Neurocomputing 123 (2014) 23–32

Contents lists available at ScienceDirect

Neurocomputing

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

Fast and simple gradient-based optimization for semi-supervised support vector machines[☆]

Fabian Gieseke[a,*], Antti Airola[b,c], Tapio Pahikkala[b,c], Oliver Kramer[a]

[a] Computer Science Department, Carl von Ossietzky Universität Oldenburg, 26111 Oldenburg, Germany
[b] Department of Information Technology, 20014, University of Turku, Finland
[c] Turku Centre for Computer Science (TUCS), Joukahaisenkatu 3-5 B, 20520 Turku, Finland

ARTICLE INFO

Article history:
Received 15 April 2012
Received in revised form
16 September 2012
Accepted 17 December 2012
Available online 21 April 2013

Keywords:
Semi-supervised support vector machines
Non-convex optimization
Quasi-Newton methods

ABSTRACT

One of the main learning tasks in machine learning is the one of classifying data items. The basis for such a task is usually a training set consisting of labeled patterns. In real-world settings, however, such labeled data are usually scarce, and the corresponding models might yield unsatisfying results. Unlabeled data, on the other hand, can often be obtained in huge quantities without much additional effort. A prominent research direction in the field of machine learning is semi-supervised support vector machines. This type of binary classification approach aims at taking the additional information provided by the unlabeled patterns into account to reveal more information about the structure of the data at hand. In some cases, this can yield significantly better classification results compared to a straightforward application of supervised models. One drawback, however, is the fact that generating such models requires solving difficult non-convex optimization tasks. In this work, we present a simple but effective gradient-based optimization framework to address the induced problems. The resulting method can be implemented easily using black-box optimization engines and yields excellent classification and runtime results on both sparse and non-sparse data sets.

1. Introduction

One of the most important machine learning tasks is classification. If sufficient labeled training data are given, there exists a variety of techniques like the *k-nearest neighbor*-classifier or *support vector machines* (SVMs) [2,3] to address such a task. However, labeled data are often rare in real-world applications. One active research field in machine learning is *semi-supervised learning* [4,5]. In contrast to supervised methods, the latter class of techniques takes both labeled and unlabeled data into account to construct appropriate models. A well-known concept in this field is *semi-supervised support vector machines* (S^3VMs) [6–8], which depict the direct extension of support vector machines to semi-supervised learning scenarios. The key idea is depicted in Fig. 1: The aim of a standard support vector machine consists in finding a hyperplane which separates both classes well such that the margin is maximized. It is obvious that, in case of lack of labeled data, suboptimal models might be obtained, see Fig. 1(a). Its semi-supervised variant aims at taking the unlabeled patterns into account by searching for a partition (into two classes) such that a *subsequent* application of a modified support vector machine leads to the best result. Under certain conditions, unlabeled data can provide valuable information, see Fig. 1(b). While being very appealing from a practical point of view, semi-supervised support vector machines lead to a combinatorial optimization task that is difficult to approach.

The original problem formulation of semi-supervised support vector machines was given by Vapnik and Sterin [8] under the name of *transductive support vector machines*. From an optimization point of view, the first approaches have been proposed in the late nineties by Joachims [7] and Bennet and Demiriz [6]. In general, there are two lines of research, namely (a) *combinatorial* and (b) *continuous* optimization schemes. The brute-force approach (which tests every possible partition), for instance, is among the combinatorial schemes since it aims at directly finding a good assignment for the unknown labels.

1.1. Related work

For both the combinatorial and the continuous research directions, a variety of different techniques has been proposed in recent years. The former one is usually addressed by label-switching

[☆] This work depicts an extended version of the associated conference paper that has been presented at the 1st International Conference on Pattern Recognition Applications and Methods [1]. It contains additional theoretical derivations related to incorporating an offset term and a balancing constraint. Moreover, the experimental evaluation has been extended by adding two more semi-supervised competitors as well as a variety of high-dimensional sparse data sets.

[*] Corresponding author. Tel.: +49 4417984374; fax: +49 4417982756.
E-mail addresses: f.gieseke@uni-oldenburg.de (F. Gieseke), antti.airola@utu.fi (A. Airola), tapio.pahikkala@utu.fi (T. Pahikkala), oliver.kramer@uni-oldenburg.de (O. Kramer).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2012.12.056

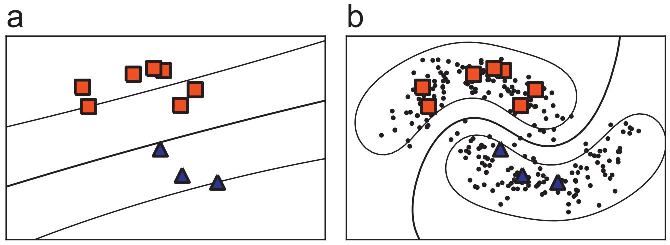

Fig. 1. The concepts of support vector machines and their extension to semi-supervised learning settings. Labeled patterns are depicted as red squares and blue triangles and unlabeled patterns as black points, respectively: (a) SVM, (b) S^3VM. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

strategies [7,9,10] or by reformulating the original task as semi-definite programming problem [11,12]. Further, since both real and integer variables are present in the optimization task (see below), mixed-integer programming solvers can be applied to compute optimal solutions up to machine precision [6]. Another way to obtain optimal solutions are branch and bound frameworks, see Chapelle et al. [13] for an appropriate algorithm.

The continuous optimization perspective leads to a real-valued but non-convex task (see below). Among the first schemes that considered this perspective was the gradient descent framework of Chapelle and Zien [14], which was based on the replacement of the original loss functions by appropriate surrogates. Similar ideas led to the continuation framework [15], to deterministic annealing methods [16,10], and to the use of the (constrained) concave–convex procedure [17–19]. An approach closely related to the one proposed in this work is the quasi-Newton framework proposed by Reddy et al. [20]; however, they do not consider differentiable surrogates and therefore apply more complicated sub-gradient methods.

Despite the methods mentioned above, a variety of other semi-supervised support vector machine variants have been proposed in the literature including, e.g., graph-based methods [21]. Due to lack of space, we refer to Chapelle et al. [4,22] and Zhu and Goldberg [5] for comprehensive surveys. It is worth pointing out that support vector machines can also be extended to unsupervised learning settings (without any labeled patterns at all) in a very similar kind of way. This variant is known as *maximum margin clustering* and has received a considerable interest in recent years [23–28].

1.2. Contribution

In this work, we will show that quasi-Newton schemes [29] along with direct computational shortcuts for sparse and non-sparse data depict simple but very effective approaches for the task at hand. In particular, we make use of an appropriate differentiable surrogate of the original objective and show that one can directly obtain computational shortcuts for non-sparse data (and arbitrary kernels) via the subset of regressors [30] scheme, and for sparse data (and the linear kernel) by taking advantage of the explicit structure of the objective function and its gradient. The induced optimization approaches are conceptually very simple and can be implemented easily via standard black-box optimization tools.[1]

As part of the contribution, we provide a detailed experimental evaluation and compare both the classification and runtime performances of our implementation with state-of-the-art semi-supervised support vector machine implementations on a variety of sparse and non-sparse data sets. The results clearly indicate the usability and effectiveness of our implementation.

1.3. Notations

We use $[m]$ to denote the set $\{1,\ldots,m\}$. Given a vector $\mathbf{y}\in\mathbb{R}^n$, we use y_i to denote its i-th coordinate. Further, the set of all $m\times n$ matrices with real coefficients is denoted by $\mathbb{R}^{m\times n}$. Given a matrix $\mathbf{M}\in\mathbb{R}^{m\times n}$, we denote the element in the i-th row and j-th column by $[\mathbf{M}]_{i,j}$. For two sets $R=\{i_1,\ldots,i_r\}\subseteq[m]$ and $S=\{k_1,\ldots,k_s\}\subseteq[n]$ of indices, we use $\mathbf{M}_{R,S}$ to denote the matrix that contains only the rows and columns of $\mathbf{M}$ that are indexed by R and S, respectively. Moreover, we set $\mathbf{M}_{R,[n]}=\mathbf{M}_R$. All vectors are assumed to be column vectors and the superscript T is used to denote the transpose of a matrix or a vector, i.e., $\mathbf{y}^{\mathrm{T}}$ is a row vector and $\mathbf{M}^{\mathrm{T}}\in\mathbb{R}^{n\times m}$ is the transpose of the matrix $\mathbf{M}\in\mathbb{R}^{m\times n}$.

2. Classification task

In the following, we will consider a set $T_l=\{(\mathbf{x}_1,y'_1),\ldots,(\mathbf{x}_l,y'_l)\}$ of labeled patterns and a set $T_u=\{\mathbf{x}_{l+1},\ldots,\mathbf{x}_{l+u}\}\subset X$ of unlabeled training patterns that belong to an arbitrary set X.

2.1. Support vector machines

The concept of support vector machines can be seen as instance of regularization problems of the form

$$\inf_{f\in\mathcal{H}}\left\{\frac{1}{l}\sum_{i=1}^{l}\mathcal{L}(y'_i,f(\mathbf{x}_i))+\lambda\|f\|^2_{\mathcal{H}}\right\},\tag{1}$$

where $\lambda>0$ is a fixed real number, $\mathcal{L}:Y\times\mathbb{R}\to[0,\infty)$ is a *loss function* and $\|f\|^2_{\mathcal{H}}$ is the squared norm in a so-called *reproducing kernel Hilbert space* $\mathcal{H}\subseteq\mathbb{R}^X=\{f:X\to\mathbb{R}\}$ induced by a *kernel function* $k:X\times X\to\mathbb{R}$ [3]. Here, the first term measures the loss caused by the prediction function on the labeled training set and the second one penalizes complex functions. Plugging in different loss functions leads to various models; one of the most popular choices is the *hinge loss* $\mathcal{L}(y,t)=\max(0,1-yt)$, which yields the original definition of support vector machines [3,31], see Fig. 2(a).[2]

2.2. Semi-supervised SVMs

Given the additional set $T_u=\{\mathbf{x}_{l+1},\ldots,\mathbf{x}_{l+u}\}\subset X$ of unlabeled training patterns, semi-supervised support vector machines [6–8] aim at finding an optimal prediction function for unseen data based on both the labeled and the unlabeled parts of the data. More precisely, we search for a function $f^*\in\mathcal{H}$ and a labeling vector $\mathbf{y}^*=(y^*_{l+1},\ldots,y^*_{l+u})^{\mathrm{T}}\in\{-1,+1\}^u$ that are optimal with respect to $\min_{f\in\mathcal{H},\ \mathbf{y}\in\{-1,+1\}^u}J(f,\mathbf{y})$ where

$$J(f,\mathbf{y})=\frac{1}{l}\sum_{i=1}^{l}\mathcal{L}^1(y'_i,f(\mathbf{x}_i))+\frac{\lambda'}{u}\sum_{i=l+1}^{l+u}\mathcal{L}^1(y_i,f(\mathbf{x}_{l+i}))+\lambda\|f\|^2_{\mathcal{H}}.\tag{2}$$

Here, $\lambda',\lambda>0$ are user-defined parameters and $\mathcal{L}^1:\mathbb{R}\times\mathbb{R}\to[0,\infty)$ a loss function. Thus, the main task consists in finding the optimal assignment vector $\mathbf{y}$ for the unlabeled part; the combinatorial nature of this task renders the optimization problem difficult to solve.

[1] The code can be obtained from the authors upon request.

[2] The latter formulation does not include a bias term $b\in\mathbb{R}$, which addresses translated data. For complex kernel functions like the RBF kernel, adding this bias term does not yield any known advantages, both from a theoretical and practical point of view [3]. In the remainder of this work, we will mostly omit the bias term for the sake of exposition; however, such a bias term can be explicitly incorporated into the optimization frameworks presented in this work, as we will show below.