Communications for Statistical Applications and Methods
2022, Vol. 29, No. 2, 193–202

https://doi.org/10.29220/CSAM.2022.29.2.193
Print ISSN 2287-7843 / Online ISSN 2383-4757

The use of support vector machines in semi-supervised classification

Hyunjoo Bae[a], Hyungwoo Kim[a], Seung Jun Shin[1,a]

[a]Department of Statistics, Korea University, Korea

Abstract

Semi-supervised learning has gained significant attention in recent applications. In this article, we provide a selective overview of popular semi-supervised methods and then propose a simple but effective algorithm for semi-supervised classification using support vector machines (SVM), one of the most popular binary classifiers in a machine learning community. The idea is simple as follows. First, we apply the dimension reduction to the unlabeled observations and cluster them to assign labels on the reduced space. SVM is then employed to the combined set of labeled and unlabeled observations to construct a classification rule. The use of SVM enables us to extend it to the nonlinear counterpart via kernel trick. Our numerical experiments under various scenarios demonstrate that the proposed method is promising in semi-supervised classification.

Keywords: dimension reduction, k-means clustering, semi-supervised classification, support vector machines

1. Introduction

In binary classification, we are given a set of data $(y_i, \mathbf{x}_i) \in \{-1, 1\} \times \mathbb{R}^p, i = 1, \ldots, n$ where y_i and $\mathbf{x}_i$ denote a class label and a p-dimensional predictor, respectively. The primal goal of binary classification is to learn a function f(classification/decision function) whose sign $sign\{f(\mathbf{x})\}$ predicts the corresponding class label y.

However, we often encounter cases where only a few observations have known labels, and most of them have unknown labels. That is, we are given two sets of data, labeled data $\mathcal{L} = \{(\mathbf{x}_i, y_i), i = 1, \ldots, \ell\}$ and unlabeled data $\mathcal{U} = \{\mathbf{x}_{\ell+1}, \ldots, \mathbf{x}_n\}$. Semi-supervised is proposed to handle such data efficiently by minimizing the information loss from the large number of missing labels in $\mathcal{U}$. Semi-supervised learning exploits both labeled and unlabeled datasets, while supervised learning only exploits the labeled dataset with a limited size. Thus, semi-supervised learning is particularly useful when the labeled data are very hard to collect while the unlabeled is much easier.

In this article, we provide a selective but systematic review of existing semi-supervised classification methods, and then propose a simple but efficient semi-supervised classification algorithm based on the support vector machine (SVM; Vapnik, 2015). To be more precise, we apply the k-means clustering algorithm to assign labels to $\mathcal{U}$. This idea is supported by the clustering assumption (Chapelle *et al.*, 2008) that is often required in semi-supervised learning community. We then apply the SVM

This work is supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (Grant No. NRF-2018R1D1A1B07043034 and NRF-2019R1A4A1028134) and Korea University (Grant No. K2105791).

[1] Corresponding author: Department of Statistics, Korea University, 145 Anam-Ro, Sungbuk-Gu, Seoul 02841, Korea. E-mail: sjshin@korea.ac.kr

Published 31 March 2022 / journal homepage: http://csam.or.kr

to train the classifier from both $\mathcal{L}$ and $\mathcal{U}$ after properly labelled. The proposed algorithm can be extended to nonlinear case by applying nonlinear dimension reduction before the clustering.

The rest of articles is organized as follows. In Section 2, we provide a selective overview of semi-supervised learning methods. In Section 3, the proposed algorithm based on k-means clustering and SVM is described in great detail, including its nonlinear extension. We conduct simulation to evaluate the performance of the proposed method in Section 4, and real data application follows in Section 5. Concluding remarks is given in Section 6.

2. Review

2.1. Self-training

The self-training algorithm (Yarowsky, 1995) first estimates f from $\mathcal{L}$ only and predicts the labels of $\mathcal{U}$. We then estimate f again from $\mathcal{L}$ and $\mathcal{U}$ with the estimated label in the previous Step. This procedure can be iterated until the estimated labels of $\mathcal{U}$ becomes stable, and the final classifier is then trained from the combined data, $\mathcal{L}$ and $\mathcal{U}$ with estimated labels. The idea is simple but requires the assumption that the model's high confidence predictions tend to be correct to guarantee that the estimated labels for $\mathcal{U}$ are accurate enough. The self-training algorithm does not impose a particular model, and hence can be applied to any classifiers such as SVM and boosting. The self-training algorithm is known to be effective in natural language processing. However, it is sensitive to the prediction accuracy of the initial classifier from $\mathcal{L}$ only. Finally, Algorithm 1 describes the self-training algorithm.

Algorithm 1 Self-training Algorithm

1. Train a classifier from the labeled data set, $\mathcal{L}$ denoted by $\tilde{f}^{(1)}$.
2. For $k = 1, 2, \ldots,$
   2-1. Predict y_j, $j = \ell + 1, \ldots, n$, the labels of $\mathcal{U}$ from $\tilde{f}^{(k)}$, $\tilde{y}_j^{(k)} = \tilde{f}^{(k)}(\mathbf{x}_j)$. Let
   $$\tilde{\mathcal{U}}^{(k)} = \left\{\left(\tilde{y}_j^{(k)}, \mathbf{x}_j\right), \quad j = \ell + 1, \ldots, n\right\}.$$
   2-2. Train a classifier from combined samples, $\mathcal{L} \cup \tilde{\mathcal{U}}^{(k)}$, denoted by $\tilde{f}^{(k+1)}$.
   2-3. If the estimated labels become stable,
   $$(n - \ell)^{-1} \sum_{j=\ell+1}^{n} \mathbb{1}\left\{\tilde{y}_j^{(k)} \neq \tilde{y}_j^{(k+1)}\right\} < \delta$$
   for a given value of $\delta \in (0, 1)$, then stop the iteration and go to the Step 3.
3. Return $\hat{f} = \tilde{f}^{(k)}$ as the final classifier.

2.2. Generative models

A generative model tries to uncover the data generating process, the distribution of $(Y, \mathbf{X})$ which directly yields a classification rule based on $P(Y \mid \mathbf{x})$. There are several models to estimate the joint distribution of $(Y, \mathbf{X})$: Gaussian mixture model (GMM), a mixture of multinomial distributions for categorical $\mathbf{x}$, and hidden Markov model (HMM) popular for natural language processing.

We focus on the Gaussian mixture model (GMM) the most popular one in a statistical community. Without loss of generality, we consider binary classification with $y \in \{-1, 1\}$ but the extension to the multi-class case is straightforward. The GMM with two components assumes the joint density of $(Y, \mathbf{X})$ denoted by $f(y, \mathbf{x} \mid \boldsymbol{\theta})$ being

$$f(y, \mathbf{x} \mid \boldsymbol{\theta}) = f(y \mid w) f\left(\mathbf{x} \mid y, \boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y\right), \tag{2.1}$$

where $\boldsymbol{\theta} = (w, \mu_+, \mu_-, \boldsymbol{\Sigma}_+, \boldsymbol{\Sigma}_-)$ denotes the model parameter vector. $f(y \mid w)$ and $f(\mathbf{x} \mid y, \boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y)$ denote the symmetric Bernoulli density with $P(Y = 1) = 1 - P(Y = -1) = w \in (0, 1)$ and the multivariate Gaussian density with mean vector $\boldsymbol{\mu}_y \in \mathbb{R}^p$ and covariance matrix $\boldsymbol{\Sigma}_y \in \mathbb{R}^{p \times p}$ where $y \in \{-1, 1\}$, respectively. Under the GMM (2.1), the class probability is

$$P(Y = 1 \mid \mathbf{x}, \boldsymbol{\theta}) = \frac{f(y = 1, \mathbf{x} \mid \boldsymbol{\theta})}{f(y = 1, \mathbf{x} \mid \boldsymbol{\theta}) + f(y = -1, \mathbf{x} \mid \boldsymbol{\theta})}. \tag{2.2}$$

The GMM is even more prevalent in unsupervised settings, when there is no labeled data available. The parameter estimation is not difficult thanks to the well-known Expectation-Maximization (EM) algorithm (Dempster *et al.*, 1977). Therefore, the GMM is naturally extended to semi-supervised problems by constructing the likelihood function (in log scale) as follows,

$$\ell(\boldsymbol{\theta} \mid \mathcal{L}, \mathcal{U}) = \underbrace{\sum_{i=1}^{\ell} \log f(y_i \mid \boldsymbol{\theta}) f(\mathbf{x}_i | y_i, \boldsymbol{\theta})}_{\mathcal{L}\text{: Labeled data}} + \underbrace{\sum_{j=\ell+1}^{n} \log \left\{ \sum_{y' \in \{-1,1\}} f(y' \mid \boldsymbol{\theta}) f(\mathbf{x}_j \mid y', \boldsymbol{\theta}) \right\}}_{\mathcal{U}\text{: Unlabeled data}}. \tag{2.3}$$

Similar to the unsupervised case, the EM algorithm can be employed to find MLE. Given the MLE of $\hat{\boldsymbol{\theta}}$ that maximizes (2.3), which directly yields a classification rule based on the plug-in estimator of (2.2).

One advantage of the generative model is that it uncovers the data generating process of $(\mathbf{X}, Y)$ which provides a transparent insight into the data. It tends to achieve optimal performance if the model assumption is satisfied. However, the final results heavily depend on the adequacy of the generative model.

2.3. Semi-supervised support vector machine (SVM)

Semi-supervised support vector machine (S^3VM; Chapelle *et al.*, 2008) is an extended version of standard SVM that exploits both the labeled and unlabeled datasets to train classification boundary that maximizes geometric margin of the data. The linear S^3VM that seeks a classification function $f(\mathbf{x}) = b + \mathbf{w}^T\mathbf{x}$ minimizes,

$$S(b, \mathbf{w}, \mathbf{y}_u) = \frac{1}{2}\|\mathbf{w}\|^2 + C_1 \sum_{i=1}^{\ell} H_1\left(y_i f(\mathbf{x}_i)\right) + C_2 \sum_{j=\ell+1}^{n} H_1\left(y_j f(\mathbf{x}_j)\right), \tag{2.4}$$

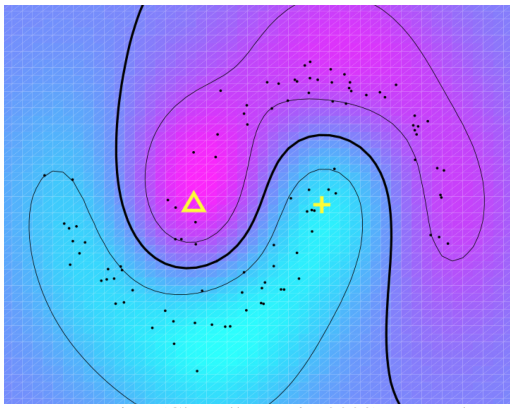

Figure 1: *Illustration of cluster assumption (Chapelle et al., 2008): Two clusters clearly separated by S^3VM solution, a solid line.*

where $\mathbf{y}_u = \{y_j, j = \ell+1, \ldots, n\}$ is the missing labels of $\mathcal{U}$, $H_1(u) = \max\{0, 1-u\}$ denotes the hinge loss function, and nonnegative constants C_1 and C_2 are tuning parameters that controls balances between the data fitting and model complexity. The first two terms corresponds to the labeled data $\mathcal{L}$ and (2.4) reduces to the SVM for $\mathcal{L}$ if $C_2 = 0$. The third term incorporates unlabeled observations, $\mathcal{U}$. That is, S^3VM tries to find a classification boundary that maximizes the margin of $\mathcal{U}$ among all possible configurations of unobserved labels $\mathbf{y}_u$. Non-linear extension on the reproducing kernel Hibert space (RKHS; Wahba, 1990) is straightforward via kernel trick just like the standard SVM.

Despite its conceptual simplicity, (2.4) is a combinatorial problem, infeasible unless n is very small. Various types of algorithms have been proposed to solve (2.4). Chapelle *et al.* (2008) provides a systematic overview of computational strategies of S^3VM and classifies the optimization strategies into two types: combinatorial optimization and continuous optimization. The combinatorial optimization seeks a minimizer of $T(\mathbf{y}_u)$ over all possible configuration of the binary vector $\mathbf{y}_u$ where

$$T(\mathbf{y}_u) = \min_{b,\mathbf{w}} S(b, \mathbf{w}, \mathbf{y}_u). \tag{2.5}$$

On the other hand, the continuous optimization modifies (2.4) in order to eliminate $\mathbf{y}_u$ in the optimization, and minimizes

$$\frac{1}{2}\|\mathbf{w}\|^2 + C_1 \sum_{i=1}^{\ell} H_1\left(y_i f(\mathbf{x}_i)\right) + C_2 \sum_{j=\ell+1}^{n} H_1\left(|f(\mathbf{x}_j|\right), \tag{2.6}$$

with respect to $(b, \mathbf{w})$. This is known as the transductive SVM (TSVM; Collobert *et al.*, 2006), another popular tool for semi-supervised classification. Collobert *et al.* (2006) further modified TSVM objective function by replacing the hinge loss function in the last term of (2.6) with the ramp loss function $R_s(u) = \min\{1-s, \max\{0, 1-u\}\}$ (Shen *et al.*, 2003; Wu and Liu, 2007) to facilitate the optimization via concave-convex procedure (CCCP; Le Thi Hoai and Tao, 1997).

3. Proposed method

Both the self-training algorithm and S^3VM introduced in Section 2 requires the cluster assumption, which states that two points $\mathbf{x}$ and $\mathbf{x}'$ belongs to the same cluster (with high probability) if there is a

"path" between the two points that moves through regions with a high density of $P(\mathbf{X} = \mathbf{x})$. Figure 1 (Chapelle *et al.*, 2008) illustrates toy examples with two labeled observations (triangle and cross) and 100 unlabeled observations. Under the cluster assumption, the S^3VM solution (a solid line) accurately identifies the true classification boundary even for two observations with labels.

The cluster assumption implies that observations in $\mathcal{U}$ should be clustered, which motivates one to develop a simple clustering $\mathcal{U}$ to predict their labels. There are several popular clustering methods in statistical learning. Among many others, the density-based spatial clustering of applications with noise (DBSCAN; Ester *et al.*, 1996) that defines clusters based on the density of observations seems conceptually well-matched to the CA assumption. However, DBSCAN has two tuning parameters whose choice is crucial for the results and usually yields the noise points that fail to belong to any clusters. For this reason, in this article, we propose to use the k-means clustering, one of the most popular clustering methods in statistics due to its simplicity. Of course, the k-means clustering also has a couple of drawbacks. First, the k-means clustering requires the number of clusters, k as an input of the algorithm. However, in our problem where cluster membership is used for labeling $\mathcal{U}$, the number of clusters is not essential. We can use sufficiently large k since it allows the same labels for different clusters. The second problem is that the k-means clustering cannot identify clusters when their boundaries are highly nonlinear, for example, as shown in Figure 1. This is the motivation of DBSCAN. To tackle this, we propose to apply the nonlinear dimension reduction such as kernel principal component analysis (KPCA) to observations in $\mathcal{U}$ before applying the clustering. The kernel PCA efficiently identifies nonlinear features of $\mathcal{U}$ expressed on RKHS, then nonlinear cluster boundary can be uncovered by the k-means algorithm on the nonlinear feature space, RKHS.

After clustering, for each cluster, we count the frequency of the observed labels and then assign the class label corresponding to the maximum frequency. That is, for the j^{th} cluster, we assign the label k_j^* to the j^{th} cluster

$$k_j^* = \arg\max_k n_j^{(k)}, \tag{3.1}$$

where $n_j^{(k)}$ denotes the number of observations from the k^{th} class in the j^{th} cluster. When k_j^* is not unique, we can randomly choose one of them.

Finally, we can use both $\mathcal{L}$ and $\mathcal{U}$ for learning classification rule since $\mathcal{U}$ is properly labeled. The proposed algorithm is summarized in the following. Due to the cluster assumption in (Chapelle

Algorithm 2 Algorithm 1 - Linear learning

Input: a training dataset, $\mathcal{D} = \mathcal{L} \cup \mathcal{U}$; the number of classes, K;

Step 1. (Clustering) Apply k-means clustering to the predictors in $\mathcal{D}$

Step 2. (Labeling) For each cluster, assign a proper label using (3.1).

Step 3. (Learning) Apply the SVM algorithm to both $\mathcal{L}$ and the labelled $\mathcal{U}$ in order to learn f.

Output: Trained model f

et al., 2008), the observations in the same cluster should have the same label. It is simple to check whether the data satisfies the cluster assumption in linear representation. However, the standard for determining the same cluster is ambiguous when the decision boundary is highly nonlinear. To tackle this issue, we propose to apply kernel principal component analysis (KPCA) to the predictors in $\mathcal{D}$. The modified algorithm for nonlinear learning is given below.

There are several minor issues to complete the algorithm, which shall be discussed in the following. First, we need to choose a proper kernel function for both KPCA and SVM. It is natural to use the

Algorithm 3 Algorithm 2 - Nonlinear learning

Input: a training dataset, $\mathcal{D} = \mathcal{L} \cup \mathcal{U}$; the number of classes, K;

Step 0. (Feature Extraction) Apply kernel PCA to predictors in $\mathcal{D}$ for extracting nonlinear features.

Step 1. (Clustering) Apply k-means clustering to the extracted features obtained in Step 0.

Step 2. (Labeling) For each cluster, assign a proper label using (3.1).

Step 3. (Learning) Apply the SVM algorithm to both $\mathcal{L}$ and the labelled $\mathcal{U}$ in order to learn f.

Output: Trained model f

same kernel function for them, and we employ the radial basis kernel $K(\mathbf{x}, \mathbf{x}') = \exp\{-\|\mathbf{x} - \mathbf{x}'\|/(2\sigma^2)\}$ with σ set to be median pairwise distance between the predictors in $\mathcal{L}$. Secondly, it is required in PCA to decide the number of features to be used for the cluster. One common choice is to use cumulative eigenvalues of the kernel working matrix. However, we alternatively propose to use the Hotelling's T^2 test (Lu *et al.*, 2005) when for the binary classification and the F statistic (Shaw and Mitchell-Olds, 1993) for the multiclass classification, instead of eigenvalues. The idea behind this is that the larger statistic implies more classification information contained in the feature. Our limited simulation demonstrates that the proposed idea based on the test statistic outperforms the eigenvalue-based one. In the upcoming examples, we use 80% for the cutoff.

4. Simulation

In this section, we conduct several simulation studies to evaluate the prediction performance of the proposed algorithm. The existing methods reviewed in Section 2 are considered as competing methods. That is, we include the self-training algorithm, the generative models based on the EM algorithm. We use `sslGmmEM()` function in R-package `SSL` to fit the generative model. We also tried the TSVM using `TSVM()` function in R-package `RSSL`. However, the function did not work in our simulation except for the linear binary classification, and we could not include it in the comparison. Instead, we include the supervised SVM for $\mathcal{L}$ only as a baseline competitor.

We generate 500 training sample size for $\mathcal{D}$ with$(\alpha \times 100)\%$ unlabelled observations where $\alpha = \{0.80, 0.90, 0.95\}$ is considered. The prediction accuracy is evaluated for independent test sample of size $n \in \{100, 500, 1000\}$.

4.1. Linear classification

For linear classification, we consider the following scenarios:

A1 (Binary classification) Let $\mathbf{x} \mid (y = 1) \sim \text{MN}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $\mathbf{x} \mid (y = -1) \sim \text{MN}(-\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with $\boldsymbol{\mu} = (0.7, \ldots, 0.7_{20})^T$ and $\boldsymbol{\Sigma} = 4\mathbf{I}_{20}$.

A2 (Multi-class classification) Let $\mathbf{x} \mid (y = 1) \sim \text{MN}(\mathbf{1}_{20}^T, 4\mathbf{I}_{20})$, $\mathbf{x} \mid (y = 2) \sim \text{MN}(\mathbf{2}_{20}^T, 4\mathbf{I}_{20})$ and $\mathbf{x} \mid (y = 3) \sim \text{MN}(\mathbf{3}_{20}^T, 4\mathbf{I}_{20})$, respectively.

Table 1 contains the test prediction accuracy for different methods, averaged over one hundred independent repetitions. The numbers in parentheses are corresponding standard deviations. Although all methods considered show comparable performance, one can observe that the proposed method outperforms as α increases, , the number of labeled observations get decreases.

Table 1: Linear classification – averaged prediction accuracy over 100 independent repetitions are reported under model A1 and A2. The numbers in parentheses are the corresponding standard deviations

Model	n	α	Self-training	Generative model	SVM	Proposed method
		0.80	0.928 (0.02)	0.927 (0.05)	0.926 (0.03)	0.926 (0.03)
	100	0.90	0.912 (0.05)	0.897 (0.11)	0.900 (0.05)	0.922 (0.03)
		0.95	0.888 (0.05)	0.841 (0.17)	0.882 (0.05)	0.903 (0.05)
		0.80	0.931 (0.01)	0.928 (0.04)	0.925 (0.01)	0.927 (0.02)
$A1$	500	0.90	0.913 (0.05)	0.896 (0.10)	0.899 (0.04)	0.914 (0.03)
		0.95	0.878 (0.04)	0.834 (0.17)	0.872 (0.03)	0.900 (0.05)
		0.80	0.932 (0.01)	0.928 (0.04)	0.927 (0.01)	0.926 (0.01)
	1000	0.90	0.915 (0.05)	0.898 (0.10)	0.901 (0.04)	0.917 (0.02)
		0.95	0.877 (0.03)	0.835 (0.17)	0.870 (0.03)	0.906 (0.04)
		0.80	0.898 (0.04)	0.868 (0.08)	0.881 (0.04)	0.885 (0.04)
	100	0.90	0.818 (0.04)	0.786 (0.13)	0.821 (0.04)	0.865 (0.04)
		0.95	0.784 (0.06)	0.656 (0.11)	0.789 (0.06)	0.814 (0.08)
		0.80	0.897 (0.03)	0.866 (0.07)	0.888 (0.03)	0.897 (0.02)
$A2$	500	0.90	0.857 (0.02)	0.774 (0.12)	0.851 (0.03)	0.884 (0.04)
		0.95	0.811 (0.05)	0.676 (0.11)	0.804 (0.04)	0.836 (0.07)
		0.80	0.898 (0.03)	0.886 (0.09)	0.887 (0.03)	0.897 (0.02)
	1000	0.90	0.859 (0.02)	0.777 (0.12)	0.852 (0.03)	0.880 (0.04)
		0.95	0.809 (0.04)	0.677 (0.11)	0.803 (0.04)	0.850 (0.07)

Table 2: Nonlinear classification - averaged prediction accuracy over 100 independent repetitions are reported under model B1 and B2. The numbers in parentheses are the corresponding standard deviations

Model	n	α	Self-training	Generative model	SVM	Proposed method
		0.80	0.516 (0.06)	0.506 (0.05)	0.515 (0.05)	0.584 (0.05)
	100	0.90	0.508 (0.06)	0.499 (0.07)	0.508 (0.06)	0.565 (0.06)
		0.95	0.513 (0.06)	0.508 (0.06)	0.514 (0.06)	0.557 (0.06)
		0.80	0.517 (0.04)	0.502 (0.04)	0.516 (0.04)	0.577 (0.04)
$B1$	500	0.90	0.506 (0.04)	0.499 (0.04)	0.507 (0.04)	0.557 (0.04)
		0.95	0.517 (0.04)	0.504 (0.04)	0.518 (0.04)	0.550 (0.05)
		0.80	0.514 (0.03)	0.502 (0.03)	0.514 (0.03)	0.577 (0.03)
	1000	0.90	0.512 (0.03)	0.502 (0.04)	0.512 (0.03)	0.564 (0.04)
		0.95	0.514 (0.04)	0.501 (0.04)	0.514 (0.04)	0.545 (0.04)
		0.80	0.517 (0.05)	0.350 (0.08)	0.523 (0.05)	0.550 (0.13)
	100	0.90	0.487 (0.06)	0.351 (0.08)	0.487 (0.06)	0.492 (0.13)
		0.95	0.443 (0.05)	0.328 (0.06)	0.446 (0.06)	0.477 (0.07)
		0.80	0.521 (0.03)	0.352 (0.07)	0.523 (0.03)	0.553 (0.10)
$B2$	500	0.90	0.488 (0.04)	0.354 (0.07)	0.489 (0.04)	0.504 (0.10)
		0.95	0.455 (0.05)	0.334 (0.04)	0.456 (0.05)	0.496 (0.08)
		0.80	0.523 (0.03)	0.353 (0.07)	0.526 (0.02)	0.564 (0.08)
	1000	0.90	0.492 (0.03)	0.350 (0.07)	0.493 (0.03)	0.526 (0.09)
		0.95	0.453 (0.05)	0.334 (0.03)	0.456 (0.04)	0.479 (0.08)

4.2. Nonlinear classification

The following models are considered to evaluate the proposed nonlinear algorithm.

B 1 (Binary classification) Let $y = \text{sign}\{f(\mathbf{x}) + k + \epsilon\}$, where $f(\mathbf{x}) = \log(2x_1^2 + 2x_2^2)$ and $x_1, x_2 \sim N(0,1)$ and $\epsilon \sim N(0, 3^2)$.

B 2 (Multi-class classification) Let y is determined by the 33 percentile and 66 percentile of $f(\mathbf{x})$, where $f(\mathbf{x}) = 2\cos(x_1) + \log(1 + x_2^2)$ and $x_1, x_2 \sim N(0,1)$.

Table 2 shows the results for the nonlinear models. The generative model fails since the model

Table 3: Results for the iris data – averaged prediction accuracy over 100 independent random partitioning for the iris data. The numbers in parentheses are the corresponding standard deviations

$n_{\text{train}} : n_{\text{test}}$	prop	Self-training	Generative model	SVM	Proposed method
100:50	0.7	0.944 (0.04)	0.668 (0.31)	0.945 (0.04)	0.937 (0.05)
	0.8	0.927 (0.05)	0.567 (0.36)	0.930 (0.05)	0.939 (0.05)
	0.9	0.868 (0.17)	0.523 (0.32)	0.868 (0.11)	0.872 (0.12)
75:75	0.7	0.942 (0.04)	0.594 (0.25)	0.940 (0.03)	0.925 (0.05)
	0.8	0.919 (0.05)	0.530 (0.35)	0.920 (0.04)	0.926 (0.05)
	0.9	0.854 (0.15)	0.515 (0.33)	0.849 (0.10)	0.875 (0.10)
50:100	0.7	0.919 (0.06)	0.564 (0.21)	0.924 (0.05)	0.922 (0.05)
	0.8	0.886 (0.12)	0.490 (0.20)	0.871 (0.10)	0.887 (0.11)
	0.9	0.797 (0.18)	0.500 (0.24)	0.809 (0.10)	0.840 (0.07)

assumption is violated, and other methods perform reasonably well. Again, the proposed method outperforms all others, just like the linear cases.

4.3. Computing time

We also compare the computing time of the proposed algorithm. Figure 2 depicts the averaged computing time in seconds over the 10 repetitions for model (A1) and (B1), for different sample sizes $n \in \{100, 200, \ldots, 1000\}$. It is observed that the elapsed time for all methods linearly increases as the sample size n increases. As expected, the supervised SVM is the fastest since it only uses $\mathcal{L}$ for estimating f. In linear cases, one can observe that the proposed method is computationally efficient than the generative model, and is comparable to the self-training algorithm. In nonlinear cases, we observed our algorithm becomes slow as n increases. This is because the K-means clustering algorithm takes a much longer time in kernel space. However, the K-means clustering algorithm is not essential for our method, and one can replace it with better and faster alternative clustering algorithms to improve computational efficiency.

5. Real data analysis

5.1. Iris data

As a final showcase, we apply the proposed method to `iris` data available in `R`. It consists of 150 observations with four covariates as follows: sepal length, sepal width, petal length, and petal width. The response variable is the three types of irises with *setosa*, *versicolor*, and *virginica*. We compare the proposed method with three learning algorithms used in Section 4 under different proportions of $\mathcal{U}$, $\alpha \in \{0.7, 0.8, 0.9\}$. We randomly split the data into the training and test datasets with different sample sizes, $n_{\text{train}} : n_{\text{test}} = \{100 : 50, 75 : 75, 50 : 100\}$. Table 3 presents the averaged prediction accuracy of the methods over a hundred independent repetitions of the random partitioning. Again, one can see that the proposed method show better performance than others when the missing label proportion increases.

5.2. Wisconsin diagnostic breast cancer data

We also apply the proposed method to Wisconsin diagnostic breast cancer (WDBC) available in UCI machine learning repository, one of the most popular data sets for binary classification. This dataset is composed of a total of 569 observations with benign and malign cases being 357 and 212 observations, and 30 real-valued covariates. To generate unlabeled data, we randomly choose 90% of samples and

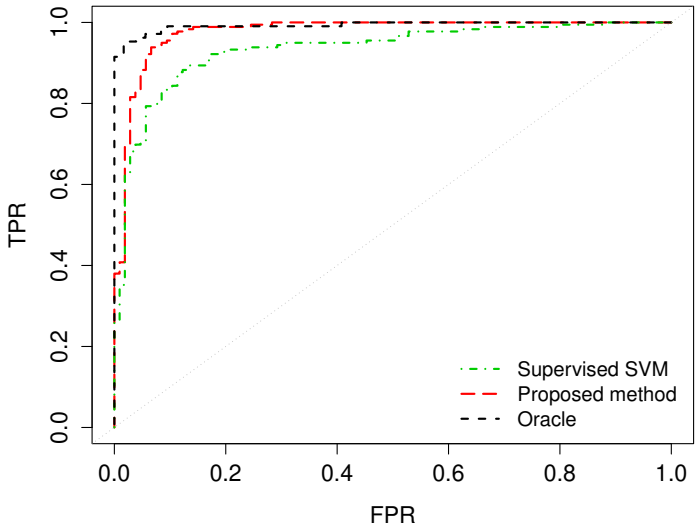

Figure 2: *Test ROC curves for WDBC data.*

remove their labels. We then randomly split the data set into the training and test set with equal sizes.

Finally, we apply the proposed algorithm to the WDBC data, and Figure 2 depicts the test ROC curve of the proposed method. We also provide two more ROC curves of supervised SVMs, one from the original data without missing labels (denoted by oracle) and another from labeled data only (denoted by supervised SVM). We remark that the oracle represents an ideal but unrealistic result that any semi-supervised methods cannot beat. However, one can see that the proposed method shows much-improved performance in terms of test ROC curve than the supervised SVM using observed label only.

6. Conclusion

In this paper, we proposed a simple algorithm for semi-supervised classification based on the SVM and the k-means clustering, both of which are very popular in a statistical community. Employing the kernel trick, the proposed method can readily be extended to the nonlinear classification problem. The proposed algorithm is conceptually straightforward and easy to implement and thus practically attractive as demonstrated by the simulation and real data analysis.

Acknowledgments

This work is partially funded by the National Research Foundation of Korea (NRF) grants 2018R1D1 A1B07043034 and 2019R1A4A1028134, and by Korea University grant K2105791.

References

Chapelle O, Sindhwani V, and Keerthi SS (2008). Optimization techniques for semi-supervised support vector machines, *Journal of Machine Learning Research*, **9**, 203–233.

Collobert R, Sinz F, Weston J, and Bottou L (2006). Large scale transductive svms, *Journal of Ma-*

chine Learning Research, **7**, 1687–1712.

Dempster AP, Laird NM, and Rubin DB (1977). Maximum likelihood from incomplete data via the em algorithm, *Journal of the Royal Statistical Society: Series B (Methodological)*, **39**, 1–22.

Ester M, Kriegel HP, Sander J, and Xu X (1996). A density-based algorithm for discovering clusters in large spatial databases with noise., *Kdd*, **96**, 226–231.

Le Thi Hoai A and Tao PD (1997). Solving a class of linearly constrained indefinite quadratic problems by dc algorithms, *Journal of global optimization*, **11**, 253–285.

Lu Y, Liu PY, Xiao P, and Deng HW (2005). Hotelling's t 2 multivariate profiling for detecting differential expression in microarrays, *Bioinformatics*, **21**, 3105–3113.

Shaw RG and Mitchell-Olds T (1993). Anova for unbalanced data: an overview, *Ecology*, **74**, 1638–1645.

Shen X, Tseng GC, Zhang X, and Wong WH (2003). On ψ-learning, *Journal of the American Statistical Association*, **98**, 724–734.

Vapnik V (2015). *The Nature of Statistical Learning Theory*, Springer science & business media.

Wahba G (1990). *Spline Models for Observational Data*, Philadelphia, SIAM.

Wu Y and Liu Y (2007). Robust truncated hinge loss support vector machines, *Journal of the American Statistical Association*, **102**, 974–983.

Yarowsky D (1995). Unsupervised word sense disambiguation rivaling supervised methods, *33rd Annual Meeting of the Association for Computational Linguistics*, 189–196.

Received August 21, 2021; Revised October 15, 2021; Accepted October 28, 2021