Ensemble approaches to semi-supervised learning
Tanha, J.

Citation for published version (APA):
Tanha, J. (2013). Ensemble approaches to semi-supervised learning

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
In this chapter we present an algorithm for multiclass semi-supervised learning. Existing semi-supervised algorithms use approaches such as one-versus-all to convert the multiclass problem to several binary classification problems. However, this is not optimal. We propose a multiclass semi-supervised boosting algorithm that solves multiclass classification problems directly. The algorithm is based on a novel multiclass loss function consisting of the margin cost on labeled data and two regularization terms on labeled and unlabeled data. Experimental results on a number of benchmark UCI datasets show that the proposed algorithm performs better than the state-of-the-art boosting algorithms for multiclass semi-supervised learning.

Preliminary research on multiclass semi-supervised learning is published by Tanha et al. [95], at the IEEE International Conference on Data Mining (ICDM) in 2012. Also, an extended version of this work is accepted for publication in the Special issue on “Partially Supervised Learning for Pattern Recognition” of the journal of Pattern Recognition Letters.

5.1 Introduction

Most of the existing semi-supervised learning methods were originally designed for binary classification problems. Nevertheless, many application domains, such as text and document classification, pattern recognition, and text mining, involve more than two classes. To solve the multiclass classification problem two main approaches have been proposed. The first is to convert the multiclass problem into a set of binary classification problems. Examples of this approach include one-vs-all, one-vs-one, and error-correcting output code [30]. This approach can have various problems such as imbalanced class distribution, increased complexity, no guarantee to have an optimal joint classifier or probability estimation, and different scales for the outputs of generated binary classifiers which complicates combining them, see [48] and [79]. The second approach is to use a multiclass
Chapter 5. Multiclass Semi-Supervised Boosting

classifier directly. Although a number of methods have been proposed for multiclass supervised learning, for example [69], [110], and [79], they are not able to handle multiclass semi-supervised learning, which is the aim of this study.

Most of the semi-supervised learning methods work iterative or additive. Iterative semi-supervised learning algorithms replace their hypothesis at each iteration by a new one. Additive algorithms add a new component to a linear combination as in boosting algorithms. Typically, self-training algorithms are iterative. Additive algorithms, like boosting [35], are generic ensemble learning frameworks, which sequentially construct linear combination of base classifiers. Boosting is one of the most successful ensemble methods for supervised learning. The boosting approach has been extended to semi-supervised learning with different strategies. Examples are MarginBoost [23] and ASSEMBLE [6]. The main difficulties in this approach are how to sample the unlabeled examples and which class labels to assign to them. MarginBoost and ASSEMBLE use the classifier to predict the class labels of the unlabeled examples, the “pseudo-labels”. These algorithms then attempt to minimize the margin cost for the labeled data and a cost associated with the “pseudo-labels” of the unlabeled data. A sample of the pseudo-labeled data is then used in the next iteration. The pseudo-labels and also the associated cost depend strongly on the classifier predictions and therefore this type of algorithm cannot exploit information from the unlabeled data and the final decision boundary is very close to that of the initial classifier, see [20] and [66].

To solve the above problem, a recent idea is to use a smoothness regularizer based on the cluster and the manifold assumptions. The idea is to not use the “pseudo-margin” from the predictions of the base learner directly. Instead, beside the margin of the labeled data, also the “consistency” is maximized. Consistency is a form of smoothness. Class labels are consistent if they are equal for data that are similar. For this the definition of the weight for data and for classifiers in the boosting algorithm must be modified. This is done in SemiBoost [66] and RegBoost [20]. Experiments show that this idea is more effective than the approach based on “pseudo-margin” or confidence, see [20] and [66]. However, this solution is limited to binary classification problems. For a multiclass semi-supervised learning problem, the state-of-the-art algorithms are now those based on the “pseudo-margin” [90] or the use of binary methods, like SemiBoost and RegBoost, to handle the multiclass case employing the one-versus-all or similar meta-algorithm, which is problematic as mentioned earlier. Recently, in [99] a new boosting method is used for semi-supervised multiclass learning which uses similarity between predictions and data. It maps labels to n-dimensional space, but this mapping may not lead to train a classifier that minimizes the margin cost properly.

In this chapter we present a boosting algorithm for multiclass semi-supervised learning, named Multi-SemiAdaBoost. Unlike many semi-supervised learning algorithms that are extensions of specific base classifiers, like SVM, to semi-
supervised learning, Multi-SemiAdaBoost can boost any kind of base classifier. It minimizes both the empirical error on the labeled data and the inconsistency over labeled and unlabeled data based on the cluster and the manifold assumption. This generalizes the SemiBoost and RegBoost algorithms from binary to multiclass classification using a coding scheme for the multiclass classification by [110] for supervised multiclass boosting. Our proposed method uses the margin on the labeled data, the similarity among labeled and unlabeled data, and the similarity among unlabeled data in the exponential loss function. We give a formal definition of this loss function and derive functions for the weights of classifiers and unlabeled data by minimizing an upper bound on the objective function. We then compare the performance of the algorithm with (a) binary algorithms with smoothness regularization used in an one-versus-all scheme to do multiclass classification (RegBoost and SemiBoost), (b) a boosting multiclass semi-supervised learner without smoothness regularization ([90] and ASSEMBLE [6]), and (c) with supervised learning from the labeled examples using the base learner and using AdaBoost with the base learner to evaluate the effect of using the unlabeled data. The results of the experiments on the benchmark UCI and text classification datasets show that Multi-SemiAdaBoost outperforms the other boosting algorithms and gives the best results.

This chapter is organized as follows: Section 5.2 presents the loss function for semi-supervised learning, Sections 5.3 and 5.4 formalize the setting and the loss function, Section 5.5 derives the weights for the boosting algorithm, the variation of the proposed algorithm is presented in Section 5.6, Sections 5.7 and 5.8 present the experiments and the results, and Section 5.9 draws the main conclusions.

5.2 Loss Function for Semi-Supervised Learning

As addressed in Chapter 2, the goal of semi-supervised learning is to use both labeled and unlabeled to construct a strong classification model. For this, many semi-supervised learning algorithms employ the unlabeled data to regularize the loss function as follows:

\[ \sum_{x_i \in X_l} \ell(y_i, F(x_i)) + \sum_{x_j \in X_u} \ell_u(F(x_j)) \]  

where \( F(.) \) is a binary classifier, \( \ell_u \) shows the penalty term for the unlabeled data, and \( l \) and \( u \) are the labeled and unlabeled data. Typically semi-supervised learning methods vary in the underlying assumptions. Two main assumptions for semi-supervised learning are the cluster and manifold assumptions.

Many graph-based methods, as introduced in Chapter 2, for semi-supervised learning are based on the manifold assumption. These methods build a graph based on the pairwise similarity between examples (labeled and unlabeled). Then, the goal is to estimate a function on the graph. This function should minimize
the loss on labeled examples and smooth on the whole graph. Many graph-based methods use the following loss function:

\[
\ell_u(F(x)) = \sum_{x' \in X_u \atop x \neq x'} S(x,x') \| F(x) - F(x') \|^2_2 \tag{5.2}
\]

where \( S(x, x') \) is pairwise similarity. This penalty term in fact enforces the classifier to predict similar class labels for similar examples such that the inconsistency in whole graph is minimized. Examples of the methods that use the above loss function are Spectral Graph Transducer [51], Label Propagation approach [112], Gaussian process based approach [55], and Manifold Regularization [4]. Two main issues in graph-based approaches are: (1) they are very sensitive to the similarity kernel methods [114], and (2) they were originally designed for binary classification problems. Another challenge in graph-based methods is to find a suitable graph structure for building graphs [113]. We use the above smoothness term in our proposed method in different way for multiclass cases.

Most of the semi-supervised learning algorithms that are based on the cluster assumption are those that use the unlabeled data to regularize the decision boundary. These algorithms usually are extensions of specific base classifier, such as SVM. Examples of these methods are Semi-Supervised SVM [5], Transductive SVM (TSVM) [50], and [19]. A TSVM uses the following regularization term in its loss function:

\[
\ell_u(F(x)) = \max\{0, 1 - |F(x)|\}, x \in u \tag{5.3}
\]

The regularization term (5.3) in the loss function pushes the decision boundary to lie in low density region in feature space by maximizing margin of unlabeled data. However, finding the exact TSVM is a NP-hard problem. Recent approaches cannot solve semi-supervised classification problems with more than a few hundred unlabeled examples. More recently Cristianini [22] uses semi-definite programming (SDP) to solve the TSVM training problem and [104] proposes a multiclass version of the SDP idea. However the proposed methods can handle only a few hundred unlabeled examples and are still computationally expensive in the case of large number of unlabeled data, specially for multiclass problem. The second issue in these methods is that they are not optimal in domains where the SVM is not the best choice for the classifier.

Another kinds of semi-supervised learning algorithms that are based on the cluster assumption are the form of iterative or additive. The main difficulty in these approaches is how to find informative unlabeled data so that using them may lead to improve the decision boundary. As mentioned in the Introduction, the iterative approach replaces the current hypothesis with a new one. Examples of this approach are Self-training [18], Co-training [8], Co-forest [108], and Ensemble-Co-Training [93]. The problem of this approach is that the mislabeled
Additive type of algorithms are of the boosting type. Often boosting is used to improve the classification margin by sampling a subset of high-confidence predictions of the base classifier with their “pseudo-labels”. Then, these newly-labeled examples are used besides the original labeled examples for constructing a new classifier for the next iteration. At each iteration a new classifier is constructed but this does not replace the previous. It then receives a weight and is added to ensemble of classifiers. Examples of this type of algorithm for semi-supervised learning are ASSEMBLE [6] and Marginboost [23]. This approach uses the “pseudo-margin” concept for unlabeled data and forms the following penalty term for unlabeled data:

\[ \ell_u(F(x)) = \sum_{x \in X_u} \exp(-|F(x)|) \] (5.4)

Although the additive approach improves the classification margin, there are two main issues. First, it relies only on the classification confidence to assign “pseudo-labels” and does not provide novel information from the unlabeled examples, such as similarity between examples or marginal distributions. Consequently, the new classifier that was trained on newly-labeled examples, is likely to have the same decision boundary as the first classifier instead of constructing a new one. The reason is that by adapting the decision boundary the poor predictions will not gain higher confidence, but instead the examples with high classification confidence will gain even higher confidence. This issue also has been addressed in [66] and [20]. Another issue with this method is that the proposed loss function was originally designed for binary classification problems and it may not work properly for the multiclass classification problems, which needs to define the multiclass margin and multiclass loss function. To solve the first problem, a recent approach is to use the classification confidence as well as additional information from unlabeled data like pairwise similarities together in order to design a new form of loss function. The goal of the designed loss function is to minimize the inconsistency between the classifier predictions and pairwise similarities. Examples are SemiBoost [66] and RegBoost [20]. They derive criteria to assign confidence to each unlabeled example as well as for the new constructed classifiers. Then a set of high-confidence predictions is selected based on the assigned confidence. Although, experiments show that they outperform the other methods, they are originally designed for binary classification problems.

Most recently several semi-supervised boosting algorithms were developed for Multiclass classification problems, see [99], [81], and [90]. In [99] a new boosting method is used for semi-supervised multiclass learning which uses similarity between predictions and data. It maps labels to \( n \) – dimensional space, but this mapping may not lead to train a classifier that minimizes the margin cost properly. RMSBoost [81] uses the expectation regularization principle for multiclass
Chapter 5. Multiclass Semi-Supervised Boosting

classification problem. The cross entropy between prior probability and the optimized model is used for regularization. Then label priors used in RSMBoost result in a probabilistic way of assigning priors top unlabeled examples based on underlying cluster structures. The extension of supervised Multiclass AdaBoost [110] is used in [90] to design a method for multiclass semi-supervised learning based on “pseudo-margin” approach as in ASSEMBLE, called SemiMultiAdaBoost. There is thus a need for direct multiclass algorithm for semi-supervised learning based on both cluster and manifold assumptions.

Our proposed method, Multi-SemiAdaBoost (MSAB), combines ideas from the methods that we reviewed above to the design of a multiclass semi-supervised boosting algorithm based on manifold and cluster assumption. MSAB employs the advantages of the graph-based methods to minimize the inconsistency between data as well as the ensemble approach to minimize the margin cost to solve directly the multiclass classification problem. We design an exponential loss function for multiclass problems.

5.3 Multiclass Semi-Supervised Learning

In this section, we define the multiclass setting and extend it to multiclass semi-supervised learning.

5.3.1 Multiclass Setting

Suppose we are given a set of training data \((X, Y) = \{(x_i, y_i)\}_{i=1}^n\) where \(x_i \in \mathbb{R}^d, y_i \in \{1, ..., K\}\) denotes a class label, and \((X, Y)\) are drawn from unknown underlying distribution \(D\). Let \(p_k = P_D(y = k | X = x)\). The goal is to learn an optimal classification rule \(F(x) : X \rightarrow \{1, ..., K\}\). An optimal classification rule must achieve the lowest error rate. The error rate for \(F(x)\) is as follows:

\[
E_{X,Y} \mathbb{1}(F(x) \neq Y) = E_X P_D(F(x) \neq Y | X) = 1 - E_X P_D(F(x) = Y | X) = 1 - \sum_{j=1}^k E_X [\mathbb{1}_{F(x) = j} P_D(Y = j | X = x)] \tag{5.5}
\]

where \(\mathbb{1}\) is the indicator function. (5.5) can be rewritten as:

\[
F^*(x) = \arg\min_k \{1 - p_k(x)\} = \arg\max_k p_k(x) \tag{5.6}
\]

which is known as the Bayes decision rule. Implementing the Bayes decision rule is not easy because of estimating \(P_D(Y | X)\). One possible solution is to use the boosting approach that combines many weak classifiers to estimate the Bayes
5.3. Multiclass Semi-Supervised Learning

decision rule and implements the classifier as:

\[ F^*(x) = \arg\max_{k \in \{1,\ldots,K\}} \langle H^*(x), e_k \rangle \]  \hspace{1cm} (5.7)

where \( e_k \) is a standard basis vector. The goal then is to find a predictor \( H^* : X \rightarrow \mathbb{R}^K \) so that it minimizes the risk:

\[ R_M(H) = E_{X,Y}\{L_M[Y, H(x)]\} \simeq \sum_{i=1}^{n} L_M[Y_i, H(x_i)] \]  \hspace{1cm} (5.8)

and

\[ H^*(x) = \arg\min_H R_M(H) \]  \hspace{1cm} (5.9)

where \( L_M[\cdot,\cdot] \) is a multiclass loss function. Note that the class labels \( \pm 1 \) play a significant role in the formulation of loss function of binary classification problems. Therefore, there is a need to find a way to map labels \( \{1,\ldots,K\} \) to \( K\)-dimensional space so that this mapping be consistent with (5.8), which is aimed at finding an optimal predictor. For this, let \( Y_i \in \mathbb{R}^K \) be a \( K\)-dimensional vector with all entries equal to \(-\frac{1}{K-1}\) except a 1 in position \( i \) as a coding method. Each class label \( i \in \{1,\ldots,K\} \) is then mapped into a vector \( Y_i \) that indicates a class label. Each entry \( y_{i,j} \) is thus defined as follows:

\[ y_{i,j} = \begin{cases} 1 & \text{if } i = j \\ \frac{-1}{K-1} & \text{if } i \neq j \end{cases} \]  \hspace{1cm} (5.10)

where \( K \) is the number of classes. Then \( Y \) which is the set of \( K\)-dimensional vectors, will be as follows:

\[
Y = \begin{pmatrix}
(1, -\frac{1}{K-1}, \ldots, -\frac{1}{K-1})^T \\
(\frac{-1}{K-1}, 1, \ldots, -\frac{1}{K-1})^T \\
\vdots \\
(\frac{-1}{K-1}, \ldots, -\frac{1}{K-1}, 1)^T
\end{pmatrix}
\]  \hspace{1cm} (5.11)

where \( Y_i \in Y \) and \( \sum_{j=1}^{K} y_{i,j} = 0 \).

The above notation for multiclass setting was used by [110] and [56] for multiclass AdaBoost and multiclass SVM respectively. Here, we use the aforementioned multiclass notation in an exponential loss function. The resulting empirical risk is:

\[ R_M(H) \simeq \sum_{i=1}^{n} \exp\left(-\frac{1}{K}(Y_i, H(x_i))\right) \]  \hspace{1cm} (5.12)
It has been shown that $R_M(H)$ is Bayes consistent, see [110]. Then

$$\arg\max_k \langle H^*(x), e_k \rangle = \arg\max_k p_k(x)$$

(5.13)

which means that (5.7) implements the Bayes decision rule. This justifies the use of (5.12) as loss function for multiclass classification problem. We use the above notation to formulate the loss function for multiclass semi-supervised learning.

In multiclass semi-supervised learning for the labeled points $X_l = (x_1, x_2, ..., x_l)$ labels $\{1, ..., K\}$ are provided and for the unlabeled points $X_u = (x_{l+1}, x_{l+2}, ..., x_{l+u})$, the labels are not known. We use the coding method in (5.11) to formulate the multiclass semi-supervised learning task.

For our algorithm we need a (symmetric) similarity matrix $S = [S_{i,j}]_{n \times n}$, where $S_{i,j} = S_{j,i}$ is the similarity between the points $x_i$ and $x_j$. $S^u = [S_{i,j}]_{n_l \times n_u}$ denotes the similarity matrix of the labeled and unlabeled data and $S^{uu} = [S_{i,j}]_{n_u \times n_u}$ of the unlabeled data. Our algorithm is a “meta-learner” that uses a supervised learning algorithm as base learner.

We assume that the labeled and unlabeled data are drawn independently from the same data distribution. In applications of semi-supervised learning normally $l \ll u$, where $l$ is the number of labeled data and $u$ is the number of unlabeled data.

5.4 Loss Function for Multiclass Semi-Supervised Learning

The loss functions of the ASSEMBLE and MarginBoost algorithms for semi-supervised boosting algorithms are the sum of empirical loss on the labeled data and on the unlabeled data:

$$C_1 \sum_{x \in X_l} L(y, H(x)) + C_2 \sum_{x \in X_u} L_u(|H(x)|)$$

(5.14)

where $H(.)$ is an ensemble classifier and $L_u(.)$ is the penalty term related to the unlabeled examples, and $C_1$ and $C_2$ are the contribution of the labeled and unlabeled data respectively. The terms $L(.)$ and $L_u(.)$ define the margin and “pseudo-margin” for labeled and unlabeled examples respectively. Since there is no true label for unlabeled examples, “pseudo-labels” are used for them and therefore instead of margin, “pseudo-margin” is defined for the unlabeled examples, like $|H(x)|$ in ASSEMBLE. Considering (5.14) in the boosting setting, the goal of each iteration is to sample the unlabeled examples and assign “pseudo-labels” to them. Then a new component classifier is built based on the new training set which minimizes (5.14). For sampling and labeling the prediction of the boosted classifier $H(x)$ is used.
5.4. Loss Function for Multiclass Semi-Supervised Learning

Unlike the existing semi-supervised boosting methods that only rely on the classifier predictions to labeling, our approach defines the loss for the unlabeled data in terms of consistency, the extent to which examples that are similar have the same label or “pseudo-label”. This explicitly includes the similarity between examples. If examples that are similar, according to the similarity matrix, are assigned different classes then this adds a penalty to the loss function. For this we design an exponential multiclass loss function instead of directly using a binary loss function. This gives three terms for the loss function: (1) the empirical loss on the labeled data, (2) the consistency between labeled and unlabeled data, and (3) the consistency between unlabeled data. Combining these terms results in a minimization problem with the following objective function:

$$F(Y, S, H) = C_1 F_l(Y, H) + C_2 F_{lu}(Y, S_{lu}, H) + C_3 F_{uu}(S_{uu}, H)$$

where $C_1$, $C_2$, and $C_3$ are weights for the three terms. As we showed in (5.13), the proposed multiclass exponential loss function in multiclass AdaBoost effectively implements the Bayes decision rule. We adapt this loss function for multiclass semi-supervised learning. The loss function for labeled data becomes:

$$F_l(Y, H) = \sum_{i=1}^{n_l} \exp\left(-\frac{1}{K} \langle H(x_i), Y_i \rangle \right)$$

where $K$ is the number of classes and $H(.)$ is a multiclass ensemble classifier.

In (5.15), $F_{lu}(Y, S_{lu}, H)$ has been designed to measure the consistency between the similarity information and the classifier predictions on labeled and unlabeled data. For this term we define the multiclass loss function as follows:

$$F_{lu}(Y, S_{lu}, H) = \sum_{i=1}^{n_l} \sum_{j=1}^{n_u} S_{lu}(x_i, x_j) \exp\left(-\frac{1}{K-1} \langle H(x_j), Y_i \rangle \right)$$

where $n_l$ and $n_u$ are the number of labeled and unlabeled data respectively.

The third term of (5.15) measures the consistency between the unlabeled examples in terms of the similarity information and the classifier predictions. We use the harmonic function to define $F_{uu}(S_{uu}, H)$. This is a popular way to define the consistency in many graph-based methods, for example [115].

$$F_{uu}(S_{uu}, H) = \sum_{i,j \in n_u} S_{uu}(x_i, x_j) \exp\left(\frac{1}{K-1} \langle H(x_i) - H(x_j), \overrightarrow{1} \rangle \right)$$

Combining the above three terms leads the following optimization problem:

$$\text{minimize } F(Y, S, H)$$

$$\text{subject to } H_1(x) + ... + H_k(x) = 0, x \in \mathcal{X}.$$
Chapter 5. Multiclass Semi-Supervised Boosting

where $X$ is the training set. Here, we briefly show that $F(Y, S, H)$ maintains the convexity. The first and second term of (5.15) are in terms of exponential function and dot product. Since $\exp(.)$ and its argument are convex, and $S_{i,j}$ is positive semi-definite. Then (5.16) and (5.17) are convex. In (5.18) using power series result in $cosh(.)$ function, which is a convex function. Therefore, (5.15) is convex. The convex optimization problem can be effectively solved by numerical methods. Another valuable point in the loss function (5.19) is that on one hand it uses the boosting approach to exploit the cluster assumption as in many boosting methods and on the other hand it employs (5.18) as regularization term for unlabeled data, which is used in many graph-based methods, such as LapSVM [4], for manifold regularization. Therefore, the proposed method effectively exploits both the cluster and the manifold assumption. The relationship between the manifold regularization and (5.18) is shown in Appendix A.

5.5 Multiclass Semi-Supervised Boosting

In this section we derive the Multi-SemiAdaBoost algorithm, which has the form of a boosting algorithm. Given a set of labeled and unlabeled data, a similarity metric, and a supervised (multiclass) learning algorithm, Multi-SemiAdaBoost effectively minimizes the objective function (5.15). In Multi-SemiAdaBoost, a series of classifiers is constructed by applying a supervised base learner to varying sets of training data. Each classifier gets a weight. When a termination condition is reached, the weighted combination of the classifiers becomes the final hypothesis. The algorithm starts with the labeled data and at each iteration adds some unlabeled data that received a “pseudo-label” from the classification model constructed in the previous iteration. For this algorithm we need to find the weights for the classifiers and the unlabeled data.

Suppose that $H^t(x) : X \rightarrow R^k$ defines the linear combination of classification models after $t - 1$ iterations. Then, at the $t$-th iteration, $H^t(x)$ is computed as:

$$H^t(x) = H^{t-1}(x) + \beta^t h^t(x)$$  (5.20)

where $\beta^t \in R$ is the weight of the base classifier $h^t(x)$, and $h^t(x)$ is determined by $P(x)$, which is a multiclass base learner. Let also $h^t(x)$ denote a classifier as follows:

$$\forall x \in R^p \ h^t(x) : X \rightarrow Y$$  (5.21)

where $Y$ was defined in (5.11). The goal of each iteration is to find a new component classifier $h^t(x)$ and its weight $\beta^t$ that minimizes the objective function (5.19) and computes the weights for the data.

Two main approaches to solve the optimization problem (5.19) are: gradient descent [20] and bound optimization [66]. A difficulty of the gradient descent method is to determine the step size at each iteration. We therefore use the
5.5. Multiclass Semi-Supervised Boosting

bound optimization approach to derive the boosting algorithm for (5.19), which automatically determines the step size at each iteration of boosting process.

At the $t$-th iteration, the optimization problem (5.15) is derived by substituting (5.20) in (5.19):

$$F = \arg\min_{h^t(x), \beta^t} C_1 \sum_{i=1}^{n_l} \exp(-\frac{1}{K} \langle H^{t-1}(x_i) + \beta^t h^t(x_i), Y_i \rangle)$$

$$+ C_2 \sum_{i=1}^{n_l} \sum_{j=1}^{n_u} S^{\text{lu}}(x_i, x_j) \exp(-\frac{1}{K-1} \langle H^{t-1}(x_j) + \beta^t h^t(x_j), Y_i \rangle)$$

$$+ C_3 \left( \sum_{i, j \in \mathcal{L}} S^{\text{uu}}(x_i, x_j) \exp\left(\frac{1}{K-1} \sum_{k \in \mathcal{L}} (\langle H^{t-1}(x_i) + \beta^t h^t(x_i) \rangle - (H^{t-1}(x_j) + \beta^t h^t(x_j)), e_k \rangle)\right) \right)$$

(5.22)

To solve (5.22), we first simplify it and then with reformulation we derive a criterion to assign weights to data. Finally, we propose a criterion to assign class labels and $\beta$ as weight for the new classifier $h^t(x)$. For this, we use a standard basis vector $e_k$ to represent class membership. In Appendix A, it is shown that minimizing (5.22) is equivalent to minimizing the following optimization problem:

$$F_1 = C_1 \sum_{i=1}^{n_l} \exp(-\frac{1}{K} \langle H^{t-1}_i + \beta^t h^t_i, Y_i \rangle)$$

$$+ C_2 \sum_{i=1}^{n_l} \sum_{j=1}^{n_u} S^{\text{lu}}(x_i, x_j) \exp\left(\frac{-1}{K-1} \langle H^{t-1}_i, e_k \rangle \right)$$

$$\exp\left(\frac{-\beta^t}{K-1} \langle h^t_i, e_k \rangle \right) \delta(Y_i, e_k)$$

$$+ C_3 \sum_{i, j \in \mathcal{L}} S^{\text{uu}}(x_i, x_j) \exp\left(\frac{1}{K-1} \langle H^{t-1}_i, H^{t-1}_j, e_k \rangle \right)$$

$$\exp\left(\frac{\beta^t}{K-1} \langle h^t_i - h^t_j, e_k \rangle \right)$$

(5.23)

Subject to:

$$H^{t}_1(x_i) + \ldots + H^{t}_k(x_i) = 0$$

where $\delta(Y_i, e_k)$ is:

$$\delta(Y_i, e_k) = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{if } i \neq k \end{cases}$$

(5.24)

where $F_1$ is an upper bound on $F$ and minimizing $F_1$ will also minimize $F$. To simplify the notation we write $H^{t-1}_i$ for $H^{t-1}(x_i)$ and $h^t_i$ for $h^t(x_i)$. For now
assume that \( C_1 = C_2 = C_3 = 1 \). As mentioned before, (5.23) is a nonlinear function. To solve this problem and also to find a criterion for assigning “pseudo-labels” to the unlabeled examples, in Appendix A, we show that (5.23) can be reformulated as (5.25):

\[
F_2 = \sum_{i=1}^{n_l} W_i \exp\left(\frac{-\beta^t}{K} \langle h^t_i, Y_i \rangle \right) + \sum_{i=1}^{n_u} \sum_{k \in l} P_{i,k} \exp\left(\frac{-\beta^t}{K - 1} \langle h^t_i, e_k \rangle \right)
\]

where

\[
W_i = \exp\left(\frac{-1}{K} \langle H^{t-1}_i, Y_i \rangle \right)
\]

and

\[
P_{i,k} = \sum_{j=1}^{n_l} S^{lu}(x_i, x_j) e^{\left(\frac{1}{K - 1} \langle H^{t-1}_j, e_k \rangle \right)} \delta(Y_i, e_k)
+ \sum_{j=1}^{n_u} S^{uu}(x_i, x_j) e^{\left(\frac{1}{K - 1} \langle H^{t-1}_j - H^{t-1}_i, e_k \rangle \right)}
\]

In the optimization problem (5.25), \( P_{i,k} \) used for weighting the unlabeled examples and \( W_i \) is used for weighting labeled data. According to these criteria the algorithm decides whether to select a sample for the new training set or not. The expression in (5.25) is in terms of \( \beta^t \) and \( h^t(x) \) and hence make it difficult to solve. In Appendix A, we show the derivative of (5.25) which leads to the optimal class value and weights for the unlabeled data as well as weights for the classifiers. Then,

\[
\hat{Y}_i = \arg \max_k (P_{i,k})
\]

and the weight for the example will be \( w'_i = \max |P_{i,k}| \). Meanwhile the value for \( \beta \) that minimizes \( F_2 \) is:

\[
\beta = \frac{(K - 1)^2}{K} \left( \log(K - 1) + \log \left( \frac{\sum_{i \in n_u} \sum_{k \in l} P_{i,k} \delta(h^t_i, e_k, P_i = k) + \sum_{i \in n_l} W_i}{\sum_{i \in n_u} \sum_{k \in l} P_{i,k} \delta(h^t_i, e_k, P_i \neq k) + \sum_{i \in n_l} W_i} \right) \right)
\]

where \( P_i \equiv P(x_i) \) is a multiclass base learner as in (5.20).

Now, let \( \epsilon^t \) denote the weighted error made by the classifier \( h^t \). We define \( \epsilon^t \) as follows:

\[
\epsilon^t = \frac{\sum_{i \in n_u} \sum_{k \in l} P_{i,k} \delta(h_i, e_k, P_i \neq k) + \sum_{i \in n_l} W_i}{\sum_{i \in n_u} \sum_{k \in l} P_{i,k} + \sum_{i \in n_l} W_i}
\]
5.5. Multiclass Semi-Supervised Boosting

Replacing (5.30) in (5.29) leads to the following $\beta^t$:

$$\beta^t = \frac{(K - 1)^2}{K} \left( \log(K - 1) + \log\left(\frac{1 - \epsilon^t}{\epsilon^t}\right) \right)$$

(5.31)

which is a generalization of the weighting factor of multiclass AdaBoost [110]. If the classification problem is binary then $k = 2$, we obtain the formula for $\beta$ that is used in AdaBoost [35]. Our formulation can thus be seen as a generalization of AdaBoost to multiclass semi-supervised learning.

5.5.1 The Multi-SemiAdaBoost Algorithm

Based on the previous discussion, we can provide the details of the algorithm. In Algorithm 8, first the values of $P_{i,k}$ and $W_i$ are computed for each unlabeled

**Algorithm 8 Multi-SemiAdaBoost**

Initialize: $L$, $U$, $S$, $H^0(x) = 0$; $L$: Labeled data; $U$: Unlabeled data; $S$: Similarity Matrix; $H^0(x)$: Ensemble of Classifiers; $t \leftarrow 1$;

while ($\beta^t > 0$) and ($t < M$) do // $M$ is the number of iteration

for each $x_i \in L$ do

- Compute $W_i$ for labeled example $x_i$ based on (5.26)

for each $x_i \in U$ do

- Compute $P_{i,k}$ for unlabeled example $x_i$ based on the pairwise similarity and classifier predictions using (5.27)

- Assign pseudo-labels to unlabeled examples based on (5.28)

- Normalize the weights of labeled and unlabeled examples

- Sample a set of high-confidence examples from labeled and unlabeled examples

- Build a new weighted classifier $h^t(x)$ based on the newly-labeled and original labeled examples

- Compute the weights and $\beta^t$ for each new classifier $h^t(x)$ using (5.29)

- Update $H^t \leftarrow H^{t-1} + \beta^t h^t$

- $t \leftarrow t + 1$

end while

Output: Generate final hypothesis based on the weights and classifiers

and labeled example. In order to compute the value of $P_{i,k}$, we use the similarity information among data and the classifier predictions and $W_i$ is computed as in multiclass AdaBoost (SAMME). Then (5.28) is used to assign a “pseudo-label” to an unlabeled example and the value of (5.28) becomes the weight. In the next step, a set of high-confidence newly-labeled data besides the labeled data are used based on the weights for training a new classifier, called $h^t(x)$. The algorithm then uses the value of $P_{i,k}$ as weight to sample data which will lead to a decrease
of the value of the objective function. This new set is consistent with classifier predictions and similarity between examples. As can be seen in Algorithm 8, a new classifier is built at each iteration of the boosting process. Then the weight of \( h_t(x) \), called \( \beta_t \), is computed for each new classifier. The boosting process is repeated until it reaches a stopping condition. After finishing, the final hypothesis is the weighted combination of the generated classifiers. Although, it has been empirically determined that using a fixed number of classifiers, (around 20) is best for AdaBoost \[37\], we use \( \beta_t \leq 0 \) as a stopping criterion. This is analogous to using \( \epsilon_t \geq \frac{1}{2} \) for AdaBoost. Optionally, for pragmatic reasons the algorithm can of course be given a threshold on the number of iterations.

As can be seen in (5.27), the value of \( P_{ik} \) depends on several factors: (i) the value will be large if \( x_i \) cannot be classified confidently. This means that \( H_{t-1}^{-1} \cdot c_k \) is small and one of its close neighbors is labeled. The first term of (5.27) refers to this; (ii) the example is close to some unlabeled examples that are confidently classified, which means that large similarity and \( H_t^{-1} \) for the unlabeled example \( x_j \). This point comes from the second term of (5.27). This shows the role of similarity information for sample selection in Multi-SemiAdaBoost and makes it different from the other methods, such as ASSEMBLE, MarginBoost and SemiMultiAdaBoost \[90\].

### 5.6 Variations of Multi-SemiAdaBoost

If there are no unlabeled data then the loss function in (5.15) simplifies to the margin cost on the labeled data. However, if supervised learning on the labeled data produces poor results then it may be useful to not include it in the loss function. This may improve the results. The resulting loss function then consists of two terms: the consistency between the labeled and unlabeled data and between the unlabeled data. With small changes \( F(Y, S, H) \) is then as follows:

\[
F(Y, S, H) = C_1 F_{lu}(Y, S^{lu}, H) + C_2 F_{uu}(S^{uu}, H) \tag{5.32}
\]

where \( F_{lu}(Y, S^{lu}, H) \)

\[
F_{lu}(Y, S^{lu}, H) = \sum_{i=1}^{n_l} \sum_{j=1}^{n_u} S^{lu}(x_i, x_j) \exp\left(-\frac{1}{K} \langle H(x_j), Y_i \rangle \right) \tag{5.33}
\]

and \( F_{uu}(S^{uu}, H) \) as:

\[
F_{uu}(S^{uu}, H) = \sum_{i,j \in n_u} S^{uu}(x_i, x_j) \exp\left(\frac{1}{K} \langle H(x_i) - H(x_j), 1 \rangle \right) \tag{5.34}
\]
To solve (5.32), we used the same analysis that we did for Multi-SemiAdaBoost, more detail in Appendix B. The expressions that are required for the resulting algorithm, which we call Multiclass SemiBoost, are:

\[
P_{i,k} = \sum_{j=1}^{n_l} S_{lu}(x_i, x_j)e^{\left(\frac{1}{\pi}(H_{i}^{t-1},e_k)\right)}\delta(y_i, e_k) + \sum_{j=1}^{n_u} S_{uu}(x_i, x_j)e^{\left(\frac{1}{\pi}(H_{j}^{t-1}-H_{i}^{t-1},e_k)\right)}e^{\frac{1}{\pi}}
\]

and then,

\[
\hat{Y}_i = \arg \max_k (P_{i,k})
\]

and the weight for the example will be \(w'_i = \max |P_{i,k}|\). Meanwhile the value for \(\beta\) is:

\[
\beta^t = (K - 1)\left(\log(K - 1) + \log\left(\frac{1 - \epsilon^t}{\epsilon^t}\right)\right)
\]

where \(P_{i,k}\) is a criterion to assign the “pseudo-labels” to the unlabeled examples and \(\beta^t\) is the weight of each new component classifier \(h^t(x)\).

**Algorithm 9 Multiclass SemiBoost**

Initialize: L, U, S, \(H^0(x) = 0\); L: Labeled data; U: Unlabeled data;
S: Similarity Matrix; \(H^0(x)\);
Ensemble of Classifiers; \(t \leftarrow 1; // M\) is the number of iteration

while (\(\beta^t > 0\)) and (\(t < M\)) do

for each \(x_i \in U\) do

- Compute \(P_{i,k}\) for unlabeled example \(x_i\) based on the pairwise similarity and classifier predictions using (5.35)
- Assign pseudo-labels for unlabeled examples based on equation (5.36)
- Normalize the weights of unlabeled examples
- Sample a set of high-confidence examples from the unlabeled examples based on their weights
- Build a new classifier \(h^t(x)\) based on the newly-labeled and labeled examples
- Compute the weights and \(\beta^t\) for each new classifier \(h^t(x)\) using (5.37)
- Update \(H^t \leftarrow H^{t-1} + \beta^t h^t\)
- \(t \leftarrow t + 1\)

end while

Output: Generate final hypothesis based on the weights and classifiers

The Multiclass SemiBoost algorithm is the same as Multi-SemiAdaBoost except for the labeled data, which is not considered in this algorithm. Algorithm 9 shows the detail. As shown, at each iteration a new classifier is built and the weight of \(h^t(x)\), called \(\beta^t\), is computed for each new classifier based on
Chapter 5. Multiclass Semi-Supervised Boosting

(5.37), which is different from the weight of each component classifier in Multi-SemiAdaBoost. Assigning weights for the unlabeled data is now based on the $P_{i,k}$ in (5.35).

5.7 Experiments

In the experiments we compare Multi-SemiAdaBoost (MSAB) with several other algorithms. One comparison is with using the base learner on the labeled data only and a second is with (multiclass) AdaBoost (SAMME [110]) using the same base learner. The purpose is to evaluate if MSAB exploits the information in the unlabeled data. We also compare MSAB with a version that does not include the margin for the labeled data in the objective function, Multiclass SemiBoost (MCSB). The effect of the labeled data depends on the quality of the base classifier and we would like to evaluate its importance. We include comparisons with the state-of-the-art algorithms for semi-supervised boosting, in particular ASSEMBLE, SemiMultiAdaBoost [90], RegBoost [20], and SemiBoost [66]. Like MSAB, SemiBoost and RegBoost use smoothness regularization but they are limited to binary classification. For comparison, we use the one-vs-all method to handle the multiclass classification problem with RegBoost and SemiBoost. In our experiments, we use the WEKA implementation of the classifiers with default parameter settings [42].

Three components in Multi-SemiAdaBoost and Multiclass SemiBoost are: (i) the similarity between examples, (ii) sample from the unlabeled examples, and (iii) stopping condition for boosting process.

5.7.1 Similarity Function

There are different distance-based approaches to compute the similarity between the data. In this chapter we employ the Radial Basis Function (RBF) which is used effectively as similarity metric in many domains. For two examples $x_i$ and $x_j$ the RBF similarity is computed as:

$$S(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma^2}\right)$$  \hspace{1cm} (5.38)

where $\sigma$ is the scale parameter which plays a major role in the performance of the kernel, and should be carefully tuned to the problem [114].

5.7.2 Sampling Data for Labeling

In the sampling process only the high-confidence data points must be selected for training. Finding the best selection is a difficult problem [93]. On one hand, selecting a small set of newly-labeled examples might lead to slow convergence,
and on the other hand, selecting a large set of newly-labeled examples may include some poor examples. One possible solution for that is to use a threshold or even a fixed number which is optimized through the training process. Sampling data for labeling is based on the following measure for confidence:

\[
P_d(x_i) = \frac{\hat{Y}_i - \max\{Y_{i,k} | Y_{i,k} \neq \hat{Y}_i, \ k = 1, ..., K\}}{\sum_{i=1}^{n_u}(\hat{Y}_i - \max\{Y_{i,k} | Y_{i,k} \neq \hat{Y}_i, \ k = 1, ..., K\})}
\] (5.39)

where \(\hat{Y}_i\) is computed from (5.28). \(P_d(x_i)\) is viewed as the probability distribution of classes of the example \(x_i\), which amounts to a measure of confidence. For the labeled data we use \(W_i\) in (5.26) as weight:

\[
P_d(x_i) = \frac{W_i}{\sum_{i=1}^{n_l} W_i}
\] (5.40)

where \(x_i \in L\). We use learning from weighed examples as in AdaBoost and at each iteration we select the top 15% of the unlabeled data based on the weights and add them to the training set.

Another component in the proposed methods is to identify the stopping conditions. As mentioned in Section 5.5.1, we use both \(\beta^t \leq 0\) and a fix number of iterations as stopping conditions.

### 5.7.3 Supervised Base Learner

As mentioned earlier, we assume that the base learner as a block box in the algorithm. It means that the proposed algorithm does not need to know the inner process of the base learner. However, the performance of the method depends on the base learner. We experiment with the Decision Tree learner (J48, the Java implementation of C4.5 decision tree classifier), Naive Bayes, and Support Vector Machine (SVM) with default setting (polynomial kernel and sequential minimal optimization algorithm) as base classifiers.

### 5.7.4 UCI datasets

We use the UCI datasets for assessing the proposed algorithms. Table 5.1 summarizes the specification of 16 benchmark datasets from the UCI data repository [32] which are used in our experiments.

### 5.7.5 Setup of the experiments

For each dataset, 30 percent of the data are kept as test set, and the rest is used as training data. Training data in each experiment are first partitioned into 90 percent unlabeled data and 10 percent labeled data, keeping the class proportions in all sets similar to the original data set. We run each experiment 10 times with
Table 5.1: Overview of Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Samples</th>
<th>#Attributes</th>
<th># Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Balance</td>
<td>625</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Car</td>
<td>1728</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>Cmc</td>
<td>1473</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>Dermatology</td>
<td>366</td>
<td>34</td>
<td>6</td>
</tr>
<tr>
<td>Diabetes</td>
<td>768</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>Ecoli</td>
<td>336</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>Glass</td>
<td>214</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>Iris</td>
<td>150</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Liver</td>
<td>345</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>Optdigit</td>
<td>1409</td>
<td>64</td>
<td>10</td>
</tr>
<tr>
<td>Sonar</td>
<td>208</td>
<td>60</td>
<td>2</td>
</tr>
<tr>
<td>Soybean</td>
<td>686</td>
<td>35</td>
<td>19</td>
</tr>
<tr>
<td>Vowel</td>
<td>990</td>
<td>14</td>
<td>11</td>
</tr>
<tr>
<td>Wave</td>
<td>5000</td>
<td>21</td>
<td>3</td>
</tr>
<tr>
<td>Wine</td>
<td>178</td>
<td>13</td>
<td>3</td>
</tr>
<tr>
<td>Zoo</td>
<td>101</td>
<td>17</td>
<td>7</td>
</tr>
</tbody>
</table>

different subsets of training and testing data. The results reported refer to the test set.

5.8 Results

Tables 5.2, 5.3, and 5.4 give the results of all experiments. The second and third columns in these tables give the performance of supervised multiclass base classifiers (DT for Decision Trees, NB for Naive Bayes, and SVM for Support Vector Machine) and multiclass AdaBoost meta classifier, SAMME [110], using the same classifiers as base learner. The columns ASML, SMBoost, RBoost, and SBoost show the performance of four semi-supervised boosting algorithms ASSEMBLE, SemiMultiAdaBoost, RegBoost and SemiBoost respectively.

MSAB and Supervised Learning

The results in Tables 5.2, 5.3, and 5.4 show that MSAB significantly improves the performance of supervised base classifiers for nearly all the datasets. Using the statistical t-test, we observed that MSAB improves the performance of J48 and Naive Bayes base classifiers on 16 out of 16 and SVM base classifier on 15 out of 16 datasets. The results show that MSAB is also better than multiclass AdaBoost meta classifier trained using only labeled data. We also observe that the classification models generated by using MSAB are relatively more stable than J48 base classifier because of lower standard deviation in classification accuracy.
Table 5.2: The classification accuracy and standard deviation of different algorithms with 10% labeled examples and J48 as base learner

<table>
<thead>
<tr>
<th>DataSets</th>
<th>Supervised Learning</th>
<th>Semi-Supervised Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DT</td>
<td>SAMME</td>
</tr>
<tr>
<td>Balance</td>
<td>70.72</td>
<td>74.75</td>
</tr>
<tr>
<td>Car</td>
<td>76.93</td>
<td>78.34</td>
</tr>
<tr>
<td></td>
<td>1.6</td>
<td>2</td>
</tr>
<tr>
<td>Cmc</td>
<td>42.03</td>
<td>44.89</td>
</tr>
<tr>
<td></td>
<td>3.8</td>
<td>4.2</td>
</tr>
<tr>
<td>Dermatology</td>
<td>73</td>
<td>76.99</td>
</tr>
<tr>
<td>Diabetes</td>
<td>68.81</td>
<td>70.97</td>
</tr>
<tr>
<td>Ecoli</td>
<td>65.88</td>
<td>66.82</td>
</tr>
<tr>
<td>Glass</td>
<td>49.79</td>
<td>50.94</td>
</tr>
<tr>
<td>Iris</td>
<td>71.13</td>
<td>71.13</td>
</tr>
<tr>
<td></td>
<td>8.9</td>
<td>8.9</td>
</tr>
<tr>
<td>Liver</td>
<td>54.86</td>
<td>56.93</td>
</tr>
<tr>
<td></td>
<td>7.5</td>
<td>7.6</td>
</tr>
<tr>
<td></td>
<td>2.4</td>
<td>3.4</td>
</tr>
<tr>
<td>Optdigit</td>
<td>63.26</td>
<td>76.52</td>
</tr>
<tr>
<td>Sonar</td>
<td>60.08</td>
<td>59.87</td>
</tr>
<tr>
<td>Soybean</td>
<td>51.5</td>
<td>47.83</td>
</tr>
<tr>
<td>Vowel</td>
<td>35.92</td>
<td>42.57</td>
</tr>
<tr>
<td>Wave</td>
<td>70.1</td>
<td>76.04</td>
</tr>
<tr>
<td>Wine</td>
<td>67.36</td>
<td>67.36</td>
</tr>
<tr>
<td>Zoo</td>
<td>66.19</td>
<td>56.66</td>
</tr>
<tr>
<td></td>
<td>5.2</td>
<td>8.1</td>
</tr>
</tbody>
</table>

MSAB and Semi-Supervised Boosting Algorithms

Table 5.2, 5.3, 5.4 show that MSAB gives better results than all four semi-supervised algorithms ASSEMBLE, SemiMultiAdaBoost, RegBoost, and SemiBoost on 14 (for J48), 13 (for Naive Bayes), or 16 (for SVM) out of 16 datasets. The improvement of MSAB in most of the used datasets are significant and it outperforms the ASSEMBLE and SMBoost on 15 out of 16 datasets, when the base classifier is J48. The same results can be seen with the other base classifiers.
Table 5.3: The classification accuracy and standard deviation of different algorithms with 10% labeled examples and Naive Bayes as base learner

<table>
<thead>
<tr>
<th>DataSets</th>
<th>Supervised Learning</th>
<th>Semi-Supervised Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NB</td>
<td>SAMME</td>
</tr>
<tr>
<td>Balance</td>
<td>75.88</td>
<td>76.71</td>
</tr>
<tr>
<td></td>
<td>3.5</td>
<td>2</td>
</tr>
<tr>
<td>Car</td>
<td>77.5</td>
<td>78.51</td>
</tr>
<tr>
<td></td>
<td>1.9</td>
<td>0.5</td>
</tr>
<tr>
<td>Cmc</td>
<td>46.98</td>
<td>48.05</td>
</tr>
<tr>
<td></td>
<td>1.4</td>
<td>1.3</td>
</tr>
<tr>
<td>Dermatology</td>
<td>81.52</td>
<td>81.52</td>
</tr>
<tr>
<td></td>
<td>7.4</td>
<td>7.4</td>
</tr>
<tr>
<td>Diabetes</td>
<td>73.45</td>
<td>73.05</td>
</tr>
<tr>
<td></td>
<td>1.1</td>
<td>1.5</td>
</tr>
<tr>
<td>Ecoli</td>
<td>81.68</td>
<td>78.69</td>
</tr>
<tr>
<td></td>
<td>3.3</td>
<td>6.3</td>
</tr>
<tr>
<td>Glass</td>
<td>50</td>
<td>50.21</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>6.7</td>
</tr>
<tr>
<td>Iris</td>
<td>81.25</td>
<td>80.95</td>
</tr>
<tr>
<td></td>
<td>5.5</td>
<td>6.6</td>
</tr>
<tr>
<td>Liver</td>
<td>55.39</td>
<td>55.04</td>
</tr>
<tr>
<td></td>
<td>5.5</td>
<td>7.2</td>
</tr>
<tr>
<td>Optdigit</td>
<td>72.04</td>
<td>70.57</td>
</tr>
<tr>
<td></td>
<td>1.9</td>
<td>1.7</td>
</tr>
<tr>
<td>Sonar</td>
<td>64.41</td>
<td>63.23</td>
</tr>
<tr>
<td></td>
<td>7.1</td>
<td>5.1</td>
</tr>
<tr>
<td>Soybean</td>
<td>70</td>
<td>68.44</td>
</tr>
<tr>
<td></td>
<td>1.4</td>
<td>7.1</td>
</tr>
<tr>
<td>Vowel</td>
<td>47.52</td>
<td>48.02</td>
</tr>
<tr>
<td></td>
<td>3.4</td>
<td>2</td>
</tr>
<tr>
<td>Wave</td>
<td>80.44</td>
<td>80.91</td>
</tr>
<tr>
<td></td>
<td>1.4</td>
<td>1.9</td>
</tr>
<tr>
<td>Wine</td>
<td>84.56</td>
<td>84.56</td>
</tr>
<tr>
<td></td>
<td>5.8</td>
<td>5.8</td>
</tr>
<tr>
<td>Zoo</td>
<td>87.77</td>
<td>87.77</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>
## 5.8. Results

Table 5.4: The classification accuracy and standard deviation of different algorithms with 10% labeled examples and SVM as base learner

<table>
<thead>
<tr>
<th>DataSets</th>
<th>Supervised Learning</th>
<th>Semi-Supervised Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SVM</td>
<td>SAMME</td>
</tr>
<tr>
<td>Balance</td>
<td>82.66</td>
<td>81.69</td>
</tr>
<tr>
<td>Car</td>
<td>80.92</td>
<td><strong>83.83</strong></td>
</tr>
<tr>
<td>Cmc</td>
<td>45.12</td>
<td>45.49</td>
</tr>
<tr>
<td>Dermatology</td>
<td>86.54</td>
<td>86.84</td>
</tr>
<tr>
<td>Diabetes</td>
<td>71.3</td>
<td>72.56</td>
</tr>
<tr>
<td>Ecoli</td>
<td>79.2</td>
<td>82.35</td>
</tr>
<tr>
<td>Glass</td>
<td>47.64</td>
<td>48.23</td>
</tr>
<tr>
<td>Iris</td>
<td>84.32</td>
<td>86.45</td>
</tr>
<tr>
<td>Liver</td>
<td>57.66</td>
<td>55.01</td>
</tr>
<tr>
<td>Optdigit</td>
<td>89.92</td>
<td>89.92</td>
</tr>
<tr>
<td>Sonar</td>
<td>65.88</td>
<td>65.88</td>
</tr>
<tr>
<td>Soybean</td>
<td>72.69</td>
<td>72.69</td>
</tr>
<tr>
<td>Vowel</td>
<td>47.52</td>
<td>48.02</td>
</tr>
<tr>
<td>Wave</td>
<td>80.44</td>
<td>80.91</td>
</tr>
<tr>
<td>Wine</td>
<td>88.81</td>
<td>88.81</td>
</tr>
<tr>
<td>Zoo</td>
<td><strong>95.83</strong></td>
<td><strong>95.83</strong></td>
</tr>
</tbody>
</table>

**MSAB and Multiclass SemiBoost**

As can be seen in Table 5.2, Multiclass SemiBoost performs better than the other semi-supervised boosting algorithms on 15 out of 16 datasets. Table 5.4 shows that Multiclass SemiBoost gives better performance than the other four algorithms on 12 out of 16 datasets. Furthermore, MSAB outperforms Multiclass SemiBoost in 9 out of 16 datasets for both base learners with 2 ties for J48 and 1 for Naive Bayes. The results suggest that when boosting the base classifier works (for example for the Naive Bayes classifier on the Ecoli or Diabetes dataset) then MSAB outperforms Multiclass SemiBoost, which is what we would expect. The same result is shown in Table 5.4.
5.8.1 Different Proportions of Unlabeled Examples

To study the sensitivity of MSAB to the number of labeled data, we run a set of experiments with different proportions of labeled data which vary from 5% to 40%. We compare supervised base classifiers to supervised AdaBoost meta classifier and MSAB. We expect that the difference between supervised algorithms and MSAB algorithm decreases when more labeled data are available. Similar to the previous experiments, we use a separate test set to evaluate the performance. Figure 5.1 shows the performance on 5 datasets, Iris, Glass, Sonar, Soybean, and Wine, with different base learners.

It is observed that with the additional unlabeled data and only 5% labeled data, MSAB algorithm significantly improves the performance of the base classifier. Consistent with our hypothesis, MSAB improves the performance of the base classifier and performs better than supervised AdaBoost with the same base classifier.

5.8.2 Discussion

There are datasets where the proposed algorithms may not significantly improve the performance of the base classifiers. In these cases the supervised algorithm outperforms all the semi-supervised algorithms, for example in Table 5.3 the AdaBoost meta classifier performs the same as the proposed methods on Car, Cmc, and Diabetes datasets and outperforms the other methods. This kind of result emphasizes that the unlabeled examples do not guarantee that they always are useful and improve the performance. Comparing the results of Tables, we observed that almost in all cases MSAB and Multiclass SemiBoost improve the performance of the base learner and in most cases they outperform the other compared boosting algorithms in this study.

5.8.3 Convergence of Multi-SemiAdaBoost

In this section, we empirically show the convergence of Multi-SemiAdaBoost algorithm on two datasets, Wine and Optdigit. We use different base classifiers in this experiment, J48 is used for Wine dataset and SVM for Optdigit dataset. As shown in Figure 5.2.a and 5.2.b both the value of $\beta$ and objective function are decreasing through the boosting process for Wine dataset, which is what we would expect. For the first 10 iterations, the value of $\beta$ and objective function fall rapidly and after that they are decreasing slowly. Figure 5.2.c and 5.2.d show the same results for Optdigit dataset, when the base learner is SVM.
5.8. Results

Figure 5.1: Average Performance of MSAB with increasing proportions of labeled data on different datasets
5.8.4 Sensitivity to the Scale Parameter

As shown in similarity function (5.38), there is a parameter, Sigma, which needs to be tuned in the experiments. Selecting the value of Sigma is a difficult task in many graph-based methods [114] and several heuristic methods have been proposed to compute its value. For this, we setup a set of experiments to show the sensitivity of the proposed method to the scale parameter. The results of the experiments show that Multi-SemiAdaBoost is relatively stable with respect to the Sigma value. Figure 5.3 shows the results on two datasets with two different base learning algorithms.

5.8.5 Sensitivity to the Size of the Newly-Labeled Data

In section 5.7.2 we presented a criterion to assign confidence for predictions. In this section we perform a set of experiments to study the performance of MSAB at the size of the sampled unlabeled examples. The sample size varies from 5% to 100% in the experiments. Figure 5.4 shows the performance on two datasets with two different base classifiers. It is shown that sampling 5%-20% of the unlabeled
5.9 Conclusion and Discussion

Figure 5.3: Performance of base learning algorithms J48 and SVM with 10% labeled examples on Iris and Optdigit datasets, with increasing value of the Sigma.

Two main issues in semi-supervised boosting methods are: (1) how to sample from the unlabeled data and, (2) how to assign the high-confidence “pseudo-labels” to them. To solve these problems for a multiclass classification problem, we propose a boosting method which is based on the main semi-supervised assumptions, the manifold and the cluster. We define a novel multiclass loss function consisting of the margin cost on labeled data and the regularization on labeled and unlabeled data. For the regularization terms as in graph-based method, we define the inconsistency between data using similarity information. Then we combine the similarity information and classifier predictions to derive a criterion to assign the “pseudo-labels” and the weights to the unlabeled data and another criterion to weight the labeled data as in multiclass AdaBoost. Based on the criteria a set of high-confidence newly-labeled data and original labeled data is used to construct a new component classifier, which minimizes the objective function. The final
The classification model is formed by the combination of the generated classifiers. Experiments show that in almost all cases Multi-SemiAdaBoost and Multiclass SemiBoost significantly improve the performance of the base learner and in most cases they outperform the other boosting algorithms in this comparison.

An open problem for future work is how to efficiently find and tune a good similarity function. Another issue is the mapping for the multiclass problem to $K$-dimensional space. The mapping that is used here is not the only possibility and it seems interesting to exploit other coding schemes.