

# The Set Classification Problem and Solution Methods \*

Xia Ning<sup>†</sup>

George Karypis<sup>‡</sup>

## Abstract

This paper focuses on developing classification algorithms for problems in which there is a need to predict the class based on multiple observations (examples) of the same phenomenon (class). These problems give rise to a new classification problem, referred to as *set classification*, that requires the prediction of a set of instances given the prior knowledge that all the instances of the set belong to the same unknown class. This problem falls under the general class of problems whose instances have class label dependencies. Four methods for solving the set classification problem are developed and studied. The first is based on a straightforward extension of the traditional classification paradigm whereas the other three are designed to explicitly take into account the known dependencies among the instances of the unlabeled set during learning or classification. A comprehensive experimental evaluation of the various methods and their underlying parameters shows that some of them lead to significant gains in performance.

## 1 Introduction

Since the early days of machine learning research, classification (or the more general topic of supervised learning) has been a fundamental problem for which various formulations and solution approaches have been developed. The majority of these methods learn from a training set a model that is used to predict the class of unlabeled instances and assume that each labeled and unlabeled instance is independent of each other. However, in some cases there are dependencies between the class labels of the instances that, when exploited, can significantly improve the classification accuracy. Examples of such dependencies include the spatial auto-correlation present in many spatial datasets, correlation between the structural and functional properties of adjacent positions in DNA and protein sequences, and correlation between the topics of web-pages connected via hyperlinks.

In this paper we focus on another type of dependency that exists among the class labels of a set of unlabeled

instances. Specifically, we focus on the problem in which we need to predict the class of a set of unlabeled instances with the prior knowledge that all the instances in the set belong to the same (unknown) class. Thus, the problem becomes that of classifying the entire set and we will refer to it as the *set classification* (SC) problem. To the best of our knowledge, this is the first time that the set classification problem has been explicitly proposed and formulated.

The set classification problem arises in many applications in which there is a need to predict the class based on multiple observations of the same phenomenon (class). One such application is face recognition based on pictures obtained from different cameras (e.g., different perspectives and lighting conditions) or multiple stills obtained from video (e.g., face recognition in video streams) [2, 26]. In this setting, the multiple pictures of the same person correspond to the set that needs to be classified whereas the different individuals correspond to the classes. Another application is from the analysis of high-content screening phenotypic assays in drug discovery for identifying the protein target of small organic compounds [34, 23]. High-content phenotypic assays is an experimental methodology in which a large number of compounds (potential drug candidates) are tested to determine whether or not they induce a desired *in vitro* phenotype. The set of compounds that induce the phenotype are then analyzed to identify the protein (or class of proteins) that they bind to<sup>1</sup>. The underlying assumption is that all compounds that induce the same desired phenotype do so by targeting the same protein or class of proteins, and therefore they should be assigned the same class label. Set classification can also be applied in ontology mapping [10] problems. Ontologies are formal conceptualization which describe a certain domain. For example, in semantic web, ontologies are used to provide semantics of web pages, and in bioinformatics, gene ontologies are used to describe protein function annotations. A same domain may have multiple overlapping ontologies so that there is a need to map ontologies in order for different parties using different ontologies upon the domain to understand each other. A concept from one ontology can be mapped to a concept from another ontology by classifying the set of instances described by the first concept using models learned from the instances of the

\*Supported by IIS-0431135, NIH RLM008713A, and by the Digital Technology Center at the University of Minnesota

<sup>†</sup>Department of Computer Science& Engineering, University of Minnesota, Twin Cities. xning@cs.umn.edu

<sup>‡</sup>Department of Computer Science& Engineering, University of Minnesota, Twin Cities. karypis@cs.umn.edu

<sup>1</sup>The problem of protein target identification is often referred to as *target fishing* (TF) [16].

second concept, giving rise to the set classification problem.

A straightforward way to solve the set classification problem is to learn a classification model from the single instances, use it to predict each one of the instances of the unlabeled set, and then derive a prediction for the set by taking into account the consensus predictions of the individual instances, e.g., assign the set to the most frequently predicted class. This approach can be applied when the instances are suitably independent. However, when there are high dependencies across instances, or most of the instances are multi-labeled, such an approach would have two limitations. First, during learning, it does not take into account the known dependencies among the instances of the unlabeled set, and second, during prediction, the set's instances are predicted independently without considering the predictions of other instances from the same set.

In this work we present three classes of methods for solving the set classification problem that are designed to overcome these problems. The first method, referred to as *subset learning*, learns models based on training examples corresponding to subsets of instances. The second method, referred to as *model-based*, builds a classification model for the instances of the unlabeled set and assigns it to the class with the most similar model. Finally, the third method, referred to as *co-learning*, determines the class of the unlabeled set by assessing the performance improvements achieved by incorporating the unlabeled instances as additional positive examples for each class. The subset learning method is designed to incorporate into learning information about the set of instances, whereas the model-based and co-learning methods are designed to take into account the entire set of unlabeled instances during classification.

We present a comprehensive experimental evaluation of the various parameters underlying these methods on seven different datasets and compare their performance against the set classification method that is based on consensus predictions. Our experiments show that the subset learning and model-based methods performance considerably better than consensus prediction. They achieve improvements ranging from 5.7% to 38.8% on six out of the seven datasets and an overall improvement of around 10% averaged over all seven datasets.

The rest of the paper is organized as follows. Section 2 formally defines the set classification problem. Section 3 reviews some related learning problems. Section 4 describes the various methods that we developed for solving the set classification problem. Section 5 describes the datasets and our experimental methodology. Section 6 presents the experimental evaluation of the various methods. Finally, Section 7 provides some concluding remarks and directions for future research.

## 2 Set Classification Problem

In the set classification problem we are given a set of  $l$  unlabeled instances  $U_j = \{u_{j_1}, \dots, u_{j_l}\}$  and we want to assign this set (i.e., all of its instances) to one of  $k$  possible classes from a pre-identified set  $\mathcal{C} = \{c_1, \dots, c_k\}$  of classes. We have no control over the size of the set  $l$ , it can contain an arbitrary number of instances, and it can vary from problem to problem. The information provided for training the classification model is a set of  $n$  instances  $\mathcal{I} = \{x_1, \dots, x_n\}$  and for each instance  $x_i$  we are given a non-empty set of classes  $\{c_{i_1}, \dots, c_{i_m}\}$  from  $\mathcal{C}$  that  $x_i$  belongs to (i.e., instances can be multi-labeled), and we have no control over the number of labels that each instance is assigned to. This information is identical to that provided for traditionally training single- and multi-label classification instances.

The motivation behind focusing on the set classification problem is to develop new classification methods, which by taking into account the a priori information that all the unlabeled instances in  $U_j$  belong to the same class, they can improve the overall classification. However, the extent to which knowledge of this a priori information can improve upon existing single-instance methods that, for example, classify each instance individually and then derive a consensus prediction for the set  $U_j$ , will depend on the characteristics of the dataset. For example, if the instances are single-labeled and the different classes are highly separable, then there is probably little benefit to be obtained by utilizing the set classification problem's a priori information. On the other hand, if the instances are multi-labeled and/or the classes are not well-separated, then better methods can potentially be developed to exploit during learning and/or classification the fact that a set of same-labeled instances need to be predicted.

## 3 Related Learning Problems

Set classification is different from traditional classification (also referred to as single instance learning (SIL)) as it imposes constraints on the types of predictions allowed (i.e., it is to classify a set of instances as a whole, and all its instances need to be assigned to the same class label). These constraints require a learner or the prediction methodology to properly exploit the additional information so as to achieve better prediction accuracy. Set classification is also distinct from multiple instance learning and collective classification, which are the two other learning problems involving sets of instances that we are aware of. We briefly review these two problems, aiming to illustrate that the SC problem we propose in the paper is different from these existing problems.

Multiple-Instance Learning (MIL) was initially motivated for the prediction of the activity of a chemical compound based on its multiple geometric conformations [8] and since then it has been applied in other domains such as doc-

ument and video classification [1]. The goal of MIL is to learn a concept from sets of instances so as to classify unlabeled sets. A set is defined as positive if at least one of its instances is positive and as negative if all of its instances are negative. Unlike the SIL problem in which each training instance has an unambiguous label and learning is performed on instances, in MIL each positive set can contain both positive and negative instances and the learning is performed on sets. Over the years many MIL algorithms have been proposed [8, 12, 22, 31, 36] and among them, the best performing approaches are based on SVM [12, 36]. Also, the original MIL problem has been recently extended for the problem of multiple-instance multi-label learning (MIML) [35], in which each set can have multiple labels.

Collective Classification (CC), also known as Relational Classification (RC), is designed to classify a set of relational instances collectively and is motivated by the observation that related instances are more likely to share common class labels [17]. Based on this, CC classifies individual instances with properties from related instances taken into account simultaneously so as to boost classification performance. In CC, a well-defined relation is required across instances. For example, in the context of hyperlinked web pages, “hyperlinked” can be considered as a relation between the web pages. A number of CC algorithms have been developed [5, 21, 28], particularly for web page mining. These algorithms usually infer relational features from predicted class labels of related data and then iteratively update the class labels until some criteria are satisfied. Many popular inference techniques like Gibbs sampling and belief propagation can be adapted.

#### 4 Approaches for Set Classification

We developed various approaches for solving the set classification problem, which are described in the rest of this section. These approaches were developed within the context of binary Support Vector Machines (SVM) [29]. SVM represents one of the current state-of-the-art supervised learning methods that outperforms other approaches on a wide-range of applications, including those used to evaluate the set classification methods developed in this paper [18, 15, 7]. Moreover, SVM is able to easily handle high-dimensional datasets, which is a characteristic of many of the datasets under consideration, and therefore we can implicitly avoid issues related to feature selection and dimensionality reduction. However, the ideas underlying the approaches that we developed are not directly tied with SVMs and can be easily used with other supervised machine learning methods (e.g., decision tree [24]), as well.

**4.1 Consensus Prediction** A straightforward way to predict the class of an unlabeled set  $U_j = \{u_{j_1}, \dots, u_{j_l}\}$  is to predict its instances and then based on them, compute a pre-

diction for the set. Specifically, for each class  $c_i$ , a binary SVM model  $M_i$  is built using a one-vs-rest training framework that is commonly employed for multi-class and multi-label classification problems. Each instance of an unlabeled set  $U_j$  is predicted by each one of these  $k$  binary models. If  $M_i(u_{j_q})$  is the SVM prediction on instance  $u_{j_q}$  with respect to class  $c_i$ , then the prediction for the set  $U_j$  with respect to class  $c_i$  is given by

$$(4.1) \quad M_i(U_j) = \frac{1}{l} \sum_{q=1}^l M_i(u_{j_q}),$$

which is nothing more than the average SVM prediction for the set of instances in  $U_j$ . Note that this approach has converted the instance-level predictions of the  $k$  one-vs-rest binary models into set-level predictions. These latter predictions can then be used to obtain the final classification by assigning the set to the class that has the maximum  $M_i(U_j)$  value, which is the most commonly used approach for deriving multi-class predictions from a set of binary one-vs-rest predictors. We refer to this approach as *consensus prediction* and will provide its performance as the baseline that the other methods developed in this paper will try to improve upon.

Note that our consensus prediction method takes into account the actual outputs of the SVM models (i.e., real values). Alternate methods to compute the consensus can be by first removing predictions that are deemed to be outliers (e.g., eliminating some of the lowest and/or highest predictions) and only utilizing the rest instance-level predictions, or can be based on binary class assignments (i.e., it is a member of class  $c_i$  or not) so as to accommodate machine learning methods that predict a binary class (i.e., boolean values). In our work we did not explore such alternatives but it may be an interesting area for future research.

**4.2 Learning from Subsets** The consensus prediction approach solves the set classification problem by taking into account its special structure only during the classification phase but it does not utilize the fact that we need a consistent prediction for a set of instances during the model learning phase. To incorporate such information during learning, we developed an approach in which subsets of the original instances are combined together to create the actual training set. In these methods, the training set consists of a set of examples, each containing a subset of the original instances. The rationale behind this is that a carefully constructed subset from original instances of the same class is expected to coherently represent and highlight the class-related features more strongly than single instances. We will refer to these examples as *instance-subsets* and the resulting learning methods as *subset learning*.

The overall learning and prediction framework of these

methods is similar to that employed by the consensus prediction approach described earlier. As before,  $k$  binary one-vs-rest SVM models are trained on the instance-subsets. For a particular class  $c_i$ , a set of positive instance-subsets is generated from the positive instances of  $c_i$  and a set of negative instance-subsets is generated from its negative instances. These instance-subsets are then used to train a model with respect to class  $c_i$ . During classification, a set of instance-subsets is constructed from the instances of the unlabeled set  $U_j$ . All these instance-subsets are predicted by each one of the  $k$  binary models and the prediction for set  $U_j$  is obtained by comparing the average of the instance-subset predictions using Equation 4.1 with  $u_{j_q}$  replaced by instance-subsets.

The next two sections describe two key parameters of the subset learning methods: (i) the approaches that we developed for generating the instance-subsets and (ii) the kernel functions for that we developed for computing the similarity between instance-subsets in the context of SVM learning.

**4.2.1 Generation of Instance-Subsets** Given a set of  $p$  instances  $X = \{x_1, \dots, x_p\}$ , our initial idea for generating instance-subsets was to use all possible subsets of  $X$  up to a certain size  $s$  ( $s = 2-5$ ). This approach has the advantage of uniformly capturing higher-order relationships between the training set instances, and also it is not tied to the size of  $X$  itself (as long as it is greater than  $s$ ), which is important since during prediction we need to be able to generate instance-subsets for arbitrary size sets. However, we quickly realized that even for small values of  $s$  (e.g.,  $s \leq 3$ ), this would have led to a dramatic increase in model training time, as the size of the training set increases exponentially on  $s$ . In addition, high-order sets themselves are computationally expensive to determine in our settings. For these reasons, we developed approaches to generate instance-subsets by selecting only certain subsets of the instances.

For each instance  $x_i \in X$ , we identify  $m$  instance subsets of size  $s-1$  from  $X$ . From each of the  $m$  subsets, an instance-subset is created by including  $x_i$  along with its  $s-1$  instances. Thus, each instance leads to  $m$  instance-subsets of size  $s$ . We developed three different schemes to identify these subsets. The first, referred to as *random*, constructs each one of the  $m$  subsets by randomly selecting a set of  $s-1$  instances from  $X - \{x_i\}$  without replacement. The second, referred to as *nearest-neighbor*, constructs each of the  $m$  subsets by selecting a set of  $s-1$  most similar instances to  $x_i$  that has not yet been selected. Finally the third, referred to as *furthest-neighbor*, constructs each of the  $m$  subsets by selecting a set of  $s-1$  least similar instances to  $x_i$  that has not yet been selected. These three methods are designed to generate instance-subsets with distinct characteristics. The random scheme is designed to provide a computationally efficient approximation of our original idea of uniformly

sampling the different subsets, the nearest-neighbor scheme is designed to generate instance-subsets that capture local clusters of instances, whereas the furthest-neighbor scheme is designed to capture the diversity among the instances that belong to the same class.

Note that the nearest- and furthest-neighbor schemes select the instances based only on their similarity to  $x_i$ . An alternate way of selecting these subsets is to also try to maximize or minimize their pairwise similarity as well. We decided not to pursue such an approach due to the increased computational complexity associated with it when the value of  $s$  becomes large.

**4.2.2 Kernels for Instance-Subsets** We developed three different approaches for using the instance-subsets in SVM-based learning. The first two approaches represent each instance-subset by a feature vector derived from the features of its constituent instances and then employ standard kernel functions that operate on instances represented via feature vectors. We used two approaches for obtaining the feature-vector of an instance-subset. The first represents each instance-subset by a feature vector derived from the features that are present in all of its constituent instances, whereas the second represents each instance-subset by a feature vector derived from the union of all the features of its constituent instances. We will denote the kernels derived from these two approaches as  $\mathcal{K}_{\text{AND},\mathcal{K}}$  and  $\mathcal{K}_{\text{OR},\mathcal{K}}$ , respectively.  $\mathcal{K}_{\text{AND},\mathcal{K}}$  attentively performs a simple feature selection process assuming the conserved features are responsible for the class assignment.  $\mathcal{K}_{\text{OR},\mathcal{K}}$  relaxes this assumption and takes into consideration the diversity of class instances.

The third scheme uses instance-subsets in SVM-based learning by defining a kernel function on the instance-subsets, denoted by  $\mathcal{K}_{\text{avg},\mathcal{K}}$ , that is based on the average instance-level kernel values between all pairs of instances. Specifically, given two instance-subsets  $U_i$  and  $U_j$ ,  $\mathcal{K}_{\text{avg},\mathcal{K}}$  is given by

$$(4.2) \quad \mathcal{K}_{\text{avg},\mathcal{K}}(U_i, U_j) = \frac{1}{|U_i||U_j|} \sum_{\substack{u_{i_p} \in U_i \\ u_{j_q} \in U_j}} \mathcal{K}(u_{i_p}, u_{j_q}).$$

The function  $\mathcal{K}$  in the above equation and in the notations for  $\mathcal{K}_{\text{AND},\mathcal{K}}$  and  $\mathcal{K}_{\text{OR},\mathcal{K}}$  represents any valid kernel function defined on the feature-vector representation of a pair of instances. Note that the nearest neighbors or the furthest neighbors of a certain instance are also generated based on  $\mathcal{K}$  as a similarity measure. Compared to  $\mathcal{K}_{\text{AND},\mathcal{K}}$  and  $\mathcal{K}_{\text{OR},\mathcal{K}}$ ,  $\mathcal{K}_{\text{avg},\mathcal{K}}$  attempts to average the conservation and diversity of the instances simultaneously. In our experiments, depending on the dataset, we used two different kernels functions for  $\mathcal{K}$ . The first is the standard radial-basis kernel function and the second is the kernel function corresponding to the extended Jacquard similarity that it is commonly referred to as the

Tanimoto coefficient in the Cheminformatics literature [32]. We will denote these two kernels by *rbf* and *tc*, respectively.

**4.3 Model Comparison** We also developed an entirely different set of methods for predicting the class of an unlabeled set  $U_j = \{u_{j_1}, \dots, u_{j_l}\}$  that explicitly leverage the fact that all the instances of  $U_j$  are part of the same class. These approaches treat the instances of  $U_j$  as the positive examples of an unknown class  $c_{i'}$ , build a SVM-model  $M_{i'}$  for that class and then compare  $M_{i'}$  against the  $k$  models  $\{M_1, \dots, M_k\}$  that were built for each one of the classes in  $\mathcal{C}$ . The class of  $U_j$  is determined as the class  $c_i$  whose model  $M_i$  is most similar to model  $M_{i'}$ . The motivation behind these approaches is that the characteristics of the different classes are encapsulated into their binary classification models and by comparing the models directly we take into account the entire set of instances that belong to each class.

There are two issues that need to be addressed in the above framework, one of which is to create the negative instances for training the  $M_{i'}$  model, and the other is to determine the similarity between two models. These issues are discussed in the rest of this section.

**4.3.1 Negative Training Instances** Even though the set  $U_j$  will provide the positive instances of class  $c_{i'}$ , a set of negative instances also need to be identified in order to learn the binary SVM model  $M_{i'}$ . One way of constructing the negative class is to randomly sample instances from the training set. However, this method can potentially create problems when the true class of  $U_j$  is  $c_i$  but some instances of  $c_i$  may be sampled as negative instances to build  $M_{i'}$ .

To address this problem, our methods, instead of building a single  $M_{i'}$  model, build  $k$  models  $M_{i',1}, \dots, M_{i',k}$  and compare model  $M_i$  against model  $M_{i',i}$ . The difference between these  $k$  models  $M_{i',i}$  is how they select the instances for the negative class. The negative class for model  $M_{i',i}$  is constructed so that it does not contain any instances from class  $c_i$ . We achieve this by splitting the training set into two different subsets. One to be used for training the  $M_i$  models and the second for selecting negative instances to train the  $M_{i',i}$  models for the unlabeled set.

**4.3.2 Determining Model Similarity** We investigated two different techniques for comparing a pair of SVM models. The first approach, referred to as *direct model similarity* and denoted by *dsim*, is based on directly comparing the decision hyperplanes of the learned models. SVM's decision function is given by

$$f(x) = \text{sgn} \left( \sum_{x_i \in SV} \alpha_i y_i \mathcal{K}(x, x_i) + \beta \right),$$

where  $x_i$  is a training instance that is part of the support vectors  $SV$ ,  $y_i$  is its class (+1/-1), and  $\alpha_i$  and  $\beta$  are weights

learned during training such that  $\alpha_i \geq 0$  and  $\sum \alpha_i y_i = 0$ . Thus, the model learned is entirely determined by the support vectors and the various weights. Since for most kernel functions, it is hard to have a closed form solution of the decision hyperplane, one way of comparing the models learned by two different SVM models is to use one to classify the other's support vectors and measure the classification performance. Specifically, given two models  $M_i$  and  $M_j$ , we measure their similarity using the following formula: (4.3)

$$\text{dsim}(M_i, M_j) = \sum_{\substack{x_{i_p} \in SV_i \\ x_{j_q} \in SV_j}} \alpha_{i_p} \alpha_{j_q} y_{i_p} y_{j_q} \mathcal{K}(x_{i_p}, x_{j_q}),$$

where  $SV_i$  and  $SV_j$  are the support vectors for models  $M_i$  and  $M_j$ , respectively. Note that the various terms associated with the  $\beta$  variables are eliminated because  $\sum \alpha_{i_p} y_{i_p} = 0$ . It is easy to see that the direct similarity between a pair of models will be high if each support vector of  $M_i$  is predicted correctly by  $M_j$ 's model. Using this similarity measure, the class of an unlabeled set  $U_j$  is given by  $\text{argmax}_i(\text{dsim}(M_{i',i}, M_i))$ .

The second approach, referred to as *cross-classification similarity* and denoted by *xsim*, compares two SVM models  $M_{i',i}$  and  $M_i$  by using  $M_{i',i}$  to classify the data that were used to train model  $M_i$  and then determines their similarity based on the classification performance (e.g., accuracy). The idea behind this approach is that if two models are similar, then they are expected to be able to correctly classify each other's training data. Note that due to the way that the various  $M_i$  and  $M_{i',i}$  models are being trained, each pair of corresponding models does not share any instances thus ensuring that the above procedure measures the performance of a model on an entirely independent dataset.

**4.4 Co-Learning** Finally, we developed a method for solving the set classification problem that borrows ideas from semi-supervised learning [3, 6]. In this approach, referred to as *co-learning*, the instances of an unlabeled set  $U_j$  of unknown class  $c_{i'}$  are used as additional positive instances to train a new model for class  $c_i$  denoted by  $M_{i+i'}$ . The performance of  $M_{i+i'}$  is then evaluated on a separate validation set and is compared against the performance obtained on the same validation set from the model  $M_i$  trained only on the original positive instances for  $c_i$ . If the instances of  $U_j$  belong to class  $c_i$ , then the performance achieved by  $M_{i+i'}$  should be better (or at least not worse) than that achieved by  $M_i$  as it is now trained using a larger training set. On the other hand, if  $U_j$  does not belong in  $c_i$ ,  $M_{i+i'}$  should achieve a lower performance as it was trained using incorrectly labeled positive instances (i.e., the instances from  $U_j$ ). Thus, by measuring the difference in performance between  $M_{i+i'}$  and  $M_i$  we can potentially determine if  $U_i$  belongs in  $c_i$  or not. Based on this, the co-learning method for solving

the set classification problem assigns an unlabeled set  $U_j$  to the class

$$(4.4) \quad \operatorname{argmax}_i (f(M_{i+i'}) - f(M_i)),$$

where  $f$  is a function that measures the classification performance of a model (e.g., classification accuracy, AUC, etc.).

## 5 Experimental Design

**5.1 Datasets** We used seven different datasets to evaluate the performance of the various set classification methods developed in this paper. Various characteristics of these datasets are shown in Table 1.

The first dataset, referred to as *face*, contains 1,012 face images of 20 people under different lighting conditions and different facial expressions. This dataset was developed by the Vision Information & Processing group at the University of Manchester [13]. The actual images are  $220 \times 220$  pixels in 256 shades of gray. We represented each image by a feature vector corresponding to the concatenation of all pixel values and each vector was normalized to be of unit length. Note that even though more sophisticated feature extraction methods can be utilized to capture structural characteristics of the faces, due to time constraints we decided not to utilize them. The second dataset, referred to as *tf*, contains a set of compounds that are experimentally tested to be active against specific protein targets. This set is constructed from PubChem<sup>2</sup>, DrugBank[33], PDSP  $K_i$  database[25], BindingDB[20] and KEGG BRITE<sup>3</sup>. There are totally 9,843 compounds against 55 protein targets in this dataset. We used the AFGEN [30] software to generate features for each compound. These features are the counts of the different connected subgraphs up to seven edges that occur in the compounds' chemical graph. The third dataset, referred to as *scene*, contains a set of images that was developed for evaluating the performance of multi-label scene classification methods [4]. It contains 2,407 images belonging to six different classes. For each image, the dataset provides a 294-dimensional feature vector generated from LUV space describing various aspects of spatial and color information [4]. In our study we used these features and did not generate any additional ones. The fourth dataset, referred to as *tmc*, is a document classification dataset used in SIAM Text Mining Competition 2007<sup>4</sup>. It has 28,596 documents belonging to 22 classes with 30,438 terms in total. Each document was represented by its binary term vector normalized to be of length one. The fifth dataset, referred to as *media*, is a subset of the dataset used in the MediaMill Challenge Problem [27] and is related to the identification of certain semantic concepts present in the video stream. The binary clas-

sification problems were combined into a multi-label classification problem<sup>5</sup>, and we used the video features generated in [27] without modification. The *media* dataset consists of 11,726 instances, 60 distinct classes, and each instance is represented by a 120-dimensional feature vector. The sixth dataset, referred to as *rcv1*, is a subset from Reuters Corpus Volume I collection for text categorization [19]. It contains 6,000 news articles, 54 classes, and a total of 47,236 pre-generated features (i.e., articles are provided in vector-space form). Finally, the seventh dataset, referred to as *yeast*, is taken from [9]. It contains 14 classes, 2,417 yeast genes, each of which is represented via 103 gene-expression and phylogenetic features.

**5.2 Performance Evaluation** The performance of the different methods was evaluated using a 3-fold cross validation framework in which the entire set of instances was randomly split into three sets. For each one of the three splits, the set of test instances that belonged to the same class was considered as an unlabeled set to be predicted by the various methods developed in this paper. Thus, if the dataset had  $k$  classes, each split required the prediction of  $k$  unlabeled sets. Note that since in some of the datasets, each instance belonged to multiple classes (i.e., it is a multi-labeled dataset), the  $k$  unlabeled sets to be predicted will overlap.

Since each one of the four methods described in Section 4 computes a prediction score between an unlabeled set  $U_j$  and each class in  $\mathcal{C}$ <sup>6</sup>, the performance of a method was evaluated by looking at the rank of the true class in the ranked list of the  $k$  classes sorted in decreasing prediction score. We used the average uninterpolated precision [14], denoted by UPrec, over the three cross-validation sets. Since we measure the performance of just a single instance, the uninterpolated precision is nothing more than  $1/\text{rank}$  of the true class. If a method always ranks the true class at the top, its UPrec value will be 1.0.

Note that depending on the method, these prediction scores are computed differently. For the consensus prediction and subset learning these scores correspond to the scores of the  $k$  binary SVM classifiers, for the direct model similarity the scores are given by Equation 4.3, for the cross-classification method the score is given by evaluating the performance of a model, and for the co-learning method the score is given by the looking at the difference in the classification performance between a pair of models (Equation 4.4). In our experiments, the performance of a model was measured using the area under the ROC curve (AUC) that plots the true positive rate vs the false positive rate for different classification thresholds [11]. We also experimented with other methods for measuring the performance (e.g., accu-

<sup>2</sup><http://pubchem.ncbi.nlm.nih.gov/>

<sup>3</sup><http://www.genome.ad.jp/kegg/brite.html>

<sup>4</sup><http://www.cs.utk.edu/tmw07/>

<sup>5</sup><http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel.html>

<sup>6</sup>The methods assign  $U_j$  to the class that achieves the highest score.

Table 1: Dataset characteristics.

Dataset	# classes	# instances	avg class size	avg classes/instance	# features	avg self-class sim	avg cross-class sim
face	20	1,012	50.6	1	48,400	0.66553	0.56486
tf	55	9,843	213.4	1.19	158,520	0.27679	0.12377
scene	6	2,407	430.8	1.07	294	0.00274	0.00014
tmc	22	28,596	2,804.9	2.16	30,438	0.13899	0.12508
media	60	11,726	290.4	1.59	120	0.52686	0.43674
rcv1	54	6,000	341.2	3.07	47,236	0.03616	0.01100
yeast	14	2,417	731.5	4.24	103	0.02102	0.00871

# class, # instances and # features represent the number of classes, number of instances, and number of features, respectively. avg class size represents the average number of instances belonging to a class. avg classes/instance represents the average class labels each instance has. avg self class sim refers to average similarity of a class to itself, and avg class sim refers to the average similarity between classes. The average self- and cross-class similarity is calculated as the average pairwise similarity of the corresponding instances accordingly.

racy) and found that AUC performs better.

**5.3 Model Training** The various SVM models were built using the publicly available support vector machine tool *SVM<sup>light</sup>* [18] which implements an efficient soft margin optimization algorithm. In all of our experiments, we used the default parameters for solving the quadratic programming problem, the default regularization parameter  $C$  which controls the margin width and the misclassification cost, and in the case of the rbf kernel, we used the default  $\gamma$  value.

For the tf dataset we used the kernel function derived from the Tanimoto coefficient, whereas for all the remaining datasets we used the rbf kernel function. This was done as prior research in building models for chemical compound classification has shown that the Tanimoto coefficient performs better than the rbf kernel [30].

Since we build one-vs-rest binary SVM models, the size of the negative class in our models is significantly greater than that of the positive class, which can affect SVM’s performance in building a model for the positive class. Two ways are commonly used to address this problem, which are either to increase the misclassification cost associated with the positive instances or to only use a random subset of the negative instances. In all of our experiments we used the latter approach and created a negative class whose size is approximately equal to that of the positive class. Also, the advantage of this scheme is that it has lower computational requirements due to the smaller negative training class, which is important for the subset learning methods. In the case of the subset learning method, the sampled instance-subsets were generated as follows. First, a subset  $A$  of the negative instances was randomly selected whose size was equal to the size of the positive class. These negative instances were selected in such a way so that each class representing the “other” class in the binary setting contributed an equal number of instances. Then the subset  $A$  was used to generate the instance-subsets using the various methods

described in Section 4.2.1.

In the case of the cross-classification method, the model for the unlabeled set of class  $c_i$  was built by using its instances as the positive examples and the instances of the remaining test set (i.e., the held out fold minus the instances of  $c_i$ ) as the negative examples. Note this protocol satisfies the requirements outlined in Section 4.3.1 that models  $M_{i',i}$  and  $M_i$  do not share any positive or negative instances. Finally, in the case of the co-learning method, the two folds that form the training set was split into two parts. The first part, containing 2/3 of the training set, was used to build the  $M_{i+i'}$  model and the remaining 1/3 was used as the validation set to assess its performance.

## 6 Results

In this section we present and discuss the performance achieved by the various set classification methods described in Section 4. All the results reported correspond to the UPrec values averaged over the 3-fold cross validation steps.

**6.1 Consensus Prediction** Table 2 shows the results obtained by the consensus prediction method (Section 4.1). These results provide some indications as to the inherent difficulty of each dataset, which roughly correlates to the number of training instances per class. The results obtained by this method will provide the baseline performance against which the results obtained by the other methods will be compared.

**6.2 Subset Learning Methods** Tables 3 and 4 shows the results obtained by the methods that learn models based on instance-subsets (Section 4.2). The results reported are for different instance-subset generation schemes, kernel functions, number of selected instance-subsets, and instance-subsets of size 2 and 3 (i.e., pairs and triplets). The first of these tables (Table 3) shows the average performance of the subset methods over consensus prediction across the seven

Table 2: Consensus prediction performance.

	face	tf	scene	tmc	media	rcv1	yeast
UPrec	0.55	0.31	0.93	0.87	0.56	0.79	0.49

The results for the tf dataset were obtained using the kernel derived from the Tanimoto coefficient. The results for the remaining datasets were obtained using the rbf kernel.

Table 3: Average performance of subset learning methods relative to consensus prediction over the seven datasets.

kernel	method	s	m = 5	m = 10	m = 15	m = 20
$\mathcal{K}_{AND,\mathcal{K}}$	random	2	0.90	0.91	0.96	0.96
		3	0.83	0.83	0.87	0.88
	nn	2	<b>1.01</b>	1.00	0.98	0.99
		3	0.90	0.90	0.94	0.94
	fn	2	0.78	0.87	0.91	0.92
		3	0.69	0.79	0.85	0.88
$\mathcal{K}_{OR,\mathcal{K}}$	random	2	1.00	1.00	1.00	1.00
		3	0.98	1.00	0.98	0.98
	nn	2	<b>1.03</b>	<b>1.03</b>	1.00	<b>1.01</b>
		3	0.98	0.97	0.97	0.97
	fn	2	0.94	1.00	<b>1.01</b>	<b>1.01</b>
		3	0.96	1.00	0.99	0.99
$\mathcal{K}_{avg,\mathcal{K}}$	random	2	<b>1.06</b>	<b>1.07</b>	<b>1.05</b>	<b>1.05</b>
		3	<b>1.04</b>	<b>1.04</b>	<b>1.02</b>	<b>1.02</b>
	nn	2	<b>1.03</b>	<b>1.05</b>	<b>1.04</b>	<b>1.04</b>
		3	<b>1.01</b>	<b>1.02</b>	<b>1.01</b>	<b>1.02</b>
	fn	2	<b>1.03</b>	<b>1.05</b>	<b>1.08</b>	<b>1.09</b>
		3	0.99	<b>1.02</b>	<b>1.05</b>	<b>1.04</b>

These numbers are calculated as  $2^r$ , where  $r$  is the average  $\log_2$ -ratio of the uninterpolated precision of subset learning over consensus prediction across the different datasets. Numbers that are greater than one (**boldfaced**) indicate that the subset learning outperforms consensus prediction.

datasets, whereas the second table (Table 4) shows the actual results for the different datasets. We also performed a series of experiment on instance-subsets of size 5 that showed similar trends. Due to space constraints we did not include these results. In the rest of this section we discuss how the different parameters of this method impacts its overall performance.

**Size and Number of Instance-Subsets** Comparing the performance achieved by the subset learning methods across the different datasets, kernel functions, and instance-subset generation methods we see that as the size of the instance-subset increases from two to three, its performance in all but a few cases, decreases. Even though in most cases the drop in performance is relatively small, it is quite consistent indicating that the use of larger instance-subsets does not aid the set classification performance. However, an interesting trend regarding the relative performance of the instance-subsets of

size three over that of size two is that it improves (i.e., it degrades less) as the number of instance-subsets per training and testing instance ( $m$ ) increases. Based on these trends, a potential explanation for the lower performance of size three instance-subsets is that they require substantially larger training sets in order to build models that can generalize to unseen examples.

Comparing the performance achieved by the various schemes when the number ( $m$ ) of instance-subsets that are generated for each instance and used to train the various binary models and classify the unlabeled set increases, we see that the performance is getting better. We believe that this is a direct consequence of the fact that the size of the training set increases making it possible to improve the accuracy of the models. Also note that for instance-subsets of size two, the relative gains achieved with greater values of  $m$  diminishes, indicating that beyond a certain point, the used of additional subsets leads to relatively small incremental improvements. From a computational requirements standpoint, this is positive as it indicates that subset learning can lead to improvements with relatively modest increases in computational requirements. However, the rate of diminishing returns is considerably lower for instance-subsets of size three, and  $m = 10$  achieves the best results in this case (as discussed earlier).

**Kernel Functions** Comparing the performance of the three instance-subset based kernel functions (Section 4.2.2) we see that  $\mathcal{K}_{avg,\mathcal{K}}$  achieves the best results followed, relatively close, by  $\mathcal{K}_{OR,\mathcal{K}}$ , whereas  $\mathcal{K}_{AND,\mathcal{K}}$  tends to perform considerably worse. The sometimes comparable performance of the  $\mathcal{K}_{avg,\mathcal{K}}$  and  $\mathcal{K}_{OR,\mathcal{K}}$  is not surprising as these two kernels take into account all the information between a pair of instance-subsets, even though they do so somewhat differently. On the other hand, the  $\mathcal{K}_{AND,\mathcal{K}}$  kernel is distinctly different from the other two as it only takes into account information that is shared by all the instances of each instance-subset. Even though the principle behind this kernel is reasonable, we believe that its lower performance is due to the fact that it only considers features that are conserved 100% in the constituent instances of an instance-subset. As a result, it eliminates features that may be conserved within the instance-subset but not at the 100% level.

**Instance-Subset Generation Method** Comparing the performance of the three methods for generating instance-subsets (Section 4.2) we see that their relative performance depends on the kernel function that is being used. When the best two kernel functions are used ( $\mathcal{K}_{avg,\mathcal{K}}$  and  $\mathcal{K}_{OR,\mathcal{K}}$ ), the furthest-neighbor scheme usually performs the best, whereas the nearest-neighbor scheme tends to perform the worst. However, the performance of these methods does depend on the number of instance subsets ( $m$ ) that are used. In general, the performance of the furthest-neighbor scheme im-

Table 4: Performance of subset learning methods.

kernel	methods	s	face dataset				tf dataset				scene dataset			
			m = 5	m = 10	m = 15	m = 20	m = 5	m = 10	m = 15	m = 20	m = 5	m = 10	m = 15	m = 20
$\mathcal{K}_{AND, \mathcal{K}}$	random	2	0.50	0.50	<b>0.56</b>	<b>0.56</b>	0.20	0.20	0.27	0.27	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>
		3	0.37	0.37	0.48	0.49	0.20	0.20	0.23	0.23	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>
	nn	2	<b>0.57</b>	<b>0.57</b>	<b>0.56</b>	<b>0.56</b>	0.31	0.31	0.31	0.31	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>
		3	0.37	0.37	0.49	0.50	0.29	0.29	0.30	0.31	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>
	fn	2	0.40	0.44	0.53	0.54	0.20	0.23	0.22	0.22	<b>0.96</b>	0.91	<b>1.00</b>	<b>1.00</b>
		3	0.40	0.40	0.49	0.50	0.18	0.20	0.20	0.21	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>
$\mathcal{K}_{OR, \mathcal{K}}$	random	2	<b>0.62</b>	<b>0.62</b>	0.55	0.56	0.27	0.27	0.29	0.29	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>
		3	0.51	0.53	0.48	0.48	0.27	0.30	0.30	0.30	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>
	nn	2	<b>0.59</b>	<b>0.59</b>	<b>0.56</b>	<b>0.56</b>	<b>0.32</b>	<b>0.32</b>	<b>0.32</b>	<b>0.32</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>
		3	0.51	0.51	0.49	0.50	<b>0.32</b>	0.31	0.31	0.31	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>
	fn	2	<b>0.56</b>	<b>0.60</b>	<b>0.56</b>	<b>0.56</b>	0.30	<b>0.32</b>	0.31	0.32	<b>0.96</b>	0.91	<b>1.00</b>	<b>1.00</b>
		3	<b>0.56</b>	0.53	0.49	0.49	0.30	0.31	0.31	0.31	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>
$\mathcal{K}_{avg, \mathcal{K}}$	random	2	<b>0.64</b>	<b>0.64</b>	<b>0.57</b>	<b>0.57</b>	0.30	0.30	0.31	0.31	<b>0.96</b>	<b>0.96</b>	0.93	0.93
		3	0.54	0.54	0.49	0.49	0.29	0.29	0.30	0.30	<b>0.96</b>	<b>0.96</b>	<b>0.94</b>	<b>0.94</b>
	nn	2	<b>0.57</b>	<b>0.63</b>	<b>0.56</b>	<b>0.56</b>	0.31	0.31	0.31	0.31	<b>0.96</b>	<b>0.96</b>	0.93	0.93
		3	0.52	0.53	0.51	0.51	0.31	0.31	0.31	0.31	<b>0.96</b>	<b>0.96</b>	<b>0.94</b>	<b>0.94</b>
	fn	2	<b>0.59</b>	<b>0.58</b>	<b>0.58</b>	<b>0.58</b>	<b>0.32</b>	<b>0.32</b>	<b>0.32</b>	<b>0.32</b>	0.90	0.89	<b>0.98</b>	<b>0.98</b>
		3	0.51	0.53	0.52	0.51	<b>0.31</b>	<b>0.31</b>	<b>0.32</b>	<b>0.32</b>	0.85	0.88	<b>0.94</b>	0.88
kernel	methods	s	tmc dataset				media dataset				rcv1 dataset			
			m = 5	m = 10	m = 15	m = 20	m = 5	m = 10	m = 15	m = 20	m = 5	m = 10	m = 15	m = 20
$\mathcal{K}_{avg, \mathcal{K}}$	random	2	0.82	0.82	0.82	0.82	<b>0.59</b>	<b>0.60</b>	<b>0.60</b>	<b>0.60</b>	0.65	0.65	0.65	0.65
		3	0.64	0.64	0.64	0.64	<b>0.58</b>	<b>0.58</b>	<b>0.58</b>	<b>0.58</b>	0.58	0.58	0.58	0.58
	nn	2	0.83	0.83	0.84	0.86	<b>0.59</b>	<b>0.62</b>	<b>0.62</b>	<b>0.61</b>	0.75	0.73	0.73	0.72
		3	0.82	0.82	0.82	0.81	<b>0.59</b>	<b>0.61</b>	<b>0.61</b>	<b>0.60</b>	0.63	0.66	0.66	0.64
	fn	2	0.55	0.71	0.76	0.79	0.37	0.50	0.54	<b>0.57</b>	0.70	0.67	0.66	0.69
		3	0.28	0.45	0.55	0.65	0.32	0.45	0.50	0.54	0.63	0.65	0.66	0.65
$\mathcal{K}_{OR, \mathcal{K}}$	random	2	<b>0.88</b>	<b>0.88</b>	<b>0.88</b>	<b>0.88</b>	<b>0.59</b>	<b>0.60</b>	<b>0.60</b>	<b>0.60</b>	0.75	0.74	0.74	0.74
		3	0.84	0.84	0.84	0.84	<b>0.58</b>	<b>0.58</b>	<b>0.57</b>	<b>0.57</b>	0.78	0.78	0.78	0.78
	nn	2	<b>0.87</b>	<b>0.87</b>	0.87	0.88	<b>0.59</b>	<b>0.62</b>	<b>0.61</b>	<b>0.61</b>	0.78	0.78	0.78	0.79
		3	0.87	0.87	0.87	0.87	<b>0.59</b>	<b>0.61</b>	<b>0.61</b>	<b>0.60</b>	0.74	0.74	0.77	0.76
	fn	2	<b>0.89</b>	<b>0.93</b>	<b>0.92</b>	<b>0.92</b>	0.37	0.50	0.54	<b>0.57</b>	0.77	0.75	0.75	0.74
		3	<b>0.93</b>	<b>0.96</b>	<b>0.95</b>	<b>0.92</b>	0.32	0.45	0.50	0.54	0.78	0.78	0.75	0.73
$\mathcal{K}_{avg, \mathcal{K}}$	random	2	0.86	0.87	0.86	0.86	<b>0.66</b>	<b>0.68</b>	<b>0.68</b>	<b>0.68</b>	0.74	0.74	0.76	0.76
		3	0.85	0.85	0.85	0.85	<b>0.64</b>	<b>0.64</b>	<b>0.64</b>	<b>0.64</b>	0.76	0.76	0.76	0.76
	nn	2	0.87	0.87	0.86	0.86	<b>0.63</b>	<b>0.67</b>	<b>0.67</b>	<b>0.67</b>	<b>0.78</b>	<b>0.79</b>	<b>0.79</b>	<b>0.79</b>
		3	0.87	0.87	0.86	0.86	<b>0.59</b>	<b>0.66</b>	<b>0.67</b>	<b>0.67</b>	0.77	0.77	0.78	0.77
	fn	2	<b>0.90</b>	<b>0.90</b>	<b>0.89</b>	<b>0.88</b>	0.56	<b>0.60</b>	<b>0.63</b>	<b>0.65</b>	0.76	0.73	0.74	0.75
		3	<b>0.90</b>	<b>0.91</b>	<b>0.92</b>	<b>0.91</b>	0.47	0.53	<b>0.58</b>	<b>0.61</b>	<b>0.80</b>	0.76	0.74	0.73
kernel	methods	s	yeast dataset											
			m = 5	m = 10	m = 15	m = 20								
$\mathcal{K}_{AND, rbf}$	random	2	0.47	0.47	0.47	0.47								
		3	<b>0.49</b>	<b>0.49</b>	<b>0.49</b>	<b>0.49</b>								
	nn	2	0.48	0.46	0.40	0.42								
		3	0.43	0.40	0.40	0.40								
	fn	2	0.48	<b>0.50</b>	<b>0.50</b>	0.46								
		3	<b>0.56</b>	<b>0.54</b>	<b>0.52</b>	<b>0.50</b>								
$\mathcal{K}_{OR, rbf}$	random	2	0.47	0.47	0.47	0.47								
		3	<b>0.49</b>	<b>0.49</b>	<b>0.49</b>	<b>0.49</b>								
	nn	2	0.48	0.46	0.40	0.42								
		3	0.43	0.40	0.40	0.40								
	fn	2	0.48	<b>0.50</b>	<b>0.50</b>	0.46								
		3	<b>0.56</b>	<b>0.54</b>	<b>0.52</b>	<b>0.50</b>								
$\mathcal{K}_{avg, rbf}$	random	2	<b>0.59</b>	<b>0.58</b>	<b>0.59</b>	<b>0.59</b>								
		3	<b>0.62</b>	<b>0.62</b>	<b>0.62</b>	<b>0.62</b>								
	nn	2	<b>0.51</b>	<b>0.50</b>	<b>0.52</b>	<b>0.54</b>								
		3	<b>0.51</b>	<b>0.49</b>	<b>0.49</b>	<b>0.52</b>								
	fn	2	<b>0.56</b>	<b>0.66</b>	<b>0.68</b>	<b>0.68</b>								
		3	<b>0.63</b>	<b>0.66</b>	<b>0.66</b>	<b>0.68</b>								

The results show the UPrec values. The column labeled “methods” represents the method used to generate the instance-subsets (Section 4.2.1): random, nearest-neighbor (nn), and furthest-neighbor (fn);  $s$  and  $m$  are the size of each instance-subset and the number of instance-subsets that are derived for each instance (Section 4.2.1); **Boldfaced** entries denote results that are better than those obtained by the consensus prediction method (baseline).

proves for larger values of  $m$ , whereas the performance of the nearest-neighbor scheme tends to remain the same (or decrease) as  $m$  increases. One plausible explanation for the improved gains of the furthest-neighbor scheme with increasing values of  $m$  is that because the instances that are included in these subsets are dissimilar to the instance itself, the degree of variation among the selected instance-subsets is higher than those selected by the nearest-neighbor scheme and as such, there are gains to be made by including them in training.

However, the relative performance between the furthest- and nearest-neighbor scheme is reversed when the  $\mathcal{K}_{\text{AND},\mathcal{K}}$  kernel function is used. In this case, the nearest-neighbor scheme does substantially better than the furthest-neighbor scheme. The reason for that is that because the  $\mathcal{K}_{\text{AND},\mathcal{K}}$  represents the instance-subset by using only the features that are conserved across the instances of the subset, when these instances are very different from each other (as it is in the case of the furthest-neighbor scheme), it ends up eliminating a large number of features and thus leading to a poor representation of each instance-subset.

Finally, since the random method employs a uniform sampling approach, it achieves reasonably good and consistent performance across the different datasets, kernel functions, and number of selected instance-subsets. For most problems, its performance is usually close to the best, if not the best.

**Comparison with Consensus Prediction** The results in Table 3 show that for most instance-subset generation methods and values of  $m$ , the subset learning methods outperform consensus prediction when using the  $\mathcal{K}_{\text{OR},\mathcal{K}}$  and  $\mathcal{K}_{\text{avg},\mathcal{K}}$  kernel functions. In the case of  $\mathcal{K}_{\text{OR},\mathcal{K}}$ , the best overall results are obtained for the nearest-neighbor scheme and  $m = 10$ , which outperforms the consensus prediction in five out of the seven datasets and achieves an average improvement of 3%, with individual improvements on the five datasets ranging from 1.1% to 10.7%. In the case of  $\mathcal{K}_{\text{avg},\mathcal{K}}$ , the best overall results are obtained for the furthest-neighbor scheme and  $m = 20$ , which outperforms the consensus prediction on six out of the seven datasets and achieves an average improvement of 9%, with individual improvements on the five datasets ranging from 3.2% to 38.8%.

**6.3 Model-Based and Co-learning Methods** The performance of the set classification methods based on model comparison (Section 4.3) and co-learning (Section 4.4) is shown in the first four columns of Table 5. The last three columns of that table show the performance of the consensus prediction methods and the two best schemes based on subset learning. These results are reproduced here from Tables 2 and 4 so that to make it easier to compare the performance of all schemes together.

Comparing the performance of these schemes we see

that the model-based approach that directly determines the similarity between a pair of models performs poorly and achieves results that are considerably worse than those achieved by the baseline scheme (consensus predictor). However, the cross-classification based scheme does better and outperforms the baseline scheme in four out of the seven datasets. In particular, its performance on the tf dataset is 75% higher than that of the baseline scheme. Comparing the performance of the co-learning method against the baseline scheme we see that it is not very good and underperforms the baseline scheme in all but one of the datasets.

Finally, comparing the overall performance of the various schemes in Table 5 we see that the subset-learning methods achieve the best overall results and that they lead to considerable improvements over the consensus prediction method.

**6.4 Additional Experiments** In addition to constructing subsets by looking at the instance similarities, which can be thought of as exploring the structure of the instances' input space, we also experimented with schemes that generated subsets by considering the class labels of the instances. In particular, when the instances were multi-labeled, we constructed subsets using the instances which have the most similar or dissimilar labels. This can be thought of as exploring the structure of the instances' output space. Our experiments with this approach (not reported here) showed that it did not lead to any significant improvements over consensus predictions. This may be because the labels are only a high-level description of the instances, and thus subsets from similar or dissimilar labels do not necessarily provide additional information to the learner.

We also applied the C4.5 decision tree algorithm on instance-subsets in order to evaluate that the subset learning methods are not limited only within the SVM framework. We generated a random forest of pruned trees on the features of instance-subsets for each class of each dataset. The classification of an instance-subset is the majority prediction of the forest. Our experiments showed (not reported here) that as measured in terms of recall, learning on subsets improved the classification performance over consensus prediction as well.

## 7 Conclusion and Directions for Further Research

In this paper we introduced the set classification problem and developed various approaches for solving it. These methods included the consensus predictor, which is a natural application of traditional classification methods, methods that build models on examples corresponding to various subsets of the instances, methods that build models for the testing set and compare it against those built on the training set, and finally methods that measure the improvement in the classification performance by using the test set to co-learn a model with

Table 5: Performance of all the schemes.

dataset	Model-based		co-learning	consensus prediction	$\mathcal{K}_{\text{OR},\mathcal{K}}$ nn, $s = 2, m = 10$	$\mathcal{K}_{\text{avg},\mathcal{K}}$ fn, $s = 2, m = 20$
	dsim	xsim				
face	0.42	0.45	0.39	0.55	<b>0.59</b>	<b>0.58</b>
tf	0.20	<b>0.54</b>	<b>0.44</b>	0.31	<b>0.32</b>	<b>0.32</b>
scene	0.68	<b>1.00</b>	0.78	0.93	<b>1.00</b>	<b>0.98</b>
tmc	0.35	<b>0.91</b>	0.73	0.87	<b>0.87</b>	<b>0.92</b>
media	0.11	0.27	0.23	0.56	<b>0.62</b>	<b>0.65</b>
rcv1	0.37	0.74	0.62	0.79	0.78	0.75
yeast	0.46	<b>0.50</b>	0.43	0.49	0.46	<b>0.68</b>

The ranking of the different classes for xsim and co-learning during prediction was done using AUC. **Boldfaced** entries denote results that are better than those obtained by the consensus prediction method.

the training set. We performed a detailed parameter study of these methods using four datasets. Our results showed that among these methods, significant performance gains can be obtained by the subset learning methods across the majority of the datasets.

Even though the overall results of this study have been positive, we believe that there are a number of issues that need to be further investigated for each one of the four classes of methods presented in this paper. Additional research is required for the consensus prediction approach as it relates to the methods used to derive the consensus prediction that does not utilize every individual instance but selects reliable subsets upon which to build the consensus. In the case of the subset learning methods additional research is required to determine the relation between the size of the instance-subset and the number of instance-subsets used for training. In addition, further research is required to improve the performance of the  $\mathcal{K}_{\text{AND},\mathcal{K}}$  kernel by relaxing the requirement of 100% conservation. In the case of model-based methods, being able to determine the similarity between two models by only comparing their support vectors is theoretically appealing and we believe that more effective methods can potentially be developed. Finally, even though the methods based on co-learning did not perform competitively, there may be room for further improvements by developing methods to differentially weight the positive instances that came from the original training set over those that came from the test set.

**Acknowledgements** This work was supported by IIS-0431135, NIH RLM008713A, and by the Digital Technology Center at the University of Minnesota.

## References

- [1] S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *NIPS*, pages 561–568, Vancouver, 2002. MIT press.
- [2] O. Arandjelović. and R. Cipolla. Face set classification using maximally probable mutual modes. In *Proc. IAPR International Conference on Pattern Recognition*, pages 511–514, August 2006.
- [3] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 92–100. Morgan Kaufmann Publishers, 1998.
- [4] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37:1757–1771, 2004.
- [5] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 307–318, New York, NY, USA, 1998. ACM.
- [6] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.
- [7] M. Deshpande, M. Kuramochi, N. Wale, and G. Karypis. Frequent substructure-based approaches for classifying chemical compounds. *IEEE Trans. on Knowl. and Data Eng.*, 17(8):1036–1050, 2005.
- [8] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artif. Intell.*, 89(1-2):31–71, 1997.
- [9] A. E. Elisseeff and J. Weston. A kernel method for multi-labelled classification. In *In Advances in Neural Information Processing Systems 14*, pages 681–687. MIT Press, 2001.
- [10] J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.
- [11] T. Fawcett. An introduction to roc analysis. *Pattern Recogn. Lett.*, 27(8):861–874, 2006.
- [12] T. Gärtner, P. A. Flach, A. Kowalczyk, and A. J. Smola. Multi-instance kernels. In *ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning*, pages 179–186, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.
- [13] D. B. Graham and N. M. Allinson. Characterizing virtual eigensignatures for general purpose face recognition. *Face Recognition: From Theory to Applications, NATO ASI Series F, Computer and Systems Sciences*, 163:446–456, 1998.
- [14] M. A. Hearst and J. O. Pedersen. Reexamining the clus-

- ter hypothesis: scatter/gather on retrieval results. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 76–84, New York, NY, USA, 1996. ACM.
- [15] B. Heisele, P. Ho, and T. Poggio. Face recognition with support vector machines: global versus component-based approach. In *In Proc. 8th International Conference on Computer Vision*, pages 688–694, 2001.
- [16] J. L. Jenkins, A. Bender, and J. W. Davies. In silico target fishing: Predicting biological targets from chemical structure. *Drug Discovery Today: Technologies*, 3:413–421, 2006.
- [17] D. Jensen, J. Neville, and B. Gallagher. Why collective inference improves relational classification. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 593–598, New York, NY, USA, 2004. ACM.
- [18] T. Joachims. Making large-scale svm learning practical. *Advances in Kernel Methods - Support Vector Learning.*, 1999.
- [19] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5:361–397, 2004.
- [20] T. Liu, Y. Lin, X. Wen, R. N. Jorissen, and M. K. Gilson. Bindingdb: a web-accessible database of experimentally determined protein-ligand binding affinities. *Nucleic Acids Research 00(Database Issue)*, pages D1–D4, 2006.
- [21] Q. Lu and L. Getoor. Link-based classification. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 496–503, DC, USA, 2003. AAAI.
- [22] O. Maron and T. Lozano-Pérez. A framework for multiple-instance learning. In *NIPS '97: Proceedings of the 1997 conference on Advances in neural information processing systems 10*, pages 570–576, Cambridge, MA, USA, 1998. MIT Press.
- [23] A. Nichols. High content screening as a screening tool in drug discovery. *Methods in Molecular Biology*, 356:379–387, 2007.
- [24] R. J. Quinlan. *C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning)*. Morgan Kaufmann, January 1993.
- [25] B. L. Roth, E. Lopez, S. Patel, and W. K. Kroeze. The multiplicity of serotonin receptors: Uselessly diverse molecules or an embarrassment of riches? *The Neuroscientist*, 6:252–262, 2000.
- [26] G. Shakhnarovich, I. John W. Fisher, and T. Darrell. Face recognition from long-term observations. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part III*, pages 851–868, London, UK, 2002. Springer-Verlag.
- [27] C. G. M. Snoek, M. Worring, J. C. van Gemert, J.-M. Geusebroek, and A. W. M. Smeulders. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*, pages 421–430, New York, NY, USA, 2006. ACM.
- [28] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models fro relational data. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, pages 485–492, Edmonton, Canada, 2002. Morgan Kaufmann.
- [29] V. Vapnik. *Statistical Learning Theory*. John Wiley, New York, 1998.
- [30] N. Wale, I. A. Watson, and G. Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14(3):347–375, 2008.
- [31] J. Wang and J.-D. Zucker. Solving multiple-instance problem: A lazy learning approach. In *Proceedings of 17th International Conference on Machine Learning*, pages 1119–1125, 2000.
- [32] P. Willett. Chemical similarity searching. *Journal of Chemical Information and Computer Science*, 38(6):983–996, 1998.
- [33] D. S. Wishart, C. Knox, A. C. Guo, S. Shrivastava, M. Hassanali, P. Stothard, Z. Chang, and J. Woolsey. DrugBank: a comprehensive resource for in silico drug discovery and exploration. *Nucl. Acids Res.*, 34(suppl1):D668–672, 2006.
- [34] D. W. Young, A. Bender, J. Hoyt, E. McWhinnie, G.-W. Chirn, C. Y. Tao, J. A. Tallarico, M. Labow1, J. L. Jenkins, T. J. Mitchison, and Y. Feng. Integrating high-content screening and ligand-target prediction to identify mechanism of action. *Nature Chemical Biology*, 4:59–68, 2008.
- [35] Z.-H. Zhou and M.-L. Zhang. Multi-instance multi-label learning with application to scene classification. *Advances in Neural Information Processing Systems*, 19:1609–1616, 2007.
- [36] Z.-H. Zhou and M.-L. Zhang. Solving multi-instance problems with classifier ensemble based on constructive clustering. *Knowl. Inf. Syst.*, 11(2):155–170, 2007.