Improving Classifier Performance using Data with Different Taxonomies

Tomoharu Iwata, Toshiyuki Tanaka, *Member, IEEE,* Takeshi Yamada, *Member, IEEE,* and Naonori Ueda *Member, IEEE,*

Abstract—We propose a framework for improving classifier performance by effectively using auxiliary samples. The auxiliary samples are labeled not in terms of the target taxonomy according to which we wish to classify samples, but according to classification schemes or taxonomies that are different from the target taxonomy. Our method finds a classifier by minimizing a weighted error over the target and auxiliary samples. The weights are defined so that the weighted error approximates the expected error when samples are classified into the target taxonomy. Experiments using synthetic and text data show that our method significantly improves the classifier performance in most cases compared to conventional data augmentation methods.

Index Terms—transfer learning, semi-supervised learning, text classification

1 Introduction

In general, performance of a classifier can be improved as the number of training samples is increased. In real applications, however, one does not necessarily have enough training samples to achieve a reasonable performance. In order to circumvent such difficulties, there has been great interest in methods that augment and effectively increase training samples, such as semi-supervised learning [1] and domain adaptation [2], where the former augments a set of training samples with unlabeled samples and the latter with samples from different domains. Another way to improve performance with fewer training samples is active learning [3], which constructs a set of training samples by actively selecting unlabeled samples and making queries for their labels, thereby effectively reducing required labeled samples for training.

In this paper we consider a related but different setting, in which we have not only target samples that are labeled using the same taxonomy as the one we want to classify samples, but also auxiliary samples that are labeled using different taxonomies. In many applications, there are few target samples but a lot of auxiliary samples. For example, suppose that one wants to classify web pages automatically according to her personal taxonomy. It is almost always the case that one has only a few web pages labeled with her taxonomy, since labeling web pages is a tiresome and time-consuming task. On the other hand, there are a huge collection of web pages available that have already been categorized according to various taxonomies by somebody else, such as directory search engines or users in social bookmark sites. Such auxiliary samples can be useful, because there may be classes in the auxiliary samples that are similar in their classification criteria to those in the target samples, and samples in these classes may be helpful in training a classifier for the target taxonomy. However, supervised learning methods cannot utilize such auxiliary samples, since class labels given to the auxiliary samples are semantically different from those given to the target samples; even if labels are nominally the same, the classification criteria may be different.

Our solution, proposed in this paper, is a simple framework for making efficient use of auxiliary samples with different taxonomies in order to improve performance of a classifier. Our method automatically finds auxiliary classes that are similar to target classes, and trains a classifier by using not only target samples but also auxiliary samples with class-dependent weights. The weights represent similarities between the target and auxiliary classes, and it is important to set them appropriately for improving classifier performance. Our method sets the weights so that the error with these weights over the target and auxiliary samples approximates the expected error when samples are classified into target classes. Therefore, we can learn a classifier for target classes by minimizing the weighted error. We call our framework *class adaptation*. Our framework of making use of auxiliary samples is widely applicable in conjunction with a variety of existing classifiers simply by introducing weights without modifying the classifiers themselves.

2 Problem Setting

Let $\boldsymbol{T}$ be a set of target classes, and $\boldsymbol{A}$ be a set of auxiliary classes. The goal is to find a classifier $f : \boldsymbol{x} \mapsto y$, $y \in \boldsymbol{T}$ using target samples $D_T = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N_T}$, where $y_n \in \boldsymbol{T}$, and auxiliary samples $D_A = \{(\boldsymbol{x}_n, y_n)\}_{n=N_T+1}^{N}$, where $y_n \in \boldsymbol{A}$. Here, $\boldsymbol{x}_n$ is the feature vector of the nth sample, y_n is the class label of the nth sample, N_T is the number

- *T. Iwata, T. Yamada, and N. Ueda are with NTT Communication Science Laboratories.*
- *T. Tanaka is with Graduate School of Informatics, Kyoto University.*

of samples with target class labels, and N is the number of all samples. We assume that y is a discrete variable. We also assume that $\boldsymbol{x}$ is a vector of discrete variables throughout this paper. Note that the following analysis is unchanged if some or all of elements in $\boldsymbol{x}$ are continuous, except that the summations should be replaced with integrations.

In general, a classifier is trained by minimizing the following empirical error over the given target samples:

$$E_T(f) = \frac{1}{N_T}\sum_{n=1}^{N_T} J(\boldsymbol{x}_n, y_n; f), \tag{1}$$

where the error function $J(\boldsymbol{x}, y; f)$ represents error of the classifier f given the sample $(\boldsymbol{x}, y)$. Typical error functions include negative log likelihood $J(\boldsymbol{x}, y; f) = -\log P(y|\boldsymbol{x}; f)$, and 0-1 loss function, $J(\boldsymbol{x}, y; f) = 0$ if $f(\boldsymbol{x}) = y$ and $J(\boldsymbol{x}, y; f) = 1$ otherwise. As the number of target samples N_T grows to infinity, $E_T(f)$ converges to the expected error. Therefore, the minimization of $E_T(f)$ will lead to the minimization of expected error if we have sufficient numbers of target samples. However, if we have only a few target samples, the trained classifier can overfit the given target samples, which results in poor performance.

3 Proposed method

3.1 Weighted error

We assume that the number of available target samples is small, and consider how to make efficient use of auxiliary samples in order to improve classifier performance.

Let $\boldsymbol{Y} = \boldsymbol{T} \cup \boldsymbol{A}$ be a set of all classes. Assume that we have weight $w(t|y)$ that represents correlation or similarity viewed from class $y \in \boldsymbol{Y}$ to class $t \in \boldsymbol{T}$. We learn a classifier by minimizing the following weighted error over the given target and auxiliary samples:

$$E(f) = \frac{1}{N}\sum_{n=1}^{N}\sum_{t\in\boldsymbol{T}} w(t|y_n) J(\boldsymbol{x}_n, t; f). \tag{2}$$

If we choose weights so that $w(t|y) = 1$ if $t = y$ and $w(t|y) = 0$ otherwise, then the weighted error $E(f)$ coincides up to the overall factor N_T/N with the empirical error over the target samples $E_T(f)$ in (1), which means that the weighted error (2) includes the empirical error (1) as a special case. In general, $w(t|y)$ takes a value between 0 and 1. If we wish, we can impose a constraint that the weight between a target class and another target class is zero: $w(t|y) = 0$ if $y \in \boldsymbol{T}$ and $y \neq t$. We do not use the constraint, however, because one can improve classifier performance by making use of relationships among target classes if target classes form a hierarchical structure [4].

Since we want a classifier that classifies a sample into one of the target classes with as small error as possible, we need to set weights $\{w(t|y)\}$ so that the weighted error approximates the expected error for target classes:

$$E(f) \approx \sum_{\boldsymbol{x}}\sum_{t\in\boldsymbol{T}} P(\boldsymbol{x}, t) J(\boldsymbol{x}, t; f) \equiv \mathcal{E}(f). \tag{3}$$

In order to accomplish this approximation, we consider the following generative model for the whole samples $D_T \cup D_A$. We regard that each sample consists of the triplet $(\boldsymbol{x}, y, t)$, and that only the pair $(\boldsymbol{x}, y)$ is observed as a sample, where $y \in \boldsymbol{Y}$ and $t \in \boldsymbol{T}$ represent observed and target classes of the sample, respectively. Thus, t is a latent, or hidden, variable of our generative model. Assuming that $\boldsymbol{x}$ and t are conditionally independent given y, i.e. $P(\boldsymbol{x}|y, t) = P(\boldsymbol{x}|y)$, the joint distribution of $\boldsymbol{x}$ and t is written as follows:

$$P(\boldsymbol{x}, t) = \sum_{y\in\boldsymbol{Y}} P(t)P(y|t)P(\boldsymbol{x}|y), \tag{4}$$

where $P(t)$ is a prior distribution of the target class t, where $P(y|t)$ is a conditional distribution which describes how the observed class y depends on the target class t, and where $P(\boldsymbol{x}|y)$ is a distribution of the sample attributes $\boldsymbol{x}$ given the observed class y.

By using (4), the expected error is rewritten as follows:

$$\begin{aligned}\mathcal{E}(f) &= \sum_{\boldsymbol{x}}\sum_{t\in\boldsymbol{T}} J(\boldsymbol{x}, t; f)\sum_{y\in\boldsymbol{Y}} \frac{P(t)P(y|t)}{P(y)} P(\boldsymbol{x}|y)P(y) \\ &= \sum_{\boldsymbol{x}}\sum_{t\in\boldsymbol{T}} J(\boldsymbol{x}, t; f)\sum_{y\in\boldsymbol{Y}} P(t|y)P(\boldsymbol{x}, y) \\ &= \sum_{\boldsymbol{x}}\sum_{y\in\boldsymbol{Y}} P(\boldsymbol{x}, y)\sum_{t\in\boldsymbol{T}} J(\boldsymbol{x}, t; f)P(t|y),\end{aligned} \tag{5}$$

which is the expected value of $\sum_{t\in\boldsymbol{T}} J(\boldsymbol{x}, t; f)P(t|y)$ with respect to $P(\boldsymbol{x}, y)$. The expected value can be approximated by an empirical expected value over the samples as:

$$\begin{aligned}\mathcal{E}(f) &\approx \sum_{\boldsymbol{x}}\sum_{y\in\boldsymbol{Y}} \frac{1}{N}\sum_{n=1}^{N} \delta_{(\boldsymbol{x}_n, y_n)}(\boldsymbol{x}, y)\sum_{t\in\boldsymbol{T}} J(\boldsymbol{x}, t; f)P(t|y) \\ &= \frac{1}{N}\sum_{n=1}^{N}\sum_{t\in\boldsymbol{T}} P(t|y_n)J(\boldsymbol{x}_n, t; f),\end{aligned} \tag{6}$$

where $P(\boldsymbol{x}, y)$ is approximated by the empirical distribution $P(\boldsymbol{x}, y) \approx \frac{1}{N}\sum_{n=1}^{N} \delta_{(\boldsymbol{x}_n, y_n)}(\boldsymbol{x}, y)$. Here, $\delta_{(\boldsymbol{x}_n, y_n)}(\boldsymbol{x}, y)$ represents Kronecker's delta, i.e. $\delta_{(\boldsymbol{x}_n, y_n)}(\boldsymbol{x}, y) = 1$ if $(\boldsymbol{x}_n, y_n) = (\boldsymbol{x}, y)$ and 0 otherwise. Identifying the approximated $\mathcal{E}(f)$ with the weighted error $E(f)$, we see that the weights $w(t|y)$ should be set equal to $P(t|y)$, that is, we have:

$$w(t|y) = \frac{P(t)P(y|t)}{P(y)}, \tag{7}$$

where the Bayes rule $P(t|y) = \frac{P(t)P(y|t)}{P(y)}$ is used. The above formula shows that one can calculate the proper weights $w(t|y)$ once we have estimates for $P(t)$, $P(y)$, and $P(y|t)$.

3.2 Estimation of weights

The maximum likelihood estimate of $P(y)$ is given straightforwardly by $\hat{P}(y) = N_y/N$, where N_y denotes the number of samples with class y.

In order to obtain an estimate of $P(t)$, we assume that $P(t)$, which is the probability of the target class t in the whole samples $D_T \cup D_A$, is the same as the probability of the class t in the target samples D_T. It is reasonable because the target taxonomy should be regarded as somehow personal in our assumed scenario, and it is unlikely that the external sources of auxiliary samples are affected by the target taxonomy in their classification of samples. Under this assumption, we can obtain a maximum likelihood estimate of $P(t)$, which is given by $\hat{P}(t) = N_t/N_T$, where N_t is the number of samples with class t.

We can estimate a set of conditional probabilities $\{\boldsymbol{P}_t\}_{t\in\boldsymbol{T}}$, where $\boldsymbol{P}_t = (P(y|t))_{y\in\boldsymbol{Y}}$, using the EM algorithm. Under our generative model (4), the log likelihood for class t being maximized is written as follows:

$$L(\boldsymbol{P}_t) = \sum_{n:y_n=t} \log \sum_{y\in\boldsymbol{Y}} P(y|t)\hat{P}_{-n}(\boldsymbol{x}_n|y), \tag{8}$$

where we use leave-one-out cross-validation (LOO) and thus $\hat{P}_{-n}(\boldsymbol{x}|y)$ represents an estimate of the model distribution of class y that is estimated in advance using samples with class y excluding the nth sample. For the model distributions, one can arbitrarily assume a generative model that is appropriate for the given samples, such as Gaussian or multinomial distribution. If we estimate the conditional probabilities $\boldsymbol{P}_t$ using training samples that are also used for estimating the model distribution, the estimation will become biased. Specifically, we will obtain $w(t|y) = 0$ if $y \neq t$, so that auxiliary samples will not be used at all. Therefore, we use LOO. Note that $\hat{P}_{-n}(\boldsymbol{x}|y)$ for $y \in \boldsymbol{A}$ is the same as the respective non-LOO estimate. Note also that when we use an exponential family for the model distribution, the computational cost required for the LOO estimation is no greater than that of non-LOO estimation if sufficient statistics are calculated in advance. When the model distributions $\hat{P}_{-n}(\boldsymbol{x}|y)$ are fixed, the global optimality of the estimate is guaranteed since $L(\boldsymbol{P}_t)$ is convex with respect to $\boldsymbol{P}_t$.

Let $\boldsymbol{P}_t^{(\tau)}$ be an estimate at τth step of the EM algorithm. The conditional expectation of the complete-data log likelihood being maximized is:

$$\begin{aligned} &Q(\boldsymbol{P}_t|\boldsymbol{P}_t^{(\tau)}) \\ &\quad = \sum_{n:y_n=t}\sum_{y\in\boldsymbol{Y}} P(y|\boldsymbol{x}_n, t; \boldsymbol{P}_t^{(\tau)}) \log P(y|t)\hat{P}_{-n}(\boldsymbol{x}_n|y) \end{aligned} \tag{9}$$

Here, the class posterior probability of $\boldsymbol{x}_n$ given the current estimate $\boldsymbol{P}_t^{(\tau)}$ can be computed via the Bayes rule:

$$P(y|\boldsymbol{x}_n, t; \boldsymbol{P}_t^{(\tau)}) = \frac{P^{(\tau)}(y|t)\hat{P}_{-n}(\boldsymbol{x}_n|y)}{\sum_{y'\in\boldsymbol{Y}} P^{(\tau)}(y'|t)\hat{P}_{-n}(\boldsymbol{x}_n|y')}. \tag{10}$$

TABLE 1
The algorithm of our method.

```
Input: target samples $D_T$, auxiliary samples $D_A$
Output: classifier $f$
1) Compute the estimation of the prior probabilities $\hat{P}(t)$ for $t \in \boldsymbol{T}$
   and $\hat{P}(y)$ for $y \in \boldsymbol{Y}$
2) Compute the LOO estimation of the model distributions
   $\hat{P}_{-n}(\boldsymbol{x}|y)$ for $y \in \boldsymbol{Y}$, $n = 1, \cdots, N$
3) Compute the estimation of the conditional probabilities $\hat{P}(y|t)$
   for $t \in \boldsymbol{T}$, $y \in \boldsymbol{Y}$ using EM algorithm as in (10) and (11)
4) (weight estimation) Set weights $w(t|y) = \frac{\hat{P}(t)\hat{P}(y|t)}{\hat{P}(y)}$ for $t \in \boldsymbol{T}$,
   $y \in \boldsymbol{Y}$
5) (classifier training) Find classifier $f$ by minimizing the weighted
   error $E(f) = \frac{1}{N}\sum_{n=1}^{N}\sum_{t\in\boldsymbol{T}} w(t|y_n)J(\boldsymbol{x}_n, t; f)$ while fixing
   weights
```

An estimate of the conditional probabilities at the next step, $\boldsymbol{P}_t^{(\tau+1)} = (P^{(\tau+1)}(y|t))_{y\in\boldsymbol{Y}}$, is obtained by maximizing $Q(\boldsymbol{P}_t|\boldsymbol{P}_t^{(\tau)})$ with respect to $\boldsymbol{P}_t$, yielding:

$$P^{(\tau+1)}(y|t) = \frac{1}{N_t}\sum_{n:y_n=t} P(y|\boldsymbol{x}_n, t; \boldsymbol{P}_t^{(\tau)}). \tag{11}$$

By iterating these steps until a convergence criterion is satisfied, we obtain the global optimum solution of $\boldsymbol{P}_t$.

Table 1 shows the algorithm of our method for improving classifier performance by using auxiliary samples. Our method has two parts, weight estimation and classifier training. This modularity enables us to extend existing classifiers easily to utilize auxiliary samples by the simple sample weighting method. The classifier to be trained does not have to be based on the generative model that is used in weight estimation: It can be based on a different generative model, or even on a discriminative model.

The classifier to be used in step 5) should be capable of dealing with weighted samples. It is not a severe limitation of our approach because many common classifiers are able to learn with a weighted error. For example, for learning a classifier that is based on an exponential family, such as that with multinomial or Gaussian class-conditional distributions, one has only to calculate weighted sufficient statistics.

Intuitively speaking, our method finds relationships between target and auxiliary classes, assigns target-class weights to each of the target and auxiliary samples, and then uses the weighted auxiliary samples for training to improve classifier performance.

4 Related works

A number of methods have been proposed for improving classifier performance by using auxiliary samples. There are three general approaches. The first approach utilizes auxiliary samples as unlabeled samples, such as semi-supervised learning [1], [5]. It ignores the auxiliary class information that can be helpful for improving classifier performance. The second approach uses the auxiliary class information, but requires the classes to be the same as the classes in the target samples, such as domain adaptation [2]. It cannot use samples with different taxonomies. The third approach uses the auxiliary class information even if the classes are different from those given to the target samples. Examples of the third approach include transfer learning, catalog integration, cross-training, ontology matching, and feature augmentation as well as our method. Transfer learning [6] utilizes problems similar to a target problem in order to improve classifier performance. Our method is different from previous transfer learning methods in that we train a target classifier using relationships between classes. Catalog integration [7] addresses the problem of integrating samples from different taxonomies into a target taxonomy, in which the samples need to be labeled with different taxonomies. In contrast, we do not assume the samples to be labeled. Cross-training [8] improves classifier performance by using samples with two taxonomies. Our method can utilize samples with more than two taxonomies. We compared our method and cross-training by text classification experiments in Section 6. Ontology matching [9] finds similar classes in different taxonomies. It is different from our method in that the similarity measure is not defined for improving classifier performance. In feature augmentation [10], [11], [12], class information in different taxonomies is augmented into each feature vector in target samples. The augmentation values are calculated by a classifier trained with auxiliary samples. Our method is different from them in that we find the correspondence between target and auxiliary classes. We compared our method and a feature augmentation method by experiments in Section 6.

The method proposed in [13] improves classifier performance by using proximity between classes in the ontology. The method and ours are similar in the sense that both methods incorporate contributions from samples with related classes. The difference is that we propose a principled procedure to calculate the similarities, or weights, so that the weighted error approximates the expected error by assuming a generative model as in (4), rather than assuming that the similarities are given.

5 Toy Example

We demonstrate performance of our method using simple synthetic data for a toy binary classification problem. In the toy problem, target samples are generated from two 100-dimensional Gaussian distributions where their means are $\boldsymbol{\mu}_{\mathrm{W}^*} = (-1, 0, \cdots, 0)$ and $\boldsymbol{\mu}_{\mathrm{E}^*} = (1, 0, \cdots, 0)$, respectively, and their covariances are identity matrices. For auxiliary samples, we consider the following four scenarios:

- **Identical** where class distributions are identical to those of the target samples, $\boldsymbol{\mu}_{\mathrm{W}} = (-1, 0, \cdots, 0)$, $\boldsymbol{\mu}_{\mathrm{E}} = (1, 0, \cdots, 0)$;
- **Diagonal** where the class relationship is correlated to that of the target samples, $\boldsymbol{\mu}_{\mathrm{NW}} = (-2^{-\frac{1}{2}}, 2^{-\frac{1}{2}}, 0, \cdots, 0)$, $\boldsymbol{\mu}_{\mathrm{SE}} = (2^{-\frac{1}{2}}, -2^{-\frac{1}{2}}, 0, \cdots, 0)$;
- **Orthogonal** where the class relationship is orthogonal to that of the target samples, $\boldsymbol{\mu}_{\mathrm{N}} = (0, 1, 0, \cdots, 0)$, $\boldsymbol{\mu}_{\mathrm{S}} = (0, -1, 0, \cdots, 0)$; and
- **Mix** where the mixture of Identical and Orthogonal scenarios, $\boldsymbol{\mu}_{\mathrm{W}} = (-1, 0, \cdots, 0)$, $\boldsymbol{\mu}_{\mathrm{E}} = (1, 0, \cdots, 0)$, $\boldsymbol{\mu}_{\mathrm{N}} = (0, 1, 0, \cdots, 0)$, $\boldsymbol{\mu}_{\mathrm{S}} = (0, -1, 0, \cdots, 0)$.

The elements of the mean vectors except the first two elements are set equal to zero and covariances are identity matrices in all scenarios. Figure 1(a) schematically shows the distribution of the target samples, and Figs. 1(b)$\sim$(e) show those of the auxiliary samples for the four scenarios considered, in which the circles represent the contours of the standard deviation of the first and second dimensions. The number of auxiliary classes is four in the Mix scenario, and two in the others. We generated 2, 4, 8, 16, 32, 64, 128, 256 target samples, 256 auxiliary samples, and 100 test samples for each class. We used the Gaussian classifier, that is, we used the error function $J(\boldsymbol{x}, t; f) = \frac{1}{2} \parallel \boldsymbol{x} - \boldsymbol{\mu}_t \parallel^2$, which amounts to assuming Gaussian with isotropic covariance as the model distribution and adopting negative log likelihood of the Gaussian as the error function.

Table 2 shows the classification accuracies achieved by our method with auxiliary samples for the four scenarios considered, and accuracies without auxiliary samples (labeled as "Target-only"). The values in the table represent the average accuracies over 100 experiments. The accuracies achieved by our method with the Identical scenario are significantly improved compared to those of the Target-only scenario, especially in the case where the number of target samples is small. When the distribution of an auxiliary class is identical to that of a target class, the auxiliary samples can be used as target samples if we find the correspondence between the target and auxiliary classes. Since finding the correspondence usually needs smaller number of samples than modeling the distribution itself, the classifier performance can be improved with the small number of target samples as is observed in this result. The accuracies achieved with the Diagonal scenario are also better than those with the Target-only scenario. This result implies that our method can improve classifier performance if there is some correlation between target and auxiliary classes, even though the distributions are not identical. When there is no correlation as in the Orthogonal scenario, our method is not helpful in improving classifier performance, and may worsen the performance. However, if there are also auxiliary classes that are correlated to the target classes as well as orthogonal classes as in the Mix scenario, our method can improve classifier performance by using samples in correlated classes and ignoring samples in orthogonal classes.

Figure 2 shows the estimated conditional probabilities $P(y|t)$. The probabilities in highly correlated classes are high, such as $(\mathrm{W}^*, \mathrm{W})$ and $(\mathrm{E}^*, \mathrm{E})$ elements in Identical and Mix scenarios, and they match with the characteristics of generative models of the auxiliary samples. This trend becomes stronger with the number of target

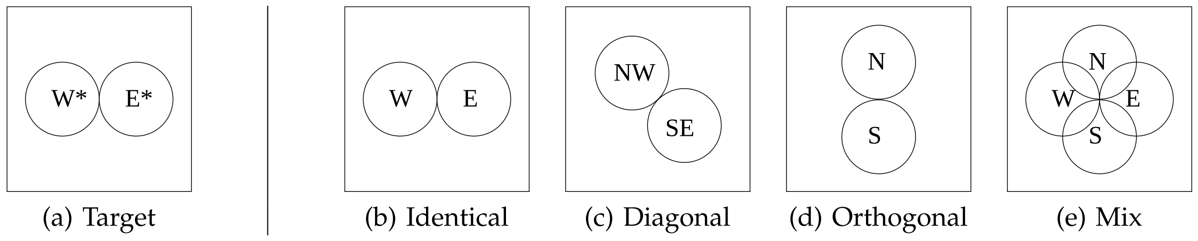

Fig. 1. Synthetic target data (a) and auxiliary data (b)∼(e). The circles represent the contours of the standard deviation of the first and second dimensions.

TABLE 2
Average accuracies (%) over 100 experiments with the synthetic data (Gaussian). The value in the parenthesis represents the standard deviation. The value is in bold face when it is better than that of Target-only.

#target samples	Target-only	Identical	Diagonal	Orthogonal	Mix
2	56.2 (4.4)	**65.0** (9.3)	**60.4** (7.7)	49.5 (4.2)	**64.0** (8.7)
4	60.9 (4.9)	**75.5** (7.4)	**65.0** (8.3)	49.9 (4.9)	**73.0** (7.7)
8	63.9 (4.3)	**80.4** (3.4)	**69.8** (6.6)	51.9 (6.2)	**78.8** (4.0)
16	68.5 (4.1)	**81.1** (2.8)	**74.2** (4.0)	59.4 (7.3)	**79.9** (2.7)
32	73.0 (3.7)	**81.2** (2.7)	**76.7** (3.5)	70.0 (5.0)	**81.6** (3.1)
64	77.6 (3.5)	**82.0** (2.7)	**78.3** (3.0)	77.2 (3.4)	**82.4** (2.4)
128	80.3 (3.0)	**82.8** (2.9)	**80.9** (2.7)	80.3 (3.4)	**82.7** (2.7)
256	82.2 (2.6)	**83.0** (2.1)	81.7 (3.1)	81.9 (3.0)	**83.0** (2.7)
average	70.3 (9.7)	**78.9** (7.5)	**73.4** (8.9)	65.0 (13.9)	**78.2** (7.8)

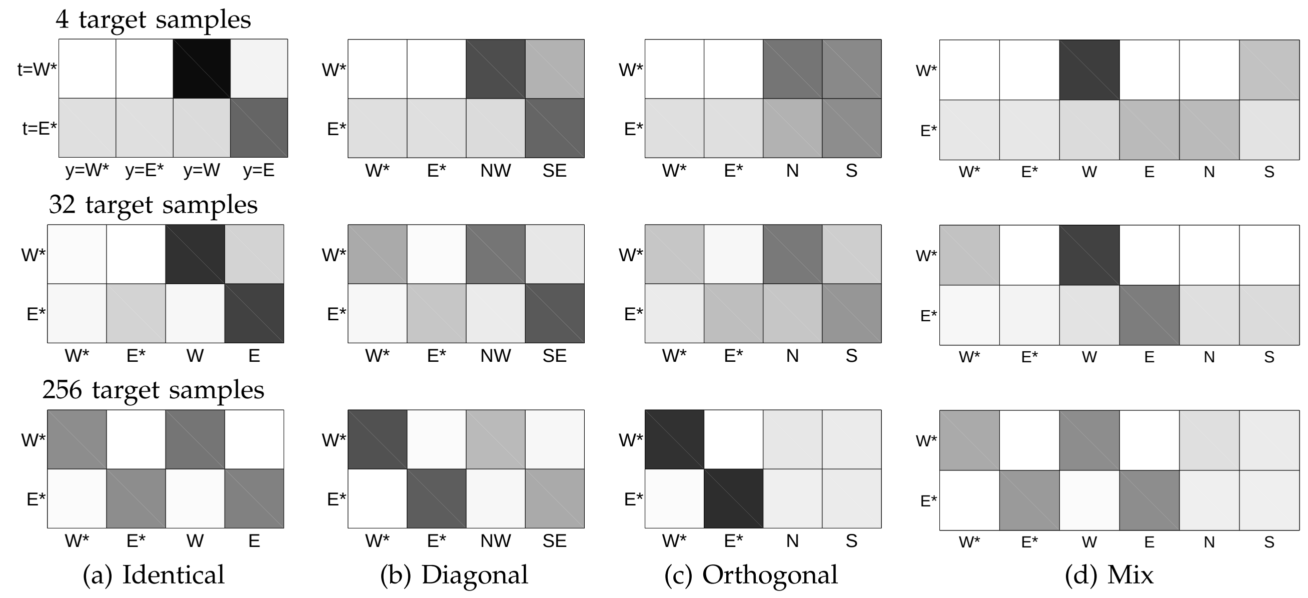

Fig. 2. Estimated conditional probabilities $P(y|t)$ for the synthetic data (Gaussian). The (t, y) element represents $P(y|t)$. Darker color indicates higher coefficient.

samples increases.

6 Application to text classification

6.1 Weight estimation

We describe application of our method to a text classification problem. We assume that a document is represented by a word count vector $\boldsymbol{x} = (x_j)_{j=1}^{V}$ where x_j is the count of word j in the document, and V is the vocabulary size of the entire collection. We use the multinomial distribution for the model distribution $P(\boldsymbol{x}|y) \propto \prod_{j=1}^{V} {\theta_{yj}}^{x_j}$ where θ_{yj} is the probability that the word j appears in a document whose class is y. The LOO maximum likelihood estimation of parameter θ_{yj} without the nth sample, denoted by $\hat{\theta}_{-n,yj}$, is obtained by:

$$\hat{\theta}_{-n,yj} = \frac{\sum_{m:y_m=y} x_{mj} - x_{nj}}{\sum_{k=1}^{V} \left(\sum_{m:y_m=y} x_{mk} - x_{nk} \right)}, \tag{12}$$

where m is a document index, $1 \leq m \leq N$. The estimates $\hat{\theta}_{-n,yj}$ are allowed to be zero by definition, but it causes difficulty in our formulation. In order to avoid the zero probability problem, we employ the following smoothing operation with a uniform distribution:

$$\tilde{\theta}_{-n,yj} = \alpha\hat{\theta}_{-n,yj} + (1-\alpha)\frac{1}{V}, \tag{13}$$

where $0 \leq \alpha < 1$ is a hyper-parameter, and use $\tilde{\theta}_{-n,yj}$ as the estimate of θ_{yj}. It should be noted that this smoothing is equivalent to using a Dirichlet prior for the multinomial parameters.

We can estimate the unknown conditional probabilities $\{\boldsymbol{P}_t\}$ and the hyper-parameter α simultaneously by maximizing the following complete-data log likelihood using the generalized EM algorithm:

$$\begin{aligned} Q_t^{(\tau)} &= \sum_{n:y_n=t}\sum_{y\in\boldsymbol{Y}} P(y|\boldsymbol{x}_n, t; \boldsymbol{P}_t^{(\tau)}, \alpha^{(\tau)}) \\ &\quad\times \Big(\log P(y|t) + \sum_{j=1}^{V} x_{nj}\log\tilde{\theta}_{-n,yj}\Big), \end{aligned} \tag{14}$$

where, and hereafter, $Q_t^{(\tau)}$ represents $Q(\boldsymbol{P}_t, \alpha|\boldsymbol{P}_t^{(\tau)}, \alpha^{(\tau)})$. The E-step and the update of conditional probabilities in the M-step follow the same procedures as the standard EM algorithm described in (10) and (11), respectively. In the M-step, we update the hyper-parameter α using the Newton method:

$$\alpha^{(\tau+1)} = \alpha^{(\tau)} - \frac{\partial Q_t^{(\tau)}}{\partial\alpha}\left(\frac{\partial^2 Q_t^{(\tau)}}{\partial\alpha^2}\right)^{-1}, \tag{15}$$

where the derivatives are evaluated as:

$$\begin{aligned} \frac{\partial Q_t^{(\tau)}}{\partial\alpha} &= \sum_{n:y_n=t}\sum_{y\in\boldsymbol{Y}} P(y|\boldsymbol{x}_n, t; \boldsymbol{P}_t^{(\tau)}, \alpha^{(\tau)}) \\ &\quad\times\sum_{j=1}^{V} x_{nj}\left(\frac{\hat{\theta}_{-n,yj} - \frac{1}{V}}{\tilde{\theta}_{-n,yj}}\right), \end{aligned} \tag{16}$$

and

$$\begin{aligned} \frac{\partial^2 Q_t^{(\tau)}}{\partial\alpha^2} &= -\sum_{n:y_n=t}\sum_{y\in\boldsymbol{Y}} P(y|\boldsymbol{x}_n, t; \boldsymbol{P}_t^{(\tau)}, \alpha^{(\tau)}) \\ &\quad\times\sum_{j=1}^{V} x_{nj}\left(\frac{\hat{\theta}_{-n,yj} - \frac{1}{V}}{\tilde{\theta}_{-n,yj}}\right)^2. \end{aligned} \tag{17}$$

Since the second-order derivative is always negative or zero, we can obtain the global optimum solution of α by iterating the update (15) in the M-step.

6.2 Text classifiers

We consider three representative text classifiers, the naive Bayes model, the maximum entropy model, and the support vector machine. The naive Bayes model assumes that each word is generated independently given a class, which is the same generative model as that used for the weight estimation. The probability of the word count vector $\boldsymbol{x}$ in the target class t is represented by:

$$P(\boldsymbol{x}|t) \propto \prod_{j=1}^{V} \phi_{tj}{}^{x_j}, \tag{18}$$

where ϕ_{tj} represents the probability that the word j occurs in the class t. When we use negative log likelihood for the error function, and a Dirichlet prior for ϕ_{tj}, the weighted error becomes:

$$\begin{aligned} E(f_{\mathrm{NB}}) = &- \frac{1}{N}\sum_{n=1}^{N}\sum_{t\in\boldsymbol{T}} w(t|y_n)\sum_{j=1}^{V} x_{nj}\log\phi_{tj} \\ &- \beta\sum_{t\in\boldsymbol{T}}\sum_{j=1}^{V}\log\phi_{tj}. \end{aligned} \tag{19}$$

Here β is a hyper-parameter characterizing the Dirichlet prior. The estimation of ϕ_{tj} that minimizes this weighted error is as follows:

$$\hat{\phi}_{tj} = \frac{\frac{1}{N}\sum_{n=1}^{N} w(t|y_n)x_{nj} + \beta}{\frac{1}{N}\sum_{k=1}^{V}\sum_{n=1}^{N} w(t|y_n)x_{nk} + \beta V}. \tag{20}$$

The maximum entropy model is a discriminative model, and it estimates a probability distribution that maximizes entropy under the constraints in the given samples. This model has been used in various research fields such as text classification [14] and collaborative filtering [15]. The maximum-entropy distribution of the target class t given the word count vector $\boldsymbol{x}$ is represented as follows:

$$P(t|\boldsymbol{x}) = \frac{\exp(\boldsymbol{\lambda}_t^\top\boldsymbol{x})}{\sum_{t'\in\boldsymbol{T}}\exp(\boldsymbol{\lambda}_{t'}^\top\boldsymbol{x})}, \tag{21}$$

where $\boldsymbol{\lambda}_t$ is an unknown parameter vector for class t, and where $\boldsymbol{\lambda}_t^\top$ represents the transpose of $\boldsymbol{\lambda}_t$. When we use negative log likelihood for the error function and Gaussian prior for $\boldsymbol{\lambda}_t$ with mean $\boldsymbol{0}$ and covariance $\gamma^{-1}\boldsymbol{I}$ [16], the weighted error becomes:

$$\begin{aligned} &E(f_{\mathrm{ME}}) \\ &= -\frac{1}{N}\sum_{n=1}^{N}\sum_{t\in\boldsymbol{T}} w(t|y_n)(\boldsymbol{\lambda}_t^\top\boldsymbol{x}_n - \log\sum_{t'\in\boldsymbol{T}}\exp(\boldsymbol{\lambda}_{t'}^\top\boldsymbol{x}_n)) \\ &\quad+\frac{\gamma}{2}\sum_{t\in\boldsymbol{T}}\parallel\boldsymbol{\lambda}_t\parallel^2. \end{aligned} \tag{22}$$

Here γ is a hyper-parameter of the Gaussian prior for $\boldsymbol{\lambda}_t$. We can estimate the unknown parameters $\{\boldsymbol{\lambda}_t\}_{t\in\boldsymbol{T}}$ by minimization via the quasi-Newton method [17]. The global optimality of the estimate is guaranteed due to the concavity of the weighted error.

The support vector machine (SVM) is a discriminative model, and it finds the decision boundary so as to maximize the margin between classes [18]. The multiclass SVM introduced in [18], [19], [20] employs the following generalized hinge loss for the loss function:

$$J_{\mathrm{hin}}(\boldsymbol{x}_n, y_n; f) = \sum_{y\neq y_n}[1 - (f(\boldsymbol{x}_n, y_n) - f(\boldsymbol{x}_n, y))]_+, \tag{23}$$

where $f(\boldsymbol{x}, y)$ is a decision function, and $[v]_+ = \max(0, v)$. Given the decision function, it classifies an input vector $\boldsymbol{x}$ to class $\hat{y}$ having the highest decision function value, $\hat{y} = \arg\max_y f(\boldsymbol{x}, y)$. In the linear kernel SVM, the decision function is represented by $f(\boldsymbol{x}, y) =$

$\boldsymbol{\lambda}_y^\top \boldsymbol{x}$. The weighted error function of the multiclass SVM with the generalized hinge loss (23) based on our framework is as follows:

$$E(f_{\mathrm{SVM}}) = \frac{1}{N}\sum_{n=1}^{N}\sum_{t\in\boldsymbol{T}} w(t|y_n) J_{\mathrm{hin}}(\boldsymbol{x}_n, t; f) + \frac{\eta}{2}\sum_{t\in\boldsymbol{T}} \parallel \boldsymbol{\lambda}_t \parallel^2, \quad (24)$$

where η is a hyper-parameter. The SVM with weights for each sample as in (24) is called the fuzzy SVM [21] or weighted SVM [22]. The optimization can be performed by solving a quadratic programming problem as described in [21], [22].

6.3 Methods compared

We considered the following nine methods for the comparison of performance in this section: NB, ME, SVM, SEMI, CT, FA, CA-NB, CA-ME, and CA-SVM.

NB is the naive Bayes model, **ME** is the maximum entropy model, and **SVM** is the linear kernel support vector machine. [1] They use only the target samples for training.

SEMI is semi-supervised learning with generative models as described in [1]. We used a mixture of multinomial distributions for the generative model, and a Dirichlet distribution for the prior. Unknown parameters were estimated by maximizing the log likelihood for labeled (target) samples and unlabeled (auxiliary) samples using the EM algorithm, where target classes in the auxiliary samples are assumed to be hidden variables. A parameter for discounting unlabeled samples were estimated by leave-one-out cross-validation.

CT is generative cross-training described in [8], in which a document is generated from a mixture of multinomial distributions over pairs of target and auxiliary classes. The parameters were estimated using the EM algorithm by maximizing the log likelihood, where auxiliary classes are assumed to be hidden variables for target samples, and target classes are assumed to be hidden variables for auxiliary samples. After this estimation, a classifier for target classes was trained by minimizing the weighted error, where auxiliary samples were weighted by posterior probability. [2] We show the results when we used NB as the classifier because NB was better than ME. We estimated a parameter for discounting auxiliary samples by cross-validation.

FA is a feature augmentation method, in which outputs of an auxiliary classifier are augmented into a feature vector for a target classifier [10]. The training procedure is as follows. First, an auxiliary classifier is trained by using auxiliary samples. Next, each feature vector of a target sample is augmented by auxiliary class posteriors that are outputs of the auxiliary classifier. [3] Finally, a target classifier is trained by using the augmented target samples. We used ME as the classifiers. FA is similar to the discriminative cross-training described in [8] although we used ME as the classifier instead of SVM.

CA-NB, **CA-ME**, and **CA-SVM** are our methods (class adaptation) using NB, ME, and SVM as the classifiers, respectively.

NB, ME, and SVM do not use auxiliary samples at all. SEMI uses auxiliary samples as unlabeled samples. CT, FA, CA-NB, CA-ME, CA-SVM use auxiliary samples as labeled samples with different taxonomies. We normalized the feature vector $\boldsymbol{x}$ for ME so that its elements sum to one. In each experiment, we sampled target samples while setting the number of samples for each class to 2, 4, 8, 16, 32, 64, 128, and 256. We used 100 test samples for each class. We evaluated performance via the average accuracies over 100 experiments. The hyper-parameters βN and γN were chosen among $\{10^{-3}, 10^{-2}, 10^{-1}, 1\}$, and ηN was chosen among $\{1, 10, 10^2, 10^3\}$ so that they maximize the accuracy for a development data generated separately from training and test data.

6.4 Data

We used the following two sets of data in the evaluations: 20News and Web.

In **20News** data, we used documents in the 20 Newsgroups corpus [23]. The corpus contains about 20,000 articles categorized into 20 discussion groups. We omitted stop-words and words that occurred only once in the entire collection. The vocabulary size was 52,647. We set the following four groups in the 20 Newsgroups corpus as the target classes: comp.graphics, rec.sport.baseball, sci.electronics, and talk.religion.misc. These target classes were selected so that they have different higher-level categories. We set the other 16 classes as the auxiliary classes, and used all samples in the auxiliary classes as auxiliary samples. No articles in the corpus were multi-posted, which means that there were no samples classified into multiple categories. The number of the auxiliary samples was 15,211.

In **Web** data, we used web pages that are categorized in goo category and yahoo category, which are Japanese directory search engines. We used those words as features that occurred more than nine times in both categories. The vocabulary size was 43,200. We set top-level 13 classes in goo category as the target classes and top-level 14 classes in yahoo category as the auxiliary classes, and used all samples in yahoo category as the auxiliary samples. The number of the auxiliary samples was 51,728.

1. We used the linear SVM because we obtained better results with the linear SVM than the nonlinear SVM in our preliminary experiments.

2. The approach is called EM2D-D in [8]. We also evaluated performance of a variant of EM2D-D, which is called EM2D-G. But the accuracy of EM2D-G was lower than that of EM2D-D in our preliminary experiments, so that it is excluded from the comparisons. This result can be explained as follows: EM2D-D could reduce data sparsity and improve the reliability of its parameter estimates as described in [8] and data used here were very sparse.

3. We can also augment a feature vector by a vector that represents the predicted label by the auxiliary classifier. However, we found that the accuracy was lower than that of the posterior augmentation in our preliminary experiments.

TABLE 3
Average accuracies (%) over 100 experiments for text classification. The value in the parenthesis represents the standard deviation.

(a) 20News

#target samples	NB	ME	SVM	SEMI	CT	FA	CA-NB	CA-ME	CA-SVM
2	52.4 (4.9)	53.7 (5.1)	52.6 (5.6)	73.0 (5.3)	74.1 (7.2)	73.7 (5.4)	**74.3** (7.3)	69.7 (7.1)	69.7 (7.7)
4	62.5 (5.3)	64.9 (4.6)	62.5 (4.8)	77.3 (4.4)	**79.9** (6.1)	77.8 (3.7)	79.0 (4.5)	72.7 (4.8)	72.0 (4.6)
8	72.9 (3.5)	73.8 (3.3)	72.7 (5.9)	79.6 (4.2)	**83.2** (3.7)	79.6 (3.3)	81.5 (3.6)	75.4 (3.7)	76.7 (3.0)
16	80.9 (2.7)	81.1 (2.3)	80.1 (2.9)	82.8 (3.4)	85.7 (2.7)	81.4 (2.2)	**86.1** (2.4)	80.8 (3.0)	79.9 (2.5)
32	87.4 (2.1)	86.6 (1.8)	86.2 (2.0)	84.9 (2.9)	87.2 (1.9)	82.0 (2.3)	**89.7** (1.8)	86.5 (2.4)	84.5 (2.3)
64	91.9 (1.3)	90.8 (1.3)	90.0 (1.5)	88.1 (2.3)	89.1 (1.6)	82.7 (2.0)	**92.9** (1.1)	91.2 (1.3)	88.9 (1.6)
128	94.7 (1.2)	93.3 (1.3)	92.2 (1.5)	91.4 (1.8)	90.9 (1.5)	83.5 (1.8)	**95.2** (1.2)	93.9 (1.2)	92.1 (1.4)
256	96.3 (1.0)	95.2 (1.0)	94.1 (1.2)	94.1 (1.6)	93.2 (1.3)	84.0 (1.6)	**96.4** (1.0)	95.8 (1.0)	94.0 (1.2)
average	79.9 (15.3)	79.9 (14.1)	78.8 (14.6)	83.9 (7.6)	85.4 (7.0)	80.6 (4.4)	**86.9** (8.3)	83.2 (10.1)	82.2 (9.3)

(b) Web

#target samples	NB	ME	SVM	SEMI	CT	FA	CA-NB	CA-ME	CA-SVM
2	20.0 (2.5)	18.5 (2.5)	16.8 (3.8)	21.3 (4.3)	21.4 (4.4)	27.5 (4.7)	**27.8** (4.6)	26.7 (4.6)	24.8 (5.1)
4	25.8 (2.4)	23.5 (2.0)	22.1 (3.7)	28.0 (4.0)	29.4 (4.3)	32.3 (2.7)	**33.4** (3.3)	31.0 (3.1)	28.6 (4.9)
8	32.7 (2.2)	29.9 (2.0)	27.1 (3.6)	33.8 (3.3)	36.1 (3.6)	36.0 (2.0)	**40.0** (2.8)	37.2 (2.7)	33.7 (4.3)
16	39.8 (1.6)	37.4 (1.7)	34.5 (4.0)	38.8 (2.9)	42.1 (2.7)	38.5 (1.5)	**45.5** (2.4)	43.4 (2.6)	39.5 (3.7)
32	46.3 (1.5)	45.0 (1.5)	40.1 (4.3)	42.5 (2.2)	46.5 (1.9)	40.9 (1.4)	**50.5** (1.6)	49.5 (1.7)	43.8 (4.4)
64	51.8 (1.6)	51.3 (1.4)	46.5 (4.0)	45.8 (2.1)	49.9 (1.7)	43.4 (1.5)	**53.7** (1.7)	**53.7** (1.7)	46.8 (3.8)
128	56.1 (1.3)	56.7 (1.4)	52.4 (4.3)	49.1 (1.8)	53.4 (1.4)	45.5 (1.4)	57.1 (1.5)	**58.5** (1.5)	51.3 (4.1)
256	59.1 (1.3)	61.1 (1.2)	57.4 (4.6)	52.6 (1.8)	56.5 (1.3)	46.8 (1.1)	59.2 (1.3)	**62.4** (1.3)	54.0 (4.3)
average	41.5 (13.5)	40.4 (14.8)	37.1 (14.2)	39.0 (10.5)	41.9 (11.8)	38.9 (6.6)	**45.9** (10.9)	45.3 (12.4)	40.3 (10.8)

TABLE 4
Average computational time (second) over 100 experiments for text classification. The rightmost column represents the computational time for the weight estimation in our class adaptation framework.

(a) 20News

#target samples	NB	ME	SVM	SEMI	CT	FA	CA-NB	CA-ME	CA-SVM	Weight Estimation
2	0.052	1.433	4.881	51.745	1.298	19.457	0.083	5.770	13.806	0.032
4	0.052	1.517	6.030	50.825	1.313	18.319	0.084	7.285	14.580	0.033
8	0.052	1.602	7.234	48.229	1.279	18.767	0.087	6.713	12.062	0.038
16	0.052	1.747	7.799	50.134	1.254	18.457	0.096	4.576	8.249	0.048
32	0.052	1.937	8.355	46.966	1.241	19.148	0.121	3.525	6.373	0.064
64	0.052	2.204	8.981	44.674	1.260	19.118	0.176	3.031	6.025	0.097
128	0.053	2.230	9.792	45.885	1.350	19.527	1.626	3.336	7.194	0.144
256	0.054	2.571	11.017	44.910	1.522	18.688	2.785	3.896	8.411	0.196

(b) Web

#target samples	NB	ME	SVM	SEMI	CT	FA	CA-NB	CA-ME	CA-SVM	Weight Estimation
2	0.893	40.533	543.972	3507.865	35.232	419.759	1.115	421.825	3903.446	0.119
4	0.897	46.866	590.406	3408.127	31.825	444.964	1.081	262.214	2499.104	0.138
8	0.889	48.958	689.073	3309.457	27.708	455.161	1.131	115.517	1051.766	0.173
16	0.890	58.922	824.476	2986.989	26.962	450.371	1.232	103.268	682.612	0.243
32	0.902	70.507	1036.839	2686.255	27.709	469.524	1.446	138.920	949.600	0.377
64	0.926	96.620	1277.893	2622.598	28.527	463.319	1.961	179.198	1456.495	0.661
128	0.945	128.851	1567.900	2411.877	29.129	455.730	3.258	237.886	2395.531	1.191
256	0.992	180.232	2059.920	2198.670	31.668	450.188	6.723	318.102	4191.200	2.263

6.5 Results

Table 3 shows the average classification accuracies on 20News and Web data. The accuracies achieved by our methods (CA-NB, CA-ME, and CA-SVM) are higher than those by the methods that use only target samples (NB, ME, and SVM), especially in the case of the small number of target samples. This result implies that our method can improve classifier performance by using auxiliary samples effectively. SEMI, CT, and FA also improve the accuracies. However, their improvements are smaller than that of our method. This can be understood from the following reasons. SEMI treats auxiliary samples as unlabeled samples and estimates their target classes; it does not use their class information at all. On the other hand, in our method the auxiliary samples are already clustered by the class labels. CT estimates a distribution for each pair of target and auxiliary classes, and the required number of models is $|\boldsymbol{T}||\boldsymbol{A}|$, where $|\cdot|$ represents the number of elements of a set. On the other hand, our method estimates a distribution for each class, and the required number of models is $|\boldsymbol{T}| + |\boldsymbol{A}|$. Therefore, our method can train classifiers more robustly with a smaller number of models to be trained than CT. FA improves the performance well when the number of target samples is small. However, as the number of target samples increases, the performance improvement of FA decreases. This result indicates that the augmented features, which are outputs of the auxiliary classifier, would have a negative effect to train classifiers because the auxiliary classes are different from the target classes.

We also performed an experiment with the following target classes: sci.crypt, sci.electronics, sci.med, and sci.space. We observed that the performance of the proposed method was not improved in this case. This situation is the same as the Orthogonal scenario in the toy example experiments presented in Section 5: Indeed, all the other classes in the corpus were with high-level categories different from sci. This experiment demonstrates that the performance will be improved only when some of the classes in different taxonomies are correlated with the target classes.

Table 4 shows the average computational time (second) on 20News and Web data. It was measured on computers having 2.66-GHz Xeon CPUs. The computational complexity for the weight estimation in our framework is small, and it took only a few second even for large data with 256 target samples per each class and 51,728 auxiliary samples in Web data. Error function $J(\boldsymbol{x}, y; f)$ is added N_T times in the objective function for only target samples (1), and it is added $N|\boldsymbol{T}|$ times in our framework (2). Therefore, our framework can be regarded as using at most $N|\boldsymbol{T}|$ samples for training. If the classifier used in our framework can handle large scale data, our framework is feasible as shown in the experiments. There are some cases where the computational time becomes shorter as the number of target samples increases. This is because that the weights estimated with large target samples are likely to be sparse, and thus, the effective number of samples decreases.

Table 5 shows the classes with high conditional probabilities $P(y|t)$ for each target class on 20News data; the number of target samples was 32. The classes that have the same parents are likely to have high coefficients such as comp.windows.x for comp.graphics target class, and rec.sport.hockey for rec.sport.baseball target class. Even if the parents are not the same, classes that are closely related have high coefficients such as soc.religion.christian and alt.atheism for talk.religion.misc target class. Similar tendencies can be observed for Web data as well, as in Table 6. For example, hobby&sport in yahoo has a high coefficient for sports&hobby&travel target class, and health&medicine in yahoo for health&medicine&beauty target class.

7 Conclusion

We have proposed a simple and promising learning framework for improving classifier performance by using auxiliary samples with different taxonomies. We have experimentally confirmed that when some of the classes in different taxonomies are correlated to the target classes, such labeled samples can significantly improve classifier performance. We have also shown experimentally the advantage of our method over conventional data augmentation methods. These results encourage us to believe that our data augmentation approach, class adaptation, will become a useful tool for designing robust classifiers.

We plan a wide-ranging exploration of the application area of our method. For example, many people put tags on many words or sentences for the wide variety of natural language processing tasks. However, these tagged data are not sufficiently utilized for similar tasks. These tagged data may improve the performance of other tasks. In our framework, weight estimation and classifier training are separated. We would like to consider the simultaneous estimation of the weights and the classifier. We used simple model distributions and classifiers in the experiments. We need to verify our framework further by applying it to other types of model distributions and classifiers. We also used simple features in our text classification experiments. Since the feature design or feature extraction might improve the classification performance further in our framework, it would be important to extend our framework to be able to extract features automatically in the future work.

References

[1] K. Nigam, A. K. McCallum, S. Thrun, and T. M. Mitchell, "Text classification from labeled and unlabeled documents using EM," *Machine Learning*, vol. 39, no. 2/3, pp. 103–134, 2000.

[2] H. Daume III and D. Marcu, "Domain adaptation for statistical classifiers," *Journal of Artificial Intelligence Research*, vol. 26, pp. 101–126, 2006.

[3] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," *Journal of Artificial Intelligence Research*, vol. 4, pp. 129–145, 1995.

[4] A. K. McCallum, R. Rosenfeld, T. M. Mitchell, and A. Y. Ng, "Improving text classification by shrinkage in a hierarchy of classes," in *ICML '98: Proceedings of the 15th International Conference on Machine Learning*, 1998, pp. 359–367.

[5] A. Fujino, N. Ueda, and K. Saito, "Semisupervised learning for a hybrid generative/discriminative classifier based on the maximum entropy principle," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 3, pp. 424–437, 2008.

[6] R. Raina, A. Y. Ng, and D. Koller, "Constructing informative priors using transfer learning," in *ICML '06: Proceedings of the 23rd International Conference on Machine learning*, 2006, pp. 713–720.

[7] R. Agrawal and R. Srikant, "On integrating catalogs," in *WWW '01: Proceedings of the 10th International Conference on World Wide Web*, 2001, pp. 603–612.

[8] S. Sarawagi, S. Chakrabarti, and S. Godbole, "Cross-training: learning probabilistic mappings between topics," in *KDD '03: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, 2003, pp. 177–186.

[9] A. Doan, J. Madhavan, P. Domingos, and A. Halevy, "Learning to map between ontologies on the semantic web," in *WWW '02: Proceedings of the 11th International Conference on World Wide Web*, 2002, pp. 662–673.

[10] R. Florian, A. Ittycheriah, H. Jing, and T. Zhang, "Named entity recognition through classifier combination," in *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*. Morristown, NJ, USA: Association for Computational Linguistics, 2003, pp. 168–171.

[11] E. Gabrilovich and S. Markovitch, "Harnessing the expertise of 70,000 human editors: Knowledge-based feature generation for text categorization," *Journal of Machine Learning Research*, vol. 8, pp. 2297–2345, 2007.

TABLE 5
High conditional probability classes for each target class on 20News data when the number of target samples for each class is 32. The value in the parenthesis represents the conditional probability.

target class	high conditional probability classes
comp.graphics	comp.windows.x (0.56), sci.space (0.19), comp.sys.ibm.pc.hardware (0.13)
rec.sport.baseball	rec.sport.hockey (0.58), rec.sport.baseball (0.13)
sci.electronics	comp.sys.ibm.pc.hardware (0.24), rec.autos (0.22), sci.space (0.13), misc.forsale (0.12)
talk.religion.misc	soc.religion.christian (0.56), alt.atheism (0.33)

TABLE 6
High conditional probability classes for each target class on Web data when the number of target samples for each class is 32. The value in the parenthesis represents the conditional probability. "g:" and "y:" indicate that the class is in goo category and yahoo category, respectively.

target class	high conditional probability classes
entertainment	y:entertainment (0.24), y:life&culture (0.22), y:computer&internet (0.21)
sports&hobby&travel	y:hobby&sports (0.59)
life&society	y:life&culture (0.26), y:regional (0.26), y:business&economics (0.23)
broadband	y:computer&internet (0.18), y:life&culture (0.17), y:regional (0.15), y:politics (0.11)
media&news&politics	g:media&news&politics (0.19), y:life&culture (0.19), y:politics (0.17)
shopping	y:business&economics (0.42), g:shopping (0.25), y:computer&internet (0.10)
education&academics&culture	y:education (0.28), y:art&humanism (0.18), y:life&culture (0.12)
science&technology	y:science&technology (0.72)
computer&internet	y:computer&internet (0.57), g:computer&internet (0.34)
document&source	y:business&economics (0.26), y:life&culture (0.18), y:computer&internet (0.16)
regional	y:regional (0.46), g:regional (0.13), y:life&culture (0.11), y:business&economics (0.10)
business&economics	y:business&economics (0.59)
health&medicine&beauty	y:health&medicine (0.75)

[12] ——, "Wikipedia-based semantic interpretation for natural language processing," *Journal of Artificial Intelligence Research*, vol. 34, pp. 443–498, 2009.

[13] G. Pandey, C. L. Myers, and V. Kumar, "Incorporating functional inter-relationships into protein function prediction algorithms," *BMC Bioinformatics*, vol. 10, no. 142, 2009.

[14] K. Nigam, J. Lafferty, and A. McCallum, "Using maximum entropy for text classification," in *Proceedings of IJCAI-99 Workshop on Machine Learning for Information Filtering*, 1999, pp. 61–67.

[15] T. Iwata, K. Saito, and T. Yamada, "Recommendation method for extending subscription periods," in *KDD '06: Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Miniining*, 2006, pp. 574–579.

[16] S. F. Chen and R. Rosenfeld, "A Gaussian prior for smoothing maximum entropy models," Computer Science Department, Carnegie Mellon University, Tech. Rep. CMUCS–99–108, 1999.

[17] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Mathematical Programming*, vol. 45, no. 3, pp. 503–528, 1989.

[18] V. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.

[19] E. J. Bredensteiner and K. P. Bennett, "Multicategory classification by support vector machines," *Computational Optimization and Applications*, vol. 12, no. 1-3, pp. 53–79, 1999.

[20] J. Weston and C. Watkins, "Support vector machines for multi-class pattern recognition," in *Proceedings of the Seventh European Symposium On Artificial Neural Networks*, 1999, pp. 219–224.

[21] C.-F. Lin and S.-D. Wang, "Fuzzy support vector machines," *IEEE. Transactions on Neural Networks*, vol. 13, no. 2, pp. 464–471, 2002.

[22] X. Yang, Q. Song, and Y. Wang, "Weighted support vector machine for data classification," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 21, no. 5, pp. 961–976, 2007.

[23] K. Lang, "NewsWeeder: learning to filter netnews," in *ICML '95: Proceedings of the 12th International Conference on Machine Learning*, 1995, pp. 331–339.

PLACE PHOTO HERE

Tomoharu Iwata received the B.S. degree in environmental information from Keio University in 2001, the M.S. degree in arts and sciences from the University of Tokyo in 2003, and the Ph.D. degree in informatics from Kyoto University in 2008. He is currently a researcher at Learning and Intelligent Systems Research Group of NTT Communication Science Laboratories, Kyoto, Japan. His research interests include data mining, machine learning, information visualization, and recommender systems.

PLACE PHOTO HERE

Toshiyuki Tanaka received the B. Eng., M. Eng., and Dr. Eng. degrees in Electronics Engineering from the University of Tokyo, Tokyo, Japan, in 1988, 1990, and 1993, respectively. From 1993 to 2005, he was with the Department of Electronics and Information Engineering at Tokyo Metropolitan University, Tokyo, Japan. He is currently a professor of the Graduate School of Informatics, Kyoto University, Kyoto, Japan.

His research interests are in the areas of neural networks, machine learning, and information and communication theory. He received DoCoMo Mobile Science Prize in 2003, and Young Scientist Award from the Minister of Education, Culture, Sports, Science and Technology, Japan, in 2005.

PLACE PHOTO HERE

Takeshi Yamada received the B.S. degree in mathematics from the University of Tokyo in 1988 and the Ph.D. degree in informatics from Kyoto University in 2003. He was a Leader of Emergent Learning and Systems Research Group of NTT Communication Science Laboratories and is currently a Senior Manager of NTT Science and Core Technology Laboratory Group. His research interests include Data Mining, Statistical Machine Learning, Graph Visualization, Metaheuristics and Combinatorial Optimization. He is a member of the ACM and IEEE.

PLACE PHOTO HERE

Naonori Ueda received the B.S., M.S., and Ph D degrees in Communication Engineering from Osaka University, Osaka, Japan, in 1982, 1984, and 1992, respectively. In 1984, he joined the Electrical Communication Laboratories, NTT, Japan, where he was engaged in research on image processing, pattern recognition, and computer vision. In 1991, he joined the NTT Communication Science Laboratories, where he has invented a significant learning principle for optimal vector quantizer design and has developed some novel models and learning algorithms. His current research interests include parametric and non-parametric Bayesian approach to machine learning, pattern recognition, data mining, signal processing, and cyber-physical systems. From 1993 to 1994, he was a visiting scholar at Purdue University, West Lafayette, USA. Currently, he is a director of NTT Communication Science Laboratories. He is an associate editor of Neurocomputing and Journal of Neural Networks, and is a fellow of the Institute of Electronics, Information, and Communication Engineers (IEICE).