\mathbf{y}}^T\mathbf{M}_4\hat{\mathbf{y}} + \hat{\mathbf{y}}^T\mathbf{t}\right) \\ \text{subject to} \quad & \hat{y}_i^2 \leq 1,\ i = 1, \ldots, Q\end{aligned} \tag{29}$$

Problem (29) is a BCQ problem, which can be solved efficiently with a computational complexity of $\mathcal{O}(Q^3)$ [8]. It is particularly known in the machine learning community for being the optimization problem arising in the SVM training [19].

We propose to solve the original problem (27) by solving instead the relaxed version (29). We have an important theoretical proof regarding this simplification:

Theorem 1. *Denote by $\mathbf{y}_u^*$ the optimal solution to (29), and by $\mathbf{z}_u^*$ the solution obtained by rounding off every entry of $\mathbf{y}_u^*$ to the nearest integer. It can be shown [25] that the following is a sufficient condition for $\mathbf{z}_u^*$ to be an optimal solution to the original problem:*

$$\text{diag}(\mathbf{z}_u^*)\mathbf{M}_4(\mathbf{z}_u^* - \mathbf{y}_u^*) \leq \lambda_{\min}(\mathbf{M}_4)\mathbf{1} \tag{30}$$

where $\lambda_{\min}(\mathbf{M}_4)$ is the lowest eigenvalue of $\mathbf{M}_4$ and $\mathbf{1}$ is a vector with all entries equal to 1.

Proof. See [25, Section 2.6.2]. □

The previous condition can be used to check for optimality of the resulting vector $\mathbf{z}_u^*$. In our experience (as is also shown in the next section), even if the sufficient condition is not met, the resulting RVFL model possesses good generalization capabilities. The final output vector can be computed as:

$$\boldsymbol{\beta}_{\text{TR}}^* = \mathbf{P}\begin{bmatrix}\mathbf{y} \\ \mathbf{z}_u^*\end{bmatrix}. \tag{31}$$

The overall algorithm for TR-RVFL is summarized in Algorithm 1.

Algorithm 1
Training algorithm for TR-RVFL.

Inputs: Training set S, unlabeled set U, regularization coefficients λ, λ_u.
Output: Optimal vector $\boldsymbol{\beta}_{\text{TR}}^*$.
1: Compute hidden matrix $\tilde{\mathbf{H}}$ according to Eq. (11).
2: Compute weighting matrix $\boldsymbol{\Lambda}$ according to Eq. (18).
3: Compute matrix $\mathbf{P}$ using Eq. (20) or Eq. (21).
4: Set $\mathbf{M} = \left(\tilde{\mathbf{H}}\mathbf{P} - \mathbf{I}\right)^T\boldsymbol{\Lambda}\left(\tilde{\mathbf{H}}\mathbf{P} - \mathbf{I}\right) + \mathbf{P}^T\mathbf{P}$.
5: Solve optimization problem in Eq. (29) and obtain $\mathbf{y}_u^*$.
6: Set $\mathbf{z}_u^* = \text{sign}(\mathbf{y}_u^*)$.
7: **return** $\boldsymbol{\beta}_{\text{TR}}^* = \mathbf{P}\begin{bmatrix}\mathbf{y} \\ \mathbf{z}_u^*\end{bmatrix}$

5. Experiments

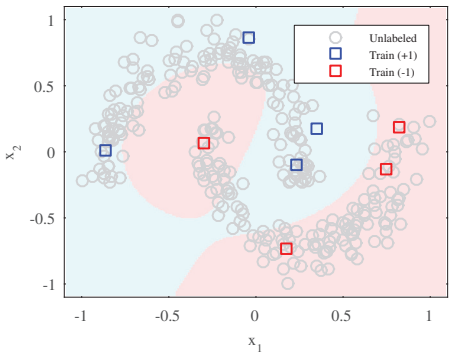
In this section we perform an extensive range of experiments to validate our proposal. MATLAB code for repeating the simulations is available under open source license on the corresponding author's website.[4] The quadratic programming problem in Eq. (29) is solved using the BOXCQP algorithm [40], which is found to be faster than the built-in MATLAB solver.

5.1. Experiment on two moons dataset

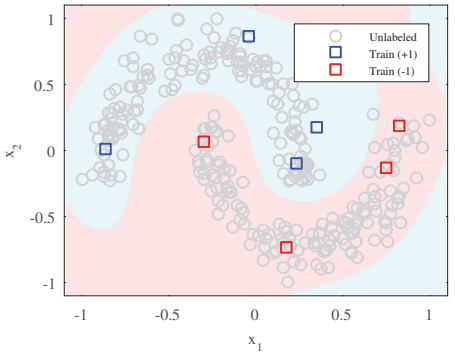
We begin our evaluation with a simple experiment on the two moons dataset (see e.g. [5]). This is a synthetic dataset, where the input is a 2-dimensional feature vector, and the two classes are arranged as two symmetrical semi-circular shapes. We consider a standard situation with few labeled points (8 in our case), and a large unlabeled set of 296 points. We compare RVFL and TR-RVFL in this setting, using sigmoid hidden functions given by:

$$h(\mathbf{x}; \mathbf{w}, b) = \frac{1}{1 + \exp\{-\mathbf{w}^T\mathbf{x} + b\}}\,, \tag{32}$$

[4] http://ispac.diet.uniroma1.it/scardapane/software/lynx/semisupervised-rvfl/ [last visited on 2015-07-09].



(a) RVFL



(b) TR-RVFL

Fig. 1. Decision boundaries of (a) RVFL and (b) TR-RVFL. Labeled training points are shown with squares, while unlabeled training points with gray circles. The area for which $\text{sign}(f(\mathbf{x})) = -1$ is shown in light blue, while the complement in light red.(For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
General description of the datasets used in the experiments.

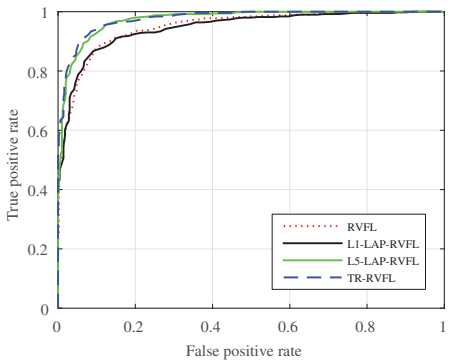
Name	Features	Instances	|Tr|	|V|	|Tst|	|U|
g50c	50	550	30	100	100	320
g241c	241	1500	100	100	100	1200

where parameters $\mathbf{w}$ and b in (32) are extracted randomly from an uniform distribution over the interval $[-\alpha, +\alpha]$, $\alpha > 0$. As is known since the seminal work in [20], a proper setting of α would require an exact knowledge of the underlying probability distribution of the training data, which is not known. After some preliminary experiments on an independent validation set, it was found that, with proper normalization of the datasets that we consider, results were equivalent for $\alpha \geq 1$, which is also in line with our previous works [33,34]. Thus, we set $\alpha = 1$ throughout the rest of the experimental section, although we acknowledge that more research is needed on this open problem. As for the rest of the parameters, we set a predefined number of hidden neurons as $B = 1000$. The regularization coefficients λ and λ_u are searched in the exponential interval 2^j, $j \in \{-25, \ldots, 25\}$ using an independent validation set of 50 elements. Average results over 50 runs are $\lambda = 2^{22}$ for RVFL and $\lambda = 2^{14}, \lambda_u = 2^{22}$ for TR-RVFL.

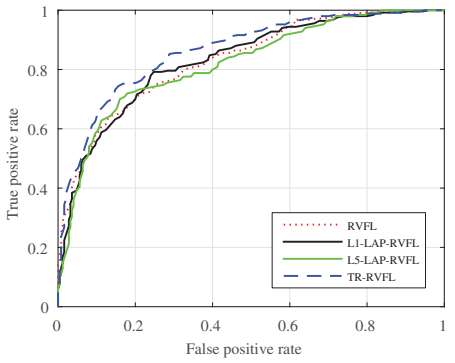
Over 50 experiments, RVFL achieves an average misclassification error (computed over a fourth test set of 50 elements) of 16%, against a misclassification error of 2% for TR-RVFL. It is particularly interesting to visually inspect the resulting decision boundaries. These are shown for a randomly sampled experiment in Fig. 1. It is easy to see that RVFL (Fig. 1b) is not able to generalize sufficiently well, starting only from the labeled points. In particular, it creates an isolated region of class -1 which extends over multiple points in the opposite class. On the contrary, TR-RVFL in Fig. 1c is able to correctly infer the overall decision boundary, with only a small distortion on the right side.

5.2. Comparison with LAP-RVFL and TR-SVM

As a second set of experiments, we evaluate the performance of TR-RVFL in two standard benchmark problems for SSL, whose characteristics are briefly summarized in Table 1. g50c is an artificial dataset, whose input is drawn with equal probability from two different 50-dimensional Gaussian distributions with unitary covariance [37]. The class centers are adjusted so that the



(a) g50c



(b) g241c

Fig. 2. ROC curves for the two datasets. The proposed algorithm is shown with a dashed blue line.(For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

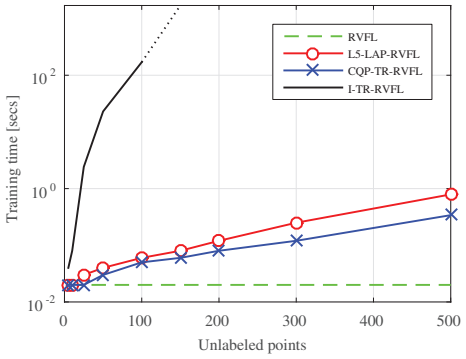
Table 2
Final misclassification error and training time for the five algorithms, together with one standard deviation. Results of the proposed algorithm are highlighted in bold.

Dataset	Algorithm	Error [%]	Time [s]
g50c	RVFL	13.90 ± 3.65	0.005 ± 0.001
	L1-LAP-RVFL	12.40 ± 2.76	0.041 ± 0.001
	L5-LAP-RVFL	8.90 ± 3.05	0.112 ± 0.005
	TR-SVM	10.90 ± 4.13	0.445 ± 0.1
	TR-RVFL	**7.70 ± 3.35**	**0.043 ± 0.001**
g241c	RVFL	26.00 ± 4.90	0.015 ± 0.004
	L1-LAP-RVFL	24.67 ± 3.77	0.522 ± 0.003
	L5-LAP-RVFL	24.33 ± 4.78	3.225 ± 0.016
	TR-SVM	20.20 ± 3.87	5.471 ± 12.82
	TR-RVFL	**19.67 ± 3.77**	**0.598 ± 0.002**

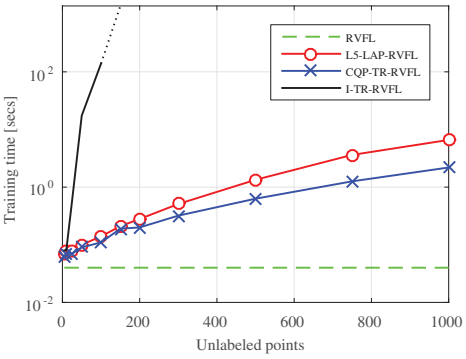
Bayes error is exactly 5%. Similarly, in the g241c the inputs are drawn from two 241-dimensional isotropic Gaussian distributions [12]. Both datasets are normalized feature-wise in the interval $[-1, +1]$ before training.

As a comparison, we use the standard RVFL and the LAP-RVFL algorithm, introduced in Section 3. We use the same experimental setting as the previous section, i.e. we split the original datasets in four components, namely a training set Tr, a validation set V, a test set Tst and an unlabeled set U. Details on these sets are given in Table 1. We search the parameters λ, λ_u for the three models using the same grid-search procedure detailed before. The number of hidden nodes B is set as $B = 1000$ for g50c, and $B = 2000$ for g241c. Following standard conventions in MR-based algorithms, the adjacency matrix $\mathbf{W}$ (see Section 2.1) is constructed as follows:

$$W_{ij} = \begin{cases} \exp\left\{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right\} & \text{if } \mathbf{x}_i \in \mathcal{N}_j \text{ or } \mathbf{x}_j \in \mathcal{N}_i \\ 0 & \text{otherwise} \end{cases}, \tag{33}$$



(a) g50c



(b) g241c

Fig. 3. Training time required by the four algorithms, when considering unlabeled datasets of growing size. The y-axis is shown in a logarithmic scale for improved readability. For the same reason, training times for I-TR-RVFL are not shown for $|U| > 100$.

where $\mathcal{N}_j$ is the set of k-nearest neighbors of $\mathbf{x}_j$, with k set *a priori*, and σ is set as the mean edge distance among the patterns in $\mathcal{N}_j$. Then, to improve performance, we replace the original Laplacian matrix $\mathbf{L}$ with the following matrix:

$$\mathbf{L}' = \left(\mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}\right)^p, \tag{34}$$

where $\mathbf{D}$ is the degree matrix defined in Section 2.1 and p is a suitable integer set by the user. The Laplacian computation is performed using the primal Laplacian SVM MATLAB library [28].[5] In order to provide a fair comparison, we do not optimize the parameters for constructing the Laplacian, but we use common heuristics that are generally found to work well. In particular, we set $k = 30$ in Eq. (33), and we test two versions of the algorithms, with $p = 1$ and $p = 5$. We denote them by L1-LAP-RVFL and L5-LAP-RVFL, respectively.

The average misclassification error and training time for the five algorithms are provided in Table 2. We also show the comparison with TR-SVM, implemented using SVMLight [21]. For TR-SVM, we use a Gaussian kernel, and we optimize its internal parameter using the same grid-search procedure detailed before. Experiments are repeated over 30 different runs. First of all, it can be seen from the third and fourth column of Table 2 that the performance of LAP-RVFL is highly dependent on the setting of p. In these two datasets, in particular, large values of p tend to provide a better accuracy, with a much larger computational cost. In fact, L5-LAP-RVFL is three times slower with respect to L1-LAP-RVFL for the g50c dataset, and almost six times slower for g241c. Although this is not shown in this set of experiments, its performance is also dependent on the number of neighbors in Eq. (33), albeit on a lower degree, and it can change by choosing a different strategy for initializing the adjacency matrix [5,36]. TR-RVFL is instead untouched by this problem, since it introduces only one free parameter in the optimization problem. Additionally, it can be seen from Table 2 that it is able to provide the best performance in both cases, with a strong improvement over the purely supervised setting, and over the two versions of LAP-RVFL. This is obtained with a training time which is comparable to L1-LAP-RVFL, but significantly smaller than L5-LAP-RVFL. The performance of TR-SVM is intermediate between LAP-RVFL and TR-RVFL, albeit its training time is considerably larger.

To further strengthen this point, we provide the ROC curves [9] for the two datasets in Fig. 2. It can be seen that TR-RVFL performs slightly better than L5-LAP-RVFL in the g50c dataset (Fig. 2a), and outperforms the three other algorithms in the g241c dataset (Fig. 2b).

[5] http://www.dii.unisi.it/~melacci/lapsvmp/ [last visited on 2015-07-09].

5.3. Comparison with integer programming solver

As a last evaluation, we provide an analysis of the training time required by TR-RVFL, when compared to RVFL and L5-LAP-RVFL. Additionally, we show the training time that would be required to directly solve the 0–1 problem in Eq. (27). This is solved using the state-of-the-art SCIP solver [1]. The OPTI toolbox[6] is used to interface the solver with MATLAB. This version of TR-RVFL is denoted as I-TR-RVFL, while the one used in the previous sections as CQP-TR-RVFL. We use the same settings as before, but we consider unlabeled datasets of growing size, with $|\mathrm{U}| = \{5, 10, 25, 50, 100, 150, 200, 300\}$. For the g241c dataset, we also consider $|\mathrm{U}| = 500$ and $|\mathrm{U}| = 1000$. Additionally, for simplicity we set $\lambda = \lambda_u = 1$ without optimizing the two parameters. Results are given in Fig. 3. For better readability, the y-axis in Fig. 3 is shown in a logarithmic scale.

Clearly, directly solving the 0–1 problem of Eq. (3) is impractical even for small unlabeled datasets, since the required time grows exponentially with respect to the number of variables. This is in line with what has been discussed in Section 4.1. At the same time, it can be seen that CQP-TR-RVFL is able to provide a significantly smaller training time with respect to L5-LAP-RVFL, and the gap between the two algorithms grows as the number of unlabeled points increases.

6. Conclusions

In this paper, we introduced a novel semi-supervised training procedure for RVFL networks, denoted as TR-RVFL. The algorithm is inspired by the transductive SVM, since it considers the unknown labels as additional variables in the optimization problem. We showed that TR-RVFL results in an unconstrained 0–1 problem, which can be approximated by a standard CQP problem, solvable in polynomial time. Our experimental evaluation proves that it is able to reach state-of-the-art performance, both in terms of misclassification error and training time. In this paper we have focused on the binary classification case. Future work will consider its extension to the case of multiclass classification, regression, one-class learning, and to the combination of multiple RVFL networks [16]. Additionally, we are investigating alternative strategies for solving the overall optimization problem, as well as the possibility of including additional constraints, such as the desired ratio of labels belonging to each class (as is common in the transductive SVM).

References

[1] T. Achterberg, SCIP: solving constraint integer programs, Math. Program. Comput. 1 (1) (2009) 1–41.
[2] M.M. Adankon, M. Cheriet, A. Biem, Semisupervised least squares support vector machine, IEEE Trans. Neural Netw. 20 (12) (2009) 1858–1870.
[3] M. Alhamdoosh, D. Wang, Fast decorrelated neural network ensembles with random weights, Inf. Sci. 264 (2014) 104–117.
[4] M. Belkin, P. Niyogi, Semi-supervised learning on Riemannian manifolds, Mach. learn. 56 (1-3) (2004) 209–239.
[5] M. Belkin, P. Niyogi, V. Sindhwani, Manifold regularization: a geometric framework for learning from labeled and unlabeled examples, J. Mach. Learn. Res. 7 (2006) 2399–2434.
[6] K. Bennett, A. Demiriz, Semi-supervised support vector machines, Adv. Neural Inf. Process. Syst. (1999) 368–374.
[7] A. Billionnet, S. Elloumi, Using a mixed integer quadratic programming solver for the unconstrained quadratic 0–1 problem, Math. Program. 109 (1) (2007) 55–68.
[8] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.
[9] A.P. Bradley, The use of the area under the ROC curve in the evaluation of machine learning algorithms, Pattern Recognit. 30 (7) (1997) 1145–1159.
[10] B. Chandra, M. Gupta, A novel approach for distance-based semi-supervised clustering using functional link neural network, Soft Comput. 17 (3) (2013) 369–379.
[11] O. Chapelle, V. Vapnik, J. Weston, Transductive inference for estimating values of functions., Adv. Neural Inf. Process. Syst. 12 (1999) 421–427.
[12] O. Chapelle, B. Schölkopf, A. Zien, Semi-supervised Learning, The MIT Press, 2006.
[13] O. Chapelle, V. Sindhwani, S.S. Keerthi, Optimization techniques for semi-supervised support vector machines, J. Mach. Learn. Res. 9 (2008) 203–233.
[14] R. Collobert, F. Sinz, J. Weston, L. Bottou, Large scale transductive SVMs, J. Mach. Learn. Res. 7 (2006) 1687–1712.
[15] D. Comminiello, M. Scarpiniti, L.A. Azpicueta-Ruiz, J. Arenas-Garcia, A. Uncini, Functional link adaptive filters for nonlinear acoustic echo cancellation, IEEE Trans. Audio, Speech, and Lang. Process. 21 (7) (2013) 1502–1512.
[16] D. Comminiello, M. Scarpiniti, S. Scardapane, R. Parisi, A. Uncini, Improving nonlinear modeling capabilities of functional link adaptive filters, Neural Netw. 69 (2015) 51–59.
[17] B. Geng, D. Tao, C. Xu, Y. Yang, X.-S. Hua, Ensemble manifold regularization, IEEE Trans. Pattern Anal. Mach. Intell. 34 (6) (2012) 1227–1233.
[18] H. Grabner, C. Leistner, H. Bischof, Semi-supervised On-Line Boosting for Robust Tracking, in: Proceedings of the 2008 European Conference on Computer Vision (EECV'08), Springer Berlin Heidelberg, 2008, pp. 234–247.
[19] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, 2nd edition, Springer, 2009.
[20] B. Igelnik, Y.-H. Pao, Stochastic choice of basis functions in adaptive function approximation and the functional-link net, IEEE Trans. Neural Netw. 6 (6) (1995) 1320–1329.
[21] T. Joachims, Making large-scale support vector machine learning practical, in: Advances in Kernel Methods, MIT Press, Cambridge, MA, USA, 1999a, pp. 169–184.
[22] T. Joachims, Transductive inference for text classification using support vector machines, in: Proceedings of the 16th International Conference on Machine Learning (ICML'99), 1999b.
[23] L. Käll, J.D. Canterbury, J. Weston, W.S. Noble, M.J. MacCoss, Semi-supervised learning for peptide identification from shotgun proteomics datasets, Nature methods 4 (11) (2007) 923–925.
[24] G. Kochenberger, J.-K. Hao, F. Glover, M. Lewis, Z. Lü, H. Wang, Y. Wang, The unconstrained binary quadratic programming problem: a survey, J. Combinatorial Optim. 28 (1) (2014) 58–81.
[25] D. Li, X. Sun, Nonlinear Integer Programming, Springer, 2006.
[26] D. Li, X.L. Sun, C.L. Liu, An exact solution method for unconstrained quadratic 0–1 programming: a geometric approach, J. Global Optim. 52 (4) (2012) 797–829.
[27] Y.-F. Li, I.W. Tsang, J.T. Kwok, Z.-H. Zhou, Convex and scalable weakly labeled SVMs, J. Mach. Learn. Res. 14 (1) (2013) 2151–2188.
[28] S. Melacci, M. Belkin, Laplacian support vector machines trained in the primal, J. Mach. Learn. Res. 12 (2011) 1149–1184.

[6] http://www.i2c2.aut.ac.nz/Wiki/OPTI/ [last visited on 2015-07-09].

[29] Y.-H. Pao, G.-H. Park, D.J. Sobajic, Learning and generalization characteristics of the random vector functional-link net, Neurocomputing 6 (2) (1994) 163–180.
[30] S. Poljak, F. Rendl, H. Wolkowicz, A recipe for semidefinite relaxation for (0, 1)-quadratic programming, J. Glob. Optim. 7 (1) (1995) 51–73.
[31] B. Quanz, J. Huan, Model Selection for Semi-supervised Learning With Limited Labeled Data Technical report, Information Telecommunication and Technology Center, University of Kansas, 2012. ITTC-FY2013-TR-65071-01.
[32] M. Roy, S. Ghosh, A. Ghosh, A novel approach for change detection of remotely sensed images using semi-supervised multiple classifier system, Inf. Sci. 269 (2014) 35–47.
[33] S. Scardapane, R. Fierimonte, D. Wang, M. Panella, A. Uncini, Distributed music classification using random vector functional-link nets, in: Proceedings of 2015 INNS/IEEE International Joint Conference on Neural Networks (IJCNN'15), 2015.
[34] S. Scardapane, D. Wang, M. Panella, A. Uncini, Distributed learning for random vector functional-link networks, Inf. Sci. 301 (2015) 271–284.
[35] W.F. Schmidt, M.A. Kraaijveld, R.P.W. Duin, Feedforward neural networks with random weights, in: Proceedings of the 11th IAPR International Conference on Pattern Recognition (ICPR'92), IEEE, 1992, pp. 1–4.
[36] V. Sindhwani, D.S. Rosenberg, An RKHS for multi-view learning and manifold co-regularization, in: Proceedings of the 25th international conference on Machine learning (ICML'08), ACM, 2008, pp. 976–983.
[37] V. Sindhwani, P. Niyogi, M. Belkin, Beyond the point cloud: from transductive to semi-supervised learning, in: Proceedings of the 22nd international conference on Machine learning (ICML'05), ACM, 2005, pp. 824–831.
[38] I.W. Tsang, J.T. Kwok, Large-scale sparsified manifold regularization, in: Advances in Neural Information Processing Systems, 2006, pp. 1401–1408.
[39] V. Vapnik, The nature of statistical learning theory, Springer Science & Business Media, 2000.
[40] C. Voglis, I.E. Lagaris, BOXCQP: an algorithm for bound constrained convex quadratic problems, in: Proceedings of the 1st International Conference: From Scientific Computing to Computational Engineering, IC-SCCE, 2004.
[41] J. Wang, X. Shen, W. Pan, On transductive support vector machines, Contemp. Math. 443 (2007) 7–20.