Instance Annotation for Multi-Instance Multi-Label Learning

FORREST BRIGGS, XIAOLI Z. FERN, RAVIV RAICH, and QI LOU, Oregon State University

Multi-instance multi-label learning (MIML) is a framework for supervised classification where the objects to be classified are bags of instances associated with multiple labels. For example, an image can be represented as a bag of segments and associated with a list of objects it contains. Prior work on MIML has focused on predicting label sets for previously unseen bags. We instead consider the problem of predicting instance labels while learning from data labeled only at the bag level. We propose a regularized rank-loss objective designed for instance annotation, which can be instantiated with different aggregation models connecting instance-level labels with bag-level label sets. The aggregation models that we consider can be factored as a linear function of a "support instance" for each class, which is a single feature vector representing a whole bag. Hence we name our proposed methods rank-loss Support Instance Machines (SIM). We propose two optimization methods for the rank-loss objective, which is nonconvex. One is a heuristic method that alternates between updating support instances, and solving a convex problem in which the support instances are treated as constant. The other is to apply the constrained concave-convex procedure (CCCP), which can also be interpreted as iteratively updating support instances and solving a convex problem. To solve the convex problem, we employ the Pegasos framework of primal subgradient descent, and prove that it finds an ϵ-suboptimal solution in runtime that is linear in the number of bags, instances, and $\frac{1}{\epsilon}$. Additionally, we suggest a method of extending the linear learning algorithm to nonlinear classification, without increasing the runtime asymptotically. Experiments on artificial and real-world datasets including images and audio show that the proposed methods achieve higher accuracy than other loss functions used in prior work, e.g., Hamming loss, and recent work in ambiguous label classification.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; I.5.2 [**Pattern Recognition**]: Design Methodology—*Classifier design and evaluation*

General Terms: Algorithms, Performance, Experimentation

Additional Key Words and Phrases: Instance annotation, image annotation, multi-instance, multi-label, support vector machine, subgradient, bioacoustics

ACM Reference Format:
Briggs, F., Fern, X. Z., Raich, R., and Lou, Q. 2013. Instance annotation for multi-instance multi-label learning. *ACM Trans. Knowl. Discov. Data* 7, 3, Article 14 (September 2013), 30 pages.
DOI: http://dx.doi.org/10.1145/2500491

1. INTRODUCTION

Many problems in supervised classification have a certain structure, where the objects of interest (e.g., images or text documents) can naturally be decomposed into a collection of parts called a bag-of-instances representation. For example, in image classification, an image is typically a bag, and the pixels or segments in it are instances. This structure motivates multiple-instance learning (MIL) [Dietterich et al. 1997]. The

A preliminary version of this work appeared in Briggs et al. [2010].
This work is partially funded by the Ecosystems Informatics IGERT program via NSF grant DGE 0333257, NSF grant 1055113 to Xiaoli Z. Fern, and the College of Engineering, Oregon State University.
Authors' address: F. Briggs, X. Z. Fern, R. Raich, and Q. Lou, Oregon State University, School of EECS, Corvallis, Oregon; email: fbriggs@gmail.com.

DOI: http://dx.doi.org/10.1145/2500491

original formulation of MIL concerns problems where bags are associated with a single binary label. Zhou and Zhang [2007] introduced multi-instance multi-label learning (MIML), where bags are instead associated with a set of labels. For example, an image might be associated with a list of the objects it contains.

MIML arises in situations where the cost of labeling individual instances becomes prohibitive and consequently multiple instances are grouped and associated with a set of labels. For example, it is much faster to label an image with a few words than to individually label pixels. In MIML, the training dataset consists of a collection of bags of instances, where each bag is associated with multiple labels. The standard goal in MIML is to learn a classifier that predicts the label set for a previously unseen bag. Numerous algorithms for MIML have been proposed and applied to image, text [Li et al. 2009; Shen et al. 2009; Zha et al. 2008; Zhou and Zhang 2007; Zhou et al. 2012], and video [Xu et al. 2011] domains. Recently, Wang and Zhou [2012] analyzed PAC-learnability in MIML.

Learning to predict instance labels from MIML training data is a useful problem that has received little study [Zhou 2004]. For example, one might train a classifier on a collection of images paired with lists of object names in each image, then make predictions about the label for each region in an image. This problem is called the *instance annotation* problem for MIML. The key issue in instance annotation is how to learn an instance-level classifier from a MIML dataset, which presents only bag-level labels.

A common strategy in designing MIML algorithms for bag-level prediction is to learn an instance-level model by minimizing a loss function defined at the bag level. For example, several previous bag-level MIML algorithms minimize bag-level Hamming loss, which captures the disagreement between a ground-truth label set, and the predicted label set. Practically speaking, these instance-level models could be directly used for instance-level prediction. However, the loss functions (e.g., Hamming Loss) used by such bag-level MIML algorithms are designed to optimize the prediction performance at the bag level and can be inappropriate for the purpose of making instance-level predictions.

For instance annotation, typically the instance-level classifier outputs a score for each class, and the instance label is predicted as the highest scoring class. Therefore the predicted label depends on the ranking of class scores. Hamming loss is not appropriate in this context, despite its success at bag-level predictions, because it lacks a mechanism to calibrate the scores between different classes. This observation motivates us to introduce a rank-loss objective for instance annotation, which directly optimizes the ranking of classes.

In order to learn an instance-level classifier using a bag-level loss function, it is necessary to define an aggregation model that connects instance-level predictions with bag-level predictions. In this article, we examine two different aggregation models, which is equivalent to defining a "support instance" for each bag. Therefore, we name our methods Support Instance Machines (SIM).

In this article, we make the following contributions.

— We propose a regularized rank-loss objective for instance annotation that can be instantiated with different aggregation models (Section 4.1).
— The rank-loss objective is nonconvex. To address this challenge, we offer two optimization methods that are effective in practice. One is a heuristic that linearizes the aggregation model, and the other casts the objective as a difference of convex functions and monotonically decreases the objective using the constrained concave convex procedure (CCCP) [Yuille and Rangarajan 2002].

— In either optimization method, the core of the algorithm is to alternate between updating support instances, and solving a convex problem using the Pegasos framework for primal subgradient descent. We prove that the convex optimization has linear runtime in the number of bags, instances, and $\frac{1}{\epsilon}$ to find an ϵ-suboptimal solution (Section 4.3.4).
— Experiments show that SIM with rank loss achieves higher accuracy than Hamming loss or ambiguous loss (a comparable state of the art approach [Cour et al. 2009, 2011]). A novel `softmax` aggregation model generally achieves higher accuracy than the `max` model, which has been used in prior multi-instance or MIML algorithms, and CCCP improves accuracy over the heuristic with the same aggregation model (Section 5.2.4).
— We suggest a method of extending our proposed classifiers from linear to nonlinear using a fast approximate kernel trick [Rahimi and Recht 2007]. Experiments show that this method often improves accuracy significantly, while still achieving linear runtime in the number of bags and instances (Section 4.4).
— We introduce a real-world MIML dataset for instance annotation derived from over 90 minutes of bird song recordings collected in the field, containing multiple simultaneously vocalizing birds of different species. Several SIM algorithms achieve over 80% accuracy in predicting the species of bird responsible for each sound in the recordings given the list of species present in the recording (Section 5.1.1).

2. PROBLEM STATEMENT

In this section we formalize the instance annotation problem and contrast it with several related problems. Table I summarizes notation.

We are given a training set of n bags $(X_1, Y_1), \ldots, (X_n, Y_n)$. Each X_i is a bag of n_i instances, i.e., $X_i = \{\mathbf{x}_{i1}, \cdots, \mathbf{x}_{in_i}\}$, with $\mathbf{x}_{iq} \in \mathcal{X}$, where $\mathcal{X} = \mathbb{R}^d$ is a d-dimensional feature space. Each bag X_i is associated with a label set $Y_i \subseteq \mathcal{Y}$ where $\mathcal{Y} = \{1, \cdots, c\}$ and c is the total number of classes. We will assume that each instance $\mathbf{x}_{iq}$ has a hidden label $y_{iq} \in \mathcal{Y}$ and that the bag label set is equal to the union of the instance labels, i.e. $Y_i = \cup_{q=1,\ldots,n_i} y_{iq}$. The goal of instance annotation in MIML is to learn an instance-level classifier $f_{IA} : \mathcal{X} \to \mathcal{Y}$ that maps an element of the input space $\mathcal{X}$ to its corresponding class label.

We focus on classifiers for instance annotation that use one instance-level model $f_j(\mathbf{x}) = \mathbf{w}_j \cdot \mathbf{x}$ for each class j, and predict a specific label via $f(\mathbf{x}) = \arg\max_j f_j(\mathbf{x})$. The goal is to learn the weights $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_c]$ (note $\mathbf{x} \in \mathbb{R}^d, \mathbf{w}_j \in \mathbb{R}^d$, and $\mathbf{W} \in \mathbb{R}^{cd}$).

The instance annotation problem is different from the traditional MIML learning problem studied by Zhou and Zhang [2007] and many others, where the goal is to learn a bag-level classifier $F_{MIML} : 2^{\mathcal{X}} \to 2^{\mathcal{Y}}$. Nonetheless, the design principles of many traditional MIML algorithms can be used to learn instance-level classification models, which is the approach we take in this article.

Instance annotation is closely related to the classic supervised classification problem, where the goal is to learn an instance classifier, but using training data that does not have a bag structure and has each instance labeled individually. In contrast to MIML, this classic setting is often referred to as single-instance single-label (SISL) learning.

Ambiguous label classification (ALC) [Cour et al. 2011; Hüllermeier and Beringer 2006] is another related framework. In ALC, there are no bags; instead instances are paired with a set of possible labels, only one of which is correct. An ALC dataset is $(\mathbf{x}_1, Y_1), \ldots, (\mathbf{x}_m, Y_m)$, where $\mathbf{x}_i \in \mathcal{X}$ and $Y_i \subseteq \mathcal{Y}$. In ALC the goal is to learn a classifier that predicts instance labels, hence an ALC classifier is a function $f_{ALC} : \mathcal{X} \to \mathcal{Y}$. A MIML instance annotation problem can be transformed into an ALC problem by

Table I. Summary of Notation

Notation	Meaning
n	number of bags
n_i	number of instances in bag i
m	number of instances in all bags $= \sum n_i$
c	number of classes
$\mathbf{x}_{iq}$	instance q in bag i, a feature vector in $\mathbb{R}^d$
X_i	bag i, a set of instances
Y_i	label set for bag i, a subset of $\{1, \ldots, c\}$
$\bar{Y}_i$	complement of Y_i
Y_{ij}	+1 if $j \in Y_i$ and -1 otherwise
$f_j(\mathbf{x})$	instance-level model for class j
$F_j(X)$	bag-level model for class j
$\mathbf{w}_j$	instance-level weights for class j
$\mathbf{W}$	concatenation of $\mathbf{w}_1, \ldots, \mathbf{w}_c$ as a vector
$\mathbf{W}^{(t)}$	weights at the start of iteration t of CCCP or the heuristic
$\mathbf{W}^{(t,\tau)}$	weights at iteration τ of subgradient descent, iteration t of CCCP or the heuristic
$\hat{\mathbf{x}}_{ij}(\mathbf{W})$	support instance for bag i, class j as a function of $\mathbf{W}$
$\hat{\mathbf{x}}_{ij}^{(t)}$	support instance for bag i, class j at iteration t of CCCP or the heuristic
$\hat{\mathbf{x}}_{ij}^{(t,\tau)}$	support instance for bag i, class j at iteration τ of subgradient descent, iteration t of CCCP
β_i	$\frac{1}{n\lvert Y_i\rvert\lvert\bar{Y}_i\rvert}$
$\sum_{ijk}$	$\sum_{i=1}^{n}\sum_{j\in Y_i}\sum_{k\notin Y_i}$

creating one ALC instance for each instance in a MIML bag, paired with all of the labels from the bag. Hence ALC algorithms can be applied to MIML instance annotation problems. However, this reduction may discard useful bag-level structure in the MIML data. The bag-level structure in MIML implies that each label in a bag label set must be explained by at least one of the instances in that bag, whereas this constraint is lost in the reduction to ALC.

3. BACKGROUND

Here we discuss some design patterns that contribute to our proposed methods.

3.1. Connecting Instance Labels with Bag Labels

One common approach in MIML algorithms is to make bag-level predictions based on the outputs of instance-level models. The connection from instance labels to bag labels is made via the assumption that the bag label set is the union of instance labels. This assumption is used in several MIML algorithms including M^{3}MIML [Zhang and Zhou 2008], and D-MimlSvm [Zhou et al. 2012]. The following formulation approximates this assumption. Let $f_j(\mathbf{x}) : \mathcal{X} \to \mathbb{R}$ be a function that takes an instance and returns a real-valued score for class j. The output at the bag-level for class j is defined to be $F_j(X) = \max_{\mathbf{x}\in X} f_j(\mathbf{x})$. A bag-level classifier can be obtained by applying a threshold (e.g., 0) to the bag-level scores, i.e. $F(X) = \{y \in \mathcal{Y} : F_y(X) > 0\}$. Note that if an instance $\mathbf{x}^*$ within bag X is predicted to belong to class j, i.e., $f_j(\mathbf{x}^*) > 0$, the predicted label set for bag X will necessarily contain j because $F_j(X) = \max_{\mathbf{x}\in X} f_j(\mathbf{x}) \geq f_j(\mathbf{x}^*) > 0$. Hence, connecting instance and bag-level scores via the max function is related to defining the bag label set as the union of the instance labels.

3.2. Bag-Level Loss Functions

In contrast with SISL or instance annotation, which are evaluated based on instance-level accuracy, existing MIML algorithms are typically evaluated based on their label set predictions. Two common performance measures are Hamming loss, and rank loss [Zhou et al. 2012]. Hamming Loss is the number of false positives and false negatives, averaged over all classes and bags,

$$\frac{1}{nc}\sum_{i=1}^{n}\sum_{j=1}^{c} I[j \in F(X_i), j \notin Y_i] + I[j \notin F(X_i),\ j \in Y_i] \tag{1}$$

Rank loss captures the number of label pairs that are incorrectly ordered by the scores of the MIML classifier. For a given bag, classes in its true label set should receive higher scores than classes that are not. Let $\bar{Y}$ denote the complement of Y. Rank loss is defined as

$$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{|Y_i||\bar{Y}_i|}\sum_{j\in Y_i, k\in\bar{Y}_i} I[F_j(X_i) \leq F_k(X_i)] \tag{2}$$

These objectives are difficult to optimize directly because they are not continuous. Several prior algorithms for MIML can be viewed as optimizing a surrogate for Hamming loss. For example, D-MimlSvm [Zhou et al. 2012] and M^{3}MIML [Zhang and Zhou 2008] optimize variations of the following loss function (with different regularization terms).

$$\frac{1}{nc}\sum_{i=1}^{n}\sum_{j=1}^{c}\max\{0, 1 - Y_{ij}F_j(X_i)\}, \tag{3}$$

where $Y_{ij} = +1$ if $j \in Y_i$ and -1 if $j \notin Y_i$.

The hinged Hamming-loss objective (3) can be decomposed into a collection of independent MIL problems, one for each class. As such, it does not calibrate the scores between classes, which could make predicting an instance label based on the highest scoring class unreliable. To overcome this limitation, we consider rank loss instead. Rank loss has been used as an objective for single-instance multi-label SVMs [Elisseeff and Weston 2001]. However, we are not aware of any MIML algorithms that learn an instance-level model by minimizing rank loss (rank loss has only been used as a performance measure in MIML, not as an objective).

4. PROPOSED METHODS

Our proposed methods are based on the observation that the predicted instance label depends on the ranking of scores for each class. Hence, we propose a rank-loss objective that optimizes this ranking. The rank-loss objective can be instantiated with different aggregation models that connect instance labels with bag label sets. We consider aggregation models that can be factored as a linear function of *support instances*, which are feature vectors that summarize a bag. We propose two optimization methods for the rank-loss objective, which is nonconvex. One is a heuristic, and the other is based on CCCP, however both exploit the support instance structure in the optimization problem.

4.1. Rank-Loss Objective

Because we are learning from an MIML dataset, we can only measure loss at the bag level. A bag-level loss function measures the agreement between a bag label set and

the bag-level scores $F_j(X_i)$. We propose the following a bag-level loss function, which is a regularized surrogate for rank loss (2).

$$h_{RL}(\mathbf{W}) = \frac{\lambda}{2}||\mathbf{W}||^2 + \sum_{i=1}^{n} \sum_{j \in Y_i} \sum_{k \notin Y_i} \frac{1}{n|Y_i||\bar{Y}_i|} \max\{0, 1 - \big(F_j(X_i) - F_k(X_i)\big)\}. \tag{4}$$

To shorten our notation, let $\beta_i = \dfrac{1}{n|Y_i||\bar{Y}_i|}$ and $\sum_{ijk} \equiv \sum_{i=1}^{n} \sum_{j \in Y_i} \sum_{k \notin Y_i}$. Then we can rewrite the objective as

$$h_{RL}(\mathbf{W}) = \frac{\lambda}{2}||\mathbf{W}||^2 + \sum_{ijk} \beta_i \max\{0, 1 - \big(F_j(X_i) - F_k(X_i)\big)\}. \tag{5}$$

This objective is designed to encourage a correct ranking of the bag-level scores for each class. For a bag X_i with corresponding label set Y_i, if $j \in Y_i$ and $k \in \bar{Y}_i$, then the loss is zero only if $F_j(X_i) > F_k(X_i) + 1$ (requiring a difference of at least 1 promotes a large-margin solution). The objective is also designed to facilitate primal subgradient descent and CCCP optimization methods, which we discuss in Section 4.3.

4.2. Aggregation Models and Support Instances

Objective (4) can be instantiated with various aggregation models $F_j(\cdot)$ that compute bag-level scores from instance-level scores. For example, the `max` model, which has been used in prior work [Zhang and Zhou 2008; Zhou et al. 2012], with a linear instance classifier is

$$F_j(X_i) = \max_{\mathbf{x}_{iq} \in X_i} f_j(\mathbf{x}_{iq}) = \max_{\mathbf{x}_{iq} \in X_i} \mathbf{w}_j \cdot \mathbf{x}_{iq}. \tag{6}$$

It is equivalent to write $F_j(X_i) = \mathbf{w}_j \cdot \hat{\mathbf{x}}_{ij}(\mathbf{W})$, where

$$\hat{\mathbf{x}}_{ij}(\mathbf{W}) = \arg\max_{\mathbf{x}_{iq} \in X_i} \mathbf{w}_j \cdot \mathbf{x}_{iq}. \tag{7}$$

We refer to $\hat{\mathbf{x}}_{ij}(\mathbf{W})$ as the support instance for bag X_i, class j, because the bag-level output for X_i depends only on the support instance for each class (analogous to a support vector).

The `max` model represents each bag with the most characteristic instance of each class. This approach can ignore other instances that are also useful for learning, and may not be appropriate when the assumption that the bag label set is equal to the union of instance labels does not hold. We propose an alternative `softmax` model, which can also be expressed in terms of support instances, but has the advantage of basing the support instances on more than one instance per class for each bag. The `softmax` model represents each bag as a weighted average of the instances, with weights specific to each class:

$$F_j(X_i) = \sum_{\mathbf{x}_{iq} \in X_i} \alpha_{iq}^j f_j(\mathbf{x}_{iq}) = \sum_{\mathbf{x}_{iq} \in X_i} \alpha_{iq}^j \mathbf{w}_j \cdot \mathbf{x}_{iq}. \tag{8}$$

The weights are defined according to a softmax rule,

$$\alpha_{iq}^j = \frac{e^{\mathbf{w}_j \cdot \mathbf{x}_{iq}}}{\sum_{\mathbf{x}' \in X_i} e^{\mathbf{w}_j \cdot \mathbf{x}'}}. \tag{9}$$

We can also write the softmax model as $F_j(X_i) = \mathbf{w}_j \cdot \hat{\mathbf{x}}_{ij}(\mathbf{W})$, with the support instances defined as

$$\hat{\mathbf{x}}_{ij}(\mathbf{W}) = \sum_{\mathbf{x}_{iq} \in X_i} \alpha_{iq}^j \mathbf{x}_{iq}. \tag{10}$$

For either model, the rank-loss objective in terms of support instances is

$$\hat{h}_{RL}(\mathbf{W}) = \frac{\lambda}{2}||\mathbf{W}||^2 + \sum_{ijk} \beta_i \max\{0, 1 + \mathbf{w}_k \cdot \hat{\mathbf{x}}_{ik}(\mathbf{W}) - \mathbf{w}_j \cdot \hat{\mathbf{x}}_{ij}(\mathbf{W})\}. \tag{11}$$

4.3. Optimization Methods for Rank-Loss Support Instance Machines

SIM could be instantiated with any aggregation model that fits the pattern of (11). However, depending on the aggregation model, different optimization techniques are required. Using the max and softmax models, $\hat{\mathbf{x}}_{ij}(\mathbf{W})$ is a function of $\mathbf{w}_j$, and the objective is nonconvex. We propose two optimization strategies to handle this nonconvex objective: (1) if $F_j(\cdot)$ is a convex function of $\mathbf{W}$, we can apply CCCP, or (2) as a heuristic, treat $\hat{\mathbf{x}}_{ij}$ as constant, which makes the objective convex.

4.3.1. Constrained Convex Concave Procedure (CCCP). The bag-level surrogate loss functions that arise in multi-instance learning are often nonconvex, but can sometimes be manipulated into a difference-of-convex (DC) form, which allows the use of CCCP [Sriperumbudur and Lanckriet 2009; Yuille and Rangarajan 2002]. The advantage of CCCP is that it decreases the objective monotonically and converges to a local optimum or a saddle point. CCCP can be applied to optimization problems of the form

$$\min_{\mathbf{x}} \quad f_0(\mathbf{x}) - g_0(\mathbf{x}) \tag{12}$$

$$s.t. \quad f_i(\mathbf{x}) - g_i(\mathbf{x}) \leq 0, i = 1, \ldots, k, \tag{13}$$

where all f_i and g_i are convex. CCCP solves a sequence of convex upper bounds to the original problem by constructing a linear upper bound of the concave part of the objective and constraints at the current point. Let $\mathbf{x}^{(t)}$ be the current point, then the next point $\mathbf{x}^{(t+1)}$ is obtained by solving the following convex problem.

$$\min_{\mathbf{x}} \quad f_0(\mathbf{x}) - \left[g_0(\mathbf{x}^{(t)}) + \nabla g_0(\mathbf{x}^{(t)}) \cdot (\mathbf{x} - \mathbf{x}^{(t)})\right] \tag{14}$$

$$s.t. \quad f_i(\mathbf{x}) - \left[g_i(\mathbf{x}^{(t)}) + \nabla g_i(\mathbf{x}^{(t)}) \cdot (\mathbf{x} - \mathbf{x}^{(t)})\right] \leq 0, \quad i = 1, \ldots, k. \tag{15}$$

If g_i is nondifferentiable, a subgradient $\mathbf{u} \in \partial g_i(\mathbf{x}^{(t)})$ can be used in place of the gradient $\nabla g_i(\mathbf{x}^{(t)})$. This is the case for our derivations using CCCP, as g_i involves the max function.

4.3.2. CCCP for Rank-Loss Support Instance Machines. When the aggregation model $F_j(\cdot)$ is a convex function of $\mathbf{W}$, CCCP can be applied to the rank-loss objective. The max model is convex, but the softmax model is not. The objective (11) is not a DC function, but we can still use CCCP with a simple transformation of the problem. The unconstrained problem (11) is equivalent to a constrained problem

$$\min_{\mathbf{W}, \xi} \quad \frac{\lambda}{2}||\mathbf{W}||^2 + \sum_{ijk} \beta_i \xi_{ijk} \tag{16}$$

$$s.t. \quad F_j(X_i) - F_k(X_i) \geq 1 - \xi_{ijk}, \text{ for } i = 1, \ldots, n, \; j \in Y_i, k \notin Y_i. \tag{17}$$

$$\xi_{ijk} \geq 0 \tag{18}$$

The constraint (17) is not convex, but when the aggregation model $F_j(\cdot)$ is convex, the constraint is DC. As an example of applying the CCCP framework to a convex aggergation model, we will use the `max` model. Substituting in the `max` model and rearranging, the constraint becomes

$$\underbrace{1 - \xi_{ijk} + \max_{\mathbf{x} \in X_i} \mathbf{w}_k \cdot \mathbf{x}}_{f_{ijk}} \underbrace{- \max_{\mathbf{x} \in X_i} \mathbf{w}_j \cdot \mathbf{x}}_{g_{ijk}} \leq 0. \tag{19}$$

We define the support instances at iteration t of CCCP as

$$\hat{\mathbf{x}}_{ij}^{(t)} = \hat{\mathbf{x}}_{ij}(\mathbf{W}^{(t)}), \tag{20}$$

where $\mathbf{W}^{(t)}$ are the weights at iteration t.

Following CCCP and constructing the linear upper bound for the concave part $-g_{ijk}$, we get

$$1 - \xi_{ijk} + \max_{\mathbf{x} \in X_i} \mathbf{w}_k \cdot \mathbf{x} - \mathbf{w}_j \cdot \hat{\mathbf{x}}_{ij}^{(t)} \leq 0. \tag{21}$$

Hence CCCP solves the following sequence of convex problems

$$\min_{\mathbf{W}, \xi} \quad \frac{\lambda}{2} ||\mathbf{W}||^2 + \sum_{ijk} \beta_i \xi_{ijk} \tag{22}$$

$$s.t. \quad 1 - \xi_{ijk} + \max_{\mathbf{x} \in X_i} \mathbf{w}_k \cdot \mathbf{x} - \mathbf{w}_j \cdot \hat{\mathbf{x}}_{ij}^{(t)} \leq 0, \text{ for } i = 1, \ldots, n,\ j \in Y_i, k \notin Y_i. \tag{23}$$

$$\xi_{ijk} \geq 0 \tag{24}$$

This problem could be further simplified to a convex QP, but we instead solve the equivalent unconstrained convex problem (because it is amenable to efficient primal subgradient descent):

$$h_{RL,cccp}^{(t)}(\mathbf{W}) = \frac{\lambda}{2} ||\mathbf{W}||^2 + \sum_{ijk} \beta_i \max\{0, 1 + \max_{\mathbf{x} \in X_i} \mathbf{w}_k \cdot \mathbf{x} - \mathbf{w}_j \cdot \hat{\mathbf{x}}_{ij}^{(t)}\}. \tag{25}$$

We discuss methods for solving (25) in Section 4.3.4. In summary, the trick we use to efficiently solve the original nonconvex rank-loss SIM objective with the `max` model (or potentially any convex model) is to convert it to an equivalent constrained problem, construct the CCCP upper bound, then convert the bound back to an unconstrained problem.

4.3.3. Heuristic for Nonconvex Aggregation Models. If $F_j(X_i)$ is not a convex function of $\mathbf{W}$, then the constraint (17) does not naturally decompose into a DC function, and CCCP cannot be applied. This is the situation with the `softmax` model. In this case, we propose a heuristic of alternating between computing the support instances, and solving a convex problem where the support instances are treated as constant. The heuristic is similar to CCCP, except the convex objective solved in each step changes to

$$h_{RL,h}^{(t)}(\mathbf{W}) = \frac{\lambda}{2} ||\mathbf{W}||^2 + \sum_{ijk} \beta_i \max\{0, 1 + \mathbf{w}_k \cdot \hat{\mathbf{x}}_{ik}^{(t)} - \mathbf{w}_j \cdot \hat{\mathbf{x}}_{ij}^{(t)}\}. \tag{26}$$

In contrast with the CCCP algorithm, this heuristic is not proven to decrease objective (4) monotonically. The difference between the convex problems solved by CCCP and the heuristic is whether the aggregation model applied to bags with negative labels $F_k(X_i)$ is linearized in terms of the support instance, or not. The heuristic can also be applied when the aggregation model is convex.

ALGORITHM 1: Projected Subgradient with Pegasos Learning Rate

Input: Initial point $\mathbf{W}^{(0)} \in S$, number of iterations K
for $\tau = 1, \ldots, K$ **do**
  Compute a subgradient $\mathbf{V} \in \partial h(\mathbf{W}^{(\tau-1)})$
  $\mathbf{W}^{(\tau)} \leftarrow P[\mathbf{W}^{(\tau-1)} - \frac{1}{\lambda\tau}\mathbf{V}]$
end
return $\mathbf{W}_{best} = \arg\min_{\mathbf{W}^{(\tau)}} h(\mathbf{W}^{(\tau)})$

4.3.4. Subgradient Descent. To solve the convex problem in each step of CCCP or the heuristic [(25) or (26)], we use a subgradient descent method similar to Pegasos, an algorithm for training linear two-class SVMs [Shalev-Shwartz et al. 2007]. The Pegasos algorithm is based on a general framework for optimizing regularized convex objectives [Shalev-Shwartz and Singer 2007]. This framework can be applied to convex optimization problems in the form:

$$\min_{\mathbf{W}\in S} h(\mathbf{W}) \text{ where } h(\mathbf{W}) = \frac{\lambda}{2}||\mathbf{W}||^2 + loss(\mathbf{W}).$$

For such problems, Algorithm 1 is efficient. Let a convex set S be the feasible space; $P[\mathbf{W}] = \arg\min_{\mathbf{W}'\in S} ||\mathbf{W} - \mathbf{W}'||^2$ is a projection back into the feasible space. Note that when S is a ball of radius r, i.e. $S = \{\mathbf{W} : ||\mathbf{W}|| \leq r\}$, the projection simplifies to $P[\mathbf{W}] = \min\{1, \frac{r}{||\mathbf{W}||}\}\mathbf{W}$ (for reasons that become evident in the runtime analysis, we will introduce such a constraint into our optimization problems).

Consider iteration t of CCCP or the heuristic, where the convex objective is $h^{(t)}_{RL,cccp}(\mathbf{W})$ (25) or $h^{(t)}_{RL,h}(\mathbf{W})$ (26) respectively. We will show how to apply subgradient descent to these objectives. We will start by summarizing relevant notation and stating a subgradient of each objective.

First consider $h_{RL,h}$ as an example. Recall that the decision variable $\mathbf{W}$ is the concatenation of components $\mathbf{w}_q$ for each class $q = 1, \ldots, c$. We denote the component of the subgradient of $h^{(t)}_{RL,h}$ corresponding to $\mathbf{w}_q$ as $\mathbf{v}^{(t,\tau),q}_{RL,h}$, where the superscript t indicates the outer iteration of the heuristic, τ indicates the inner iteration of subgradient descent, and q indicates the class. The full subgradient is $\mathbf{V}^{(t,\tau)}_{RL,h} = [\mathbf{v}^{(t,\tau),1}_{RL,h}, \ldots, \mathbf{v}^{(t,\tau),c}_{RL,h}]$. The components of the subgradient of $h^{(t)}_{RL,cccp}$ are defined similarly, e.g., $\mathbf{v}^{(t,\tau),q}_{RL,cccp}$.

For the heuristic, it is sufficient to compute the support instances $\hat{\mathbf{x}}^{(t)}_{ij}$ once at the beginning of each outer iteration t; then at every inner iteration τ of subgradient descent, the subgradients depend only on $\hat{\mathbf{x}}^{(t)}_{ij}$. The subgradient for the heuristic is

$$\mathbf{v}^{(t,\tau),q}_{RL,h} = \lambda\mathbf{w}^{(t,\tau)}_q + \sum_{ijk} \beta_i I[1 + \mathbf{w}^{(t,\tau)}_k \cdot \hat{\mathbf{x}}^{(t)}_{ik} - \mathbf{w}^{(t,\tau)}_j \cdot \hat{\mathbf{x}}^{(t)}_{ij} > 0] \begin{cases} \hat{\mathbf{x}}^{(t)}_{iq} & \text{if } q = k \\ -\hat{\mathbf{x}}^{(t)}_{iq} & \text{if } q = j \\ 0 & \text{otherwise} \end{cases} . \quad (27)$$

For CCCP, the subgradients depend on both $\hat{\mathbf{x}}^{(t)}_{ij}$, and a different set of support instances that change with each inner iteration τ of sub-gradient descent. We introduce the notation

$$\hat{\mathbf{x}}^{(t,\tau)}_{ik} = \hat{\mathbf{x}}_{ik}(\mathbf{W}^{(t,\tau)}) \quad (28)$$

This notation indicates the support instance computed from the weights at iteration τ of subgradient descent within iteration t of CCCP, in contrast with $\hat{\mathbf{x}}_{ik}^{(t)}$, which indicates the support instance computed from the weights at the start of iteration t of CCCP. The subgradient for CCCP at the inner iteration τ is

$$\mathbf{v}_{RL,cccp}^{(t,\tau),q} = \lambda \mathbf{w}_q^{(t,\tau)} + \sum_{ijk} \beta_i I[1 + \mathbf{w}_k^{(t,\tau)} \cdot \hat{\mathbf{x}}_{ik}^{(t,\tau)} - \mathbf{w}_j^{(t,\tau)} \cdot \hat{\mathbf{x}}_{ij}^{(t)} > 0] \begin{cases} \hat{\mathbf{x}}_{iq}^{(t,\tau)} & \text{if } q = k \\ -\hat{\mathbf{x}}_{iq}^{(t)} & \text{if } q = j \\ 0 & \text{otherwise} \end{cases} . \quad (29)$$

We have not discussed how to compute the subgradients efficiently, or explained why a ball-constraint is introduced. These subjects fit naturally into a discussion of the runtime of subgradient descent.

Runtime of Subgradient Descent. Let $\mathbf{W}^*$ be the solution, i.e. $\mathbf{W}^* = \arg\min_{\mathbf{W} \in S} h(\mathbf{W})$. Shalev-Shwartz and Singer [2007] showed the following convergence rate for Algorithm 1,

$$\min_{\tau} h(\mathbf{W}^{(\tau)}) \leq h(\mathbf{W}^*) + O(\frac{\log K}{K} \frac{L}{\lambda}),$$

where L is a constant bounding the magnitude of the subgradient, i.e. $\forall \tau, ||\mathbf{V}||^2 \leq L$. For practical purposes, the number of iterations of sub-gradient descent K is small enough to treat $\log K$ as constant. Hence, to obtain a solution that is within ϵ of optimal, it suffices to run $K \approx O(\frac{L}{\lambda \epsilon})$ iterations of the preceding algorithm [Shalev-Shwartz and Singer 2007].

To prove a rate of convergence for subgradient descent, we must establish L, the bound on the subgradient square magnitude. We begin with the following Lemma.

LEMMA 4.1. *Consider any objective of the form* $h(\mathbf{W}) = \frac{\lambda}{2}||\mathbf{W}||^2 + loss(\mathbf{W})$*, such that* $loss(\mathbf{W}) \geq 0$ *and* $loss(\mathbf{0}) = 1$*. Let the solution be* $\mathbf{W}^* = \arg\min_{\mathbf{W}} h(\mathbf{W})$*. Then* $||\mathbf{W}^*||^2 \leq \frac{2}{\lambda}$.

PROOF. The solution must be at least as good as $\mathbf{W} = \mathbf{0}$, therefore $h(\mathbf{W}^*) \leq 1$. Furthermore, $loss(\mathbf{W}^*) \leq 1$ (assuming the contrary implies $h(\mathbf{W}^*) > 1$, which is a contradiction). Because the loss is nonnegative, $0 \leq loss(\mathbf{W}^*) \leq 1$. We will use this property to finish the proof:

$$\begin{aligned} h(\mathbf{W}^*) = \frac{\lambda}{2}||\mathbf{W}^*||^2 + loss(\mathbf{W}^*) &\leq 1 \\ \frac{\lambda}{2}||\mathbf{W}^*||^2 &\leq 1 - loss(\mathbf{W}^*) \leq 1 \\ ||\mathbf{W}^*||^2 &\leq \frac{2}{\lambda}. \end{aligned}$$

□

Now we will briefly derive a bound $L = c\left(\sqrt{2\lambda} + R\right)^2$. The $h_{RL,cccp}^{(t)}(\mathbf{W})$ and $h_{RL,h}^{(t)}(\mathbf{W})$ objectives satisfy the criteria of Lemma 4.1, so we can replace unconstrained minimization with minimization restricted to the set $S = \{\mathbf{W} : ||\mathbf{W}||^2 \leq \frac{2}{\lambda}\}$ without changing the solution. It is also necessary to bound the magnitude of an instance feature vector, $||\mathbf{x}|| \leq R$. Note that $\mathbf{W} \in S$ implies $||\lambda \mathbf{w}_q|| \leq \lambda \sqrt{\frac{2}{\lambda}} = \sqrt{2\lambda}$. By the triangle inequality,

ALGORITHM 2: Subgradient $\mathbf{V}_{RL,h}^{(t,\tau)}$

Input: $\hat{\mathbf{x}}_{ij}^{(t)}$, $\mathbf{W}^{(t,\tau)}$, $\{(X_i, Y_i)\}_{i=1}^{n}$
for $q = 1, \ldots, c$: **do**
  $\mathbf{v}_q \leftarrow \lambda \mathbf{w}_q^{(t,\tau)}$
end
for $i = 1, \ldots, n$; $j \in Y_i, k \in \bar{Y}_i$ **do**
  if $1 + \mathbf{w}_k^{(t,\tau)} \cdot \hat{\mathbf{x}}_{ik}^{(t)} - \mathbf{w}_j^{(t,\tau)} \cdot \hat{\mathbf{x}}_{ij}^{(t)} > 0$ **then**
    $\mathbf{v}_j \leftarrow \mathbf{v}_j - \beta_i \hat{\mathbf{x}}_{ij}^{(t)}$
    $\mathbf{v}_k \leftarrow \mathbf{v}_k + \beta_i \hat{\mathbf{x}}_{ik}^{(t)}$
  end
end
return $\mathbf{V}_{RL,h}^{(t,\tau)} = [\mathbf{v}_1, \ldots, \mathbf{v}_c]$

ALGORITHM 3: Subgradient $\mathbf{V}_{RL,cccp}^{(t,\tau)}$

Input: $\hat{\mathbf{x}}_{ij}^{(t)}$, $\hat{\mathbf{x}}_{ij}^{(t,\tau)}$, $\mathbf{W}^{(t,\tau)}$, $\{(X_i, Y_i)\}_{i=1}^{n}$
for $q = 1, \ldots, c$: **do**
  $\mathbf{v}_q \leftarrow \lambda \mathbf{w}_q^{(t,\tau)}$
end
for $i = 1, \ldots, n$; $j \in Y_i, k \in \bar{Y}_i$ **do**
  if $1 + \mathbf{w}_k^{(t,\tau)} \cdot \hat{\mathbf{x}}_{ik}^{(t,\tau)} - \mathbf{w}_j^{(t,\tau)} \cdot \hat{\mathbf{x}}_{ij}^{(t)} > 0$ **then**
    $\mathbf{v}_j \leftarrow \mathbf{v}_j - \beta_i \hat{\mathbf{x}}_{ij}^{(t)}$
    $\mathbf{v}_k \leftarrow \mathbf{v}_k + \beta_i \hat{\mathbf{x}}_{ik}^{(t,\tau)}$
  end
end
return $\mathbf{V}_{RL,cccp}^{(t,\tau)} = [\mathbf{v}_1, \ldots, \mathbf{v}_c]$

$||\mathbf{v}_{RL,cccp}^{(t,\tau),q}|| \leq \sqrt{2\lambda} + R$. Summing over classes, $||\mathbf{V}_{RL,cccp}^{(t,\tau)}||^2 = \sum_{q=1}^{c} ||\mathbf{v}_{RL,cccp}^{(t,\tau),q}||^2 \leq c\left(\sqrt{2\lambda} + R\right)^2$.

With a more elaborate derivation, we obtain a tighter bound of $L' = \left(\sqrt{2\lambda} + 2R\right)^2$ (see Appendix 1), which gives a shorter runtime. The bound L' is applicable for both the heuristic and CCCP.

To compute $\mathbf{V}_{RL,h}^{(t,\tau)}$ or $\mathbf{V}_{RL,cccp}^{(t,\tau)}$, one could compute each component according to Equation (27) or (29), but this will take $O(nc^3)$ time. Algorithms (2) and (3) compute the subgradients in $O(nc^2)$ time (assuming the support instances have already been calculated). Note that updating the support instances takes $O(m)$ time, where m is the number of instances in all bags. For the heuristic method, computing the support instances is done once at the beginning of each outer iteration, hence it is not part of the runtime for a single iteration of subgradient descent. In contrast, the CCCP method must recompute the support instances in each iteration of subgradient descent, so the runtime of one iteration of subgradient descent is $O(m + nc^2)$. Running $T = O(\frac{L'}{\lambda\epsilon})$ iterations of subgradient descent gives a runtime of $O(m + \frac{nc^2R^2}{\epsilon\sqrt{\lambda}})$ time to find an ϵ-suboptimal solution for the heuristic method, or $O(\frac{(m+nc^2)R^2}{\epsilon\sqrt{\lambda}})$ for the CCCP method.

4.3.5. Summary of Differences Between CCCP and the Heuristic. Algorithms (4) and (5) list the heuristic and CCCP optimization methods for the rank-loss SIM objective. We refer to these algorithms as SIM-Heuristic and SIM-CCCP. A major difference between SIM-CCCP and SIM-Heuristic is that in SIM-CCCP, the support instances must be recomputed in each iteration of subgradient descent, and the subgradient depends on these different support instances. The heuristic also runs a constant number of iterations of subgradient descent, whereas the CCCP version runs enough to ensure monotonic decrease of the objective (see Appendix 2 for further discussion of monotonicity). The heuristic can be applied to either the `max` or `softmax` model, and the CCCP algorithm can only be applied to the `max` model because `softmax` is nonconvex.

Because the general rank-loss objective is nonconvex, the starting weights may affect the optimum that is found by SIM-CCCP or SIM-Heuristic. We discuss starting weight construction in Appendix 2.

ALGORITHM 4: SIM-Heuristic with rank-loss and `max` or `softmax` model

Input: T, K, λ, MIML dataset $\{(X_i, Y_i)\}_{i=1}^{n}$
for $t = 1, \ldots, T$ **do**
  if $t = 1$ **then**
    $\mathbf{W}^{(t)} = \mathbf{0}$
    $\forall (i,j) : \hat{\mathbf{x}}_{ij}^{(t)} = \frac{1}{n_i} \sum_{\mathbf{x}_{iq} \in X_i} \mathbf{x}_{iq}$
  else
    $\forall (i,j)$: compute $\hat{\mathbf{x}}_{ij}^{(t)}$ using the `max` or `softmax` model
  end
  $\mathbf{W}^{(t,1)} = \mathbf{W}^{(t)}$
  for $\tau = 1, \ldots, K$ **do**
    Compute $\mathbf{V} = \mathbf{V}_{RL,h}^{(t,\tau)}$ with Algorithm 2
    $\mathbf{W} = \mathbf{W}^{(t,\tau)} - \frac{1}{\lambda \tau} \mathbf{V}$
    $\mathbf{W}^{(t,\tau+1)} = \min\{1, \sqrt{\frac{2}{\lambda}} / ||\mathbf{W}||\} \mathbf{W}$
  end
  $\tau^* = \arg\min_{\tau} h_{RL,h}^{(t)}(\mathbf{W}^{(t,\tau)})$
  $\mathbf{W}^{t+1} = \mathbf{W}^{(t,\tau^*)}$
end
return $W^{(T+1)}$

4.4. Nonlinear Classification via Kernels

So far we have only considered linear classifiers. There are several ways to extend our proposed methods to nonlinear classification via kernels. One way is to use the standard dual kernel trick. Shalev-Shwartz et al. [2011] suggest a different trick for Pegasos that could be applied to our proposed methods as well. The key idea in kernelizing Pegasos is to observe that all changes to the weights in the primal algorithm are either adding a linear multiple of instance features or multiplying by a scalar. We could redefine $f_j(\mathbf{x}) = \sum_{i=1}^{m} \alpha_{ij} K(\mathbf{x}_i, \mathbf{x})$ and proceed with the primal algorithm, but make equivalent changes to α rather than $\mathbf{W}$. Both of these methods have the disadvantage of increasing the asymptotic complexity to solve the optimization problem.

We instead use the random Fourier feature method of Rahimi and Recht [2007] (Algorithm (6)), which approximates a kernel while preserving linear runtime. The main idea in this method is to transform the feature vectors first, then apply exactly the same linear training algorithm. The feature vector $\mathbf{x}$ is transformed to a feature $\mathbf{z}(\mathbf{x})$ such that with high probability $\mathbf{z}(\mathbf{x}) \cdot \mathbf{z}(\mathbf{y}) \approx k(\mathbf{x}, \mathbf{y})$, where $k(\mathbf{x}, \mathbf{y}) = k(\mathbf{x} - \mathbf{y})$ is a shift invariant kernel. This method can be applied with the radial basis function (RBF) kernel $k(\mathbf{x}, \mathbf{y}) = \exp\{-\gamma ||\mathbf{x} - \mathbf{y}||^2\}$. To use this algorithm with the RBF kernel k, we need to compute its Fourier transform $p(\omega)$ and draw D independently, identically, distributed samples $\omega_1, \ldots, \omega_D \in \mathbb{R}^d$ from $p(\omega)$, where D is a parameter that can be adjusted to control approximation accuracy. It can be shown that for the RBF kernel with parameter γ, this is equivalent to sampling each of the d components of ω_i from a normal distribution with 0 mean and variance 2γ.

5. EXPERIMENTS

Our experiments compare different loss functions, aggregation models, and optimization methods. We also investigate the effectiveness of random Fourier kernel features for nonlinear classification.

ALGORITHM 5: SIM-CCCP with rank loss and `max` model

Input: T, K, K_{max}, λ, MIML dataset $\{(X_i, Y_i)\}_{i=1}^{n}$
for $t = 1, \ldots, T$ **do**
  if $t = 1$ **then**
    $\mathbf{W}^{(t)} = \mathbf{0}$
    $\forall(i, j) : \hat{\mathbf{x}}_{ij}^{(t)} = \frac{1}{n_i} \sum_{\mathbf{x}_{iq} \in X_i} \mathbf{x}_{iq}$
  else
    $\forall(i, j) : \hat{\mathbf{x}}_{ij}^{(t)} = \arg\max_{\mathbf{x}_{iq} \in X_i} \mathbf{w}_j \cdot \mathbf{x}_{iq}$
  end
  $\mathbf{W}^{(t,1)} = \mathbf{W}^{(t)}$
  $improved = False$
  $\tau_{total} = 0$
  while $(\neg improved) \wedge \tau_{total} < K_{max}$ **do**
    for $\tau = 1, \ldots, K$ **do**
      $\tau_{total} = \tau_{total} + 1$
      $\forall(i, j) : \hat{\mathbf{x}}_{ij}^{(t,\tau)} = \arg\max_{\mathbf{x}_{iq} \in X_i} \mathbf{w}_j \cdot \mathbf{x}_{iq}$
      Compute $\mathbf{V} = \mathbf{V}_{RL,cccp}^{(t,\tau)}$ with Algorithm 3
      $\mathbf{W} = \mathbf{W}^{(t,\tau)} - \frac{1}{\lambda \tau_{total}} \mathbf{V}$
      $\mathbf{W}^{(t,\tau+1)} = \min\{1, \sqrt{\frac{2}{\lambda}} / ||\mathbf{W}||\} \mathbf{W}$
    end
    $\tau^* = \arg\min_{\tau} h_{RL,cccp}^{(t)}(\mathbf{W}^{(t,\tau)})$
    $\mathbf{W}^{t+1} = \mathbf{W}^{(t,\tau^*)}$
    if $h_{RL,cccp}^{(t)}(\mathbf{W}^{(t+1)}) < h_{RL,cccp}^{(t)}(\mathbf{W}^{(t)})$ **then**
      $improved = True$
    end
  end
end
return $W^{(T+1)}$

ALGORITHM 6: Random Fourier Kernel Features [Rahimi and Recht 2007]

Input: Positive definite shift-invariant kernel $k(\mathbf{x}, \mathbf{y}) = k(\mathbf{x} - \mathbf{y})$, feature dimension d, parameter D
Compute the Fourier transform p of k: $p(\omega) = \frac{1}{2\pi} \int e^{-j\omega \cdot \Delta} k(\Delta) d\Delta$
Draw D i.i.d. samples $\omega_1, \ldots, \omega_D \in \mathbb{R}^d$ from $p(\omega)$
Let $\mathbf{z}(\mathbf{x}) = \sqrt{\frac{1}{D}} [cos(\omega_1 \cdot \mathbf{x}), sin(\omega_1 \cdot \mathbf{x}), \ldots, cos(\omega_D \cdot \mathbf{x}), sin(\omega_D \cdot \mathbf{x})]$

5.1. Experimental Setup

This section discusses the datasets, baseline methods, parameter optimization, and transductive or inductive modes used in our experiments.

5.1.1. Datasets. Table II summarizes the properties of each dataset used in our experiments. All of these datasets are available online.[1]

[1] http://web.engr.oregonstate.edu/~briggsf/kdd2012datasets

Table II. MIML Datasets

Dataset	Classes	Dimensions	Bags	Instances
HJA Birdsong	13	38	548	10,232
MSRC v2	23	48	591	1,758
Letter-Carroll	26	16	166	717
Letter-Frost	26	16	144	565
Pascal VOC 2012	20	48	1,053	4,143

HJA Bird Song. Our collaborators have collected audio recordings of bird song at the H. J. Andrews (HJA) Long Term Ecological Research Site, using unattended omnidirectional microphones. Our goal is to automatically identify the species of bird responsible for each utterance in these recordings, thereby generating an automatic acoustic survey of bird populations. This problem is a natural fit for the MIML instance annotation framework. We treat a 10-second audio recording as a bag with labels corresponding to the set of species present in the recording. The instances are segments in a spectrogram. A spectrogram is a graph of the spectrum of a signal as a function of time (computed by applying the Fast Fourier Transform to successive overlapping frames of the audio signal). Figure 1 shows an example spectrogram for a 10-second audio recording containing several species of birds.

Starting with a 10-second audio recording, we first convert it to a spectrogram. A series of preprocessing steps are then applied to the spectrogram to reduce noise, and to identify bird song segments in the audio. Each segment is considered an instance and described by a 38-dimensional feature vector characterizing the shape of the segment, its time and frequency profile statistics, and a histogram of gradients.

This dataset contains 548 10-second recordings (bags), and a total of 10,232 segments (instances), of which 4998 are labeled, and the rest are unlabeled. The available instance labels were provided by a human expert. The spectrograms were originally labeled by manually drawing bounding boxes (not shown) around regions of the spectrogram corresponding to bird sound, and giving a single species label to each box. The bag-level label sets were formed by taking the union of these bounding box labels. The instances (segments) are produced by an automatic segmentation/detection algorithm, which does not necessarily match the manually drawn boxes. The labels for instances are obtained by matching each segment with the bounding box that overlaps with it most. If a segment does not overlap with any bounding box, it is designated as unlabeled. Further details on the collection of audio and feature extraction for this dataset are available in Briggs et al. [2012b].

This dataset presents some deviations from the standard MIML assumption that a bag's label set is the union of its instance labels. First, there are unlabeled instances that may not be accounted for in the bag label set. Second, sometimes when multiple birds make sounds that directly overlap in the spectrogram, the segmentation algorithm may fail to separate those sounds, and create a segment that is given a single label although it actually represents two species. In rare cases, this can lead to labels in a bag label set that do not correspond with any instance (according to the instance-level labels).

One of the goals of our experiments is to evaluate how various instance annotation algorithms handle the issue of unlabeled instances. Toward this goal, we consider two variants of this dataset: "filtered" and "unfiltered". For the filtered variant, all of the unlabeled instances are removed, and in the unfiltered variant they are left in during the training process. In both variants, the accuracy is measured only on the labeled instances.

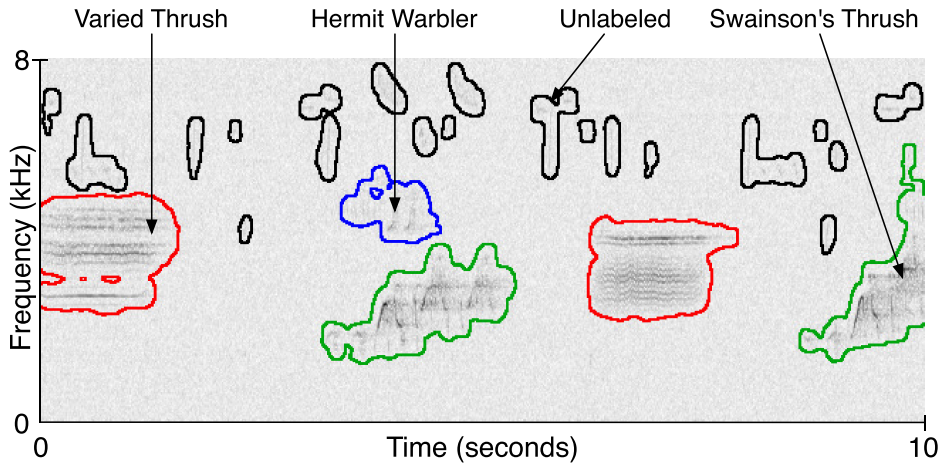

Fig. 1. An example spectrogram from the HJA Birdsong dataset. This spectrogram corresponds to one bag. Each outlined region is an instance.

Fig. 2. An image from MSRC v2 and the corresponding pixel-level labeling. The classes in this image are "sky", "trees", "grass", "body", and "car". The black regions are 'void'; we discard void regions.

Image Dataset: MSRC v2. A subset of the Microsoft Research Cambridge (MSRC) image dataset[2] [Winn et al. 2005] named "v2" contains 591 images and 23 classes. The MSRC v2 dataset is useful for the instance annotation problem, because pixel-level labels are included (Figure 2). Several authors used MSRC v2 in MIML experiments [Vijayanarasimhan and Grauman 2009; Yakhnenko 2009; Zha et al. 2008].

We construct a MIML dataset from MSRC v2 as follows. We treat each image as a bag. The bag label set is the list of all classes present in the ground-truth segmentation (the union of the instance labels). The instances correspond to each contiguous region in the ground-truth segmentation (to simplify the experiment, we use the ground-truth segmentation rather than automatic segmentation). Each instance is described by a 16-dimensional histogram of gradients [Dalal and Triggs 2005], and a 32-dimensional histogram of colors.

[2]http://research.microsoft.com/en-us/projects/objectclassrecognition/default.htm

Fig. 3. An image from PASCAL VOC 2012, the corresponding segmentation into instances, and the instance labels.

PASCAL Visual Object Challenge 2012. The PASCAL Visual Object Challenge (VOC) 2012 Segmentation dataset [Everingham et al. 2010] consists of images with per-pixel ground-truth labeling into 20 classes such as "car", "boat", "bird", "person", "cow", and "chair". Each image has a corresponding segmentation into objects, and a class-label for each region (Figure 3). We construct an MIML dataset by treating each image as a bag, and each object as an instance. In the ground-truth segmentation, each object is denoted with a different color; in some cases multiple disconnected regions are grouped as the same object. We construct the same histogram of gradients and histogram of color features as used in the MSRCV v2 dataset to describe the collection of pixels in each object.

Many images in VOC only contain one type of object, but multiple instances. Therefore all of the instances in such images are labeled unambiguously. To make the problem more challenging, we discard all images with a single label. After discarding single-label images, we are left with 1053 images.

Synthetic MIML Datasets. Limited availability of MIML datasets with instance labels has been a barrier to studying instance annotation (because instance labels are needed to evaluate accuracy). Using the Letter Recognition dataset [Frey and Slate 1991] from the UCI Machine Learning repository, we construct two synthetic MIML datasets. The Letter Recognition dataset consists of 20,000 instances with 16-dimensional features, and 26 classes. Note that randomly forming the bags will not be realistic because real-world MIML problems often have correlations between labels. Instead, we generate datasets derived from the words in two poems, "Jabberwocky" [Carroll 1896], and "The Road Not Taken" [Frost 1916]. We call these datasets Letter-Carroll and Letter-Frost. For each word in these poems, we create a bag, with instances corresponding to the letters in the word. For each instance, we sample (without replacement), an example from the Letter Recognition dataset with the corresponding letter. The bag-level labels are the union of the instance labels. For example, the word "diverged" is transformed into a bag with 8 instances, and the label set {d, i, v, e, r, g}.

5.1.2. Transductive vs. Inductive. We consider instance annotation in two different modes: transductive and inductive. In the transductive mode, the goal is to predict the instance labels for bags with known label sets. In this mode, the instance-level classifiers can only predict labels that appear in the bag label set. Formally, the instance classifier in the transductive mode is $f(\mathbf{x}_{iq}) = \arg\max_{j \in Y_i} f_j(\mathbf{x}_{iq})$. In the inductive mode, the goal is to predict instance labels in previously unseen bags (with unknown label sets). There is no restriction on which label an instance may be given in this case.

For both modes, we compute classifier accuracy as the fraction of instances correctly classified. For the inductive mode, we run 10-fold cross-validation and report average accuracy $\pm$ standard deviation in accuracy between folds. We did not use 10-fold cross validation for the VOC dataset, but instead used a partition of the dataset into "train"

and "val" subsets, which was included with the dataset. For the transductive mode, we disregard this partition and use all 1053 images, and for the inductive mode we learn from the "train" subset and evaluate accuracy on the "val" subset. That is why the VOC columns in Tables III(c) and (d) do list a standard deviation.

5.1.3. Baseline Methods. To directly compare our proposed rank loss objective with Hamming loss, we modify the SIM-Heuristic algorithm to use Hamming loss instead of rank loss. We also consider another baseline M^{3}MIML, which is designed to make predictions at the bag level but learns an instance-level model using Hamming loss. We also compare rank loss to the ambiguous loss function used by Cour et al. [2011]. Finally, as a reference we evaluate a SISL SVM classifier, which has the unfair advantage of learning directly from instance labels. In the following we summarize these baseline methods.

Hamming-Loss SIM. To compare Hamming loss to rank loss, we use the following Hamming-loss objective, with $F_j(\cdot)$ instantiated with either the `max` or `softmax` aggregation models.

$$h_{Ham}(\mathbf{W}) = \frac{\lambda}{2}||\mathbf{W}||^2 + \frac{1}{nc}\sum_{i=1}^{n}\sum_{j=1}^{c}\max\{0, 1 - Y_{ij}F_j(X_i)\}, \tag{30}$$

or in terms of support instances,

$$\hat{h}_{Ham}(\mathbf{W}) = \frac{\lambda}{2}||\mathbf{W}||^2 + \frac{1}{nc}\sum_{i=1}^{n}\sum_{j=1}^{c}\max\{0, 1 - Y_{ij}\mathbf{w}_j \cdot \hat{\mathbf{x}}_{ij}\}. \tag{31}$$

To optimize this objective, we use a variation of the SIM-Heuristic algorithm with the subgradient[3]

$$\mathbf{v}_{Ham}^{(t,\tau),q} = \lambda \mathbf{w}_q^{(t,\tau)} - \frac{1}{nc}\sum_{i=1}^{n} I[\,Y_{iq}\mathbf{w}_q^{(t,\tau)} \cdot \hat{\mathbf{x}}_{iq}^{(t)} < 1]\, Y_{iq}\hat{\mathbf{x}}_{iq}^{(t)}. \tag{32}$$

M^{3}MIML Algorithm. M^{3}MIML is a MIML algorithm designed to make predictions at the bag level, however, it learns an instance-level model, which can be used for instance annotation. Similar to our approach, M^{3}MIML learns one linear model per class and uses the `max` bag-level model for F_j. The optimization problem for M^{3}MIML is formulated as minimizing $||\mathbf{W}||^2$, where $\mathbf{W} = [\,\mathbf{w}_1, \ldots, \mathbf{w}_c]$, subject to the constraints of correct output at the bag level,

$$Y_{ij}F_j(X_i) \geq 1 \text{ for } i = 1, \ldots, n,\ j = 1, \ldots, c. \tag{33}$$

Note that the equivalent unconstrained objective is regularized Hamming-loss. If $Y_{ij} = -1$, the constraint is convex, and is converted to a set of linear constraints

$$-\max_{\mathbf{x} \in X_i} \mathbf{w}_j \cdot \mathbf{x} \ \geq\ 1 \tag{34}$$

$$\forall \mathbf{x} \in X_i : -\mathbf{w}_j \cdot \mathbf{x} \ \geq\ 1. \tag{35}$$

[3]Note that (30) satisfies the conditions for Lemma 4.1 and a bound on the magnitude of the subgradient of $L = \frac{c}{2}\left(\sqrt{2\lambda} + \frac{R}{c}\right)^2$ suffices. The proof of this bound is similar to the short derivation in Section 4.3.4.

If $Y_{ij} = +1$, the constraint is nonconvex, and M^{3}MIML's formulation replaces it with a linear upper bound:

$$\max_{\mathbf{x} \in X_i} \mathbf{w}_j \cdot \mathbf{x} \geq \frac{1}{n_i} \sum_{\mathbf{x} \in X_i} \mathbf{w}_j \cdot \mathbf{x} \geq 1. \tag{36}$$

After several more steps (e.g., adding slack variables and switching to the dual), M^{3}MIML is formulated as a QP with linear constraints. Unlike the CCCP approach, only a single QP is solved rather than a sequence.

Ambiguous Label Classification (ALC). Cour et al. [2009] proposed an SVM formulation for the ALC problem. We refer to their algorithm as CLPL because it is implemented in the *Convex Learning from Partial Labels Toolbox* [Cour et al. 2011]. We compare our proposed method to CLPL because they both learn one linear model per class $f_j(\mathbf{x}) = \mathbf{w}_j \cdot \mathbf{x}$ and predict the instance label as $\arg\max_j f_j(\mathbf{x})$, and both use an L_2 regularized loss function. The primary difference in Cour's ALC method is that the loss function is designed for use with ALC data (instead of the bag-level loss functions we use for MIML data). The ALC loss function is a convex upper bound to the 0/1 loss with respect to the true (unknown) instance labels,

$$L(f, \mathbf{x}, Y) = \max\{0, 1 - \frac{1}{|Y|} \sum_{j \in Y} f_j(\mathbf{x})\}^2 + \sum_{j \notin Y} \max\{0, 1 + f_j(\mathbf{x})\}^2.$$

Minimizing this loss function can be accomplished by a reduction to a SISL SVM problem with squared hinge-loss, and solved using an off-the-shelf linear SVM. In our experiments, we use the L_2 regularized square-loss dual solver in LIBLINEAR [Fan et al. 2008] to implement CLPL.

Single Instance Single Label SVM. We also run a standard SISL SVM for the inductive mode, whose performance can be interpreted as an empirical upper bound for inductive instance annotation because it is trained using unambiguously labeled instances. For this experiment, we use LIBSVM [Chang and Lin 2001] with a linear kernel. Note that LIBSVM uses one linear model for each pair of classes rather than one for each class.

5.1.4. Parameter Optimization. The SIM algorithms have a regularization parameter λ. Similarly, CLPL and M^{3}MIML have a regularization parameter C. We repeat all experiments for each value of $\lambda \in \{10^{-6}, \ldots, 10^{-9}\}$, $C \in \{10^1, \ldots, 10^4\}$ for CLPL, and $C \in \{10^{-2}, \ldots, 10^6\}$ for M^{3}MIML, then report the maximum accuracy achieved by each method over all parameters (we refer to this process as post hoc parameter selection). We expect these ranges of parameters to be sufficient based on preliminary experiments in which we used larger ranges [Briggs et al. 2012a].

Where the random Fourier kernel features are used, the parameter $D = 50$ (as in Rahimi and Recht [2007]), and we jointly optimize the regularization parameter and the RBF kernel parameter over a grid with the aforementioned values of λ or C, and $\gamma \in \{10^3, 10^4, 10^5\}$. This range of values for γ was selected by manual tuning.

For the SISL SVM, the regularization parameter C is optimized by nested 10-fold cross-validation (within each fold of 10-fold cross validation, we run 10-fold cross validation in the training set to select the parameter). We search over the range $C \in \{10^1, \ldots, 10^7\}$. With values of C larger than 10^7, LIBSVM becomes prohibitively slow.

The parameters T, K, and K_{max} control the tradeoff between convergence and runtime. These parameters are manually selected by empirically observing the typical

convergence behaviors of different algorithms. In particular, for the SIM heuristic algorithms, we use $T = 10$ outer iterations, with $K = 100$ iterations of subgradient descent. For the CCCP algorithm, we use $T = 10$, $K = 100$, and $K_{max} = 1000$. Figure 4 shows that most improvement in the rank loss objective occurs within $T = 10$ outer iterations. We presented empirical results in Briggs et al. [2012a] showing that $K = 100$ is a reasonable number of iterations of subgradient descent. $K_{max} = 1000$ is chosen rather conservatively, based on empirical observation to ensure monotonicity in CCCP. Note that it is possible further increasing T, K and K_{max} values may lead to slightly better performance due to better convergence, however, we consider these values sufficient for problems with a similar number of classes.

5.2. Results

Accuracy results are listed in Table III. In particular, Tables III(a) and (b) present the accuracy results for the transductive mode (with no kernel, and the RBF kernel respectively). Tables III(c) and (d) present the results for the inductive mode.

Following the recommendations of Demšar [2006] for statistical comparison of multiple classifiers on multiple datasets, we summarize comparisons between two methods using win-tie-loss counts (and do not discard some wins or losses based on a pairwise significance test). Counts are aggregated over all datasets including Birdsong* but not including the unfiltered variant.

5.2.1. Comparison of Loss Functions. We first compare rank loss vs. Hamming loss versions of SIM-Heuristic. This is a direct comparison between the two loss functions because all other aspects are kept the same, i.e. the aggregation model and optimization method. Considering five datasets (all but the unfiltered Birdsong dataset, to avoid the compounding factor of noise instances), two different aggregation models (`max` and `softmax`), and two different settings (transductive and inductive), there are a total of 20 direct comparisons between the two loss functions. The win-tie-loss count for rank-loss vs Hamming loss is 20-0-0, a decisive win for rank loss.

We next compare rank-loss SIM-Heuristic and SIM-CCCP with M^3MIML. Since M^3MIML uses the `max` aggregation model, we compare it with the SIM-heuristic with `max`, resulting in 10 total comparisons, and a win-tie-loss count of 9-0-1 in favor of SIM-Heuristic. For SIM-CCCP vs M^3MIML, the win-tie-loss count is also 9-0-1. Finally, comparing SIM-Heuristic with rank loss and `max` or `softmax` to CLPL (ambiguous loss), the win-tie-loss count is 18-0-2. Overall, these results suggest that rank loss consistently outperforms Hamming or ambiguous loss.

5.2.2. Comparison of Aggregation Models. Next we compare the `max` and `softmax` aggregation models. Focusing on rank loss, we count wins, ties, and losses using the SIM-Heuristic algorithm across five datasets, in transductive and inductive modes, with or without a kernel, resulting in a total of 20 comparisons. The overall win-tie-loss count for `softmax` vs `max` is 13-1-6 in favor of `softmax`. Interestingly, if we focus on only linear models (no kernel features), the win-tie-loss count is 8-1-1, suggesting a dominant win for `softmax`. In contrast, when using kernel features, the count is 5-0-5, indicating a tie between the two aggregation models. We suggest that `softmax` achieves higher accuracy than `max` when using linear models because `softmax` is less sensitive to outliers and noise. We speculate that the difference between `max` and `softmax` is less when a kernel is used because outliers and noise have a local effect on the decision boundaries of the classifier, rather than skewing the boundaries globally as with a linear classifier.

5.2.3. The Effect of Unlabeled Instances. Recall that the Birdsong dataset has a filtered and unfiltered variant (Birdsong* is the filtered variant). The unfiltered variant contains instances that were left unlabeled and are not necessarily accounted for in the

Table III. Instance Annotation Accuracy Results

(a) Transductive accuracy, no kernel. * – filtered variant

Algorithm	Loss	Model	Carroll	Frost	Birdsong*	Birdsong	MSRC v2	VOC
SIM-Heuristic	Rank	max	.719	.780	.815	.801	.699	.633
SIM-CCCP	Rank	max	.744	.805	.816	.803	.720	.634
SIM-Heuristic	Rank	softmax	.721	.814	.815	.810	.718	.630
SIM-Heuristic	Hamming	max	.415	.495	.707	.599	.581	.541
SIM-Heuristic	Hamming	softmax	.500	.548	.781	.603	.603	.557
CLPL	Ambiguous	–	.672	.688	.742	.678	.678	.598
M^3MIML	Hamming	max	.454	.532	.651	–	.547	.533

(b) Transductive accuracy, random Fourier kernel features

SIM-Heuristic	Rank	max	.817	.792	.822	–	.756	.642
SIM-CCCP	Rank	max	.807	.780	.829	–	.798	.623
SIM-Heuristic	Rank	softmax	.794	.819	.833	–	.766	.634

(c) Inductive accuracy ± standard deviation over 10-fold cross validation, no kernel

SIM-Heuristic	Rank	max	.531 ± .054	.562 ± .057	.602 ± .033	.522 ± .032	.442 ± .044	.354
SIM-CCCP	Rank	max	.551 ± .038	.555 ± .065	.607 ± .038	.530 ± .021	.473 ± .029	.350
SIM-Heuristic	Rank	softmax	.540 ± .049	.573 ± .052	.618 ± .041	.556 ± .062	.460 ± .042	.357
SIM-Heuristic	Hamming	max	.114 ± .030	.141 ± .045	.239 ± .051	.133 ± .062	.152 ± .049	.143
SIM-Heuristic	Hamming	softmax	.150 ± .049	.166 ± .062	.342 ± .044	.197 ± .052	.223 ± .034	.215
CLPL	Ambiguous	–	.464 ± .058	.506 ± .063	.620 ± .038	.535 ± .033	.431 ± .036	.345
M^3MIML	Hamming	max	.288 ± .041	.313 ± .041	.433 ± .073	–	.317 ± .055	.396
SISL SVM	Hinge	–	.772 ± .049	.753 ± .038	.772 ± .032	–	.638 ± .045	.440

(d) Inductive accuracy ± standard deviation over 10-fold cross validation, random Fourier kernel features

SIM-Heuristic	Rank	max	.565 ± .060	.590 ± .051	.645 ± .039	–	.499 ± .044	.339
SIM-CCCP	Rank	max	.618 ± .042	.576 ± .065	.630 ± .040	–	.519 ± .044	.343
SIM-Heuristic	Rank	softmax	.596 ± .041	.587 ± .066	.642 ± .039	–	.506 ± .038	.337

bag-level label sets. Note surprisingly, all algorithms suffer some degradation in accuracy in the presence of these unlabeled instances. There are a few interesting points to note. First, the performance degradation for rank loss is often substantially smaller relative to the other loss functions used in the comparison, including both Hamming loss and CLPL. This suggests that rank-loss is more noise robust than other loss functions. We also note that the accuracy of the max model decreases by more than the accuracy of the softmax model; max is known to be sensitive to outliers and noise. In contrast the softmax model is more robust in the presence of noise instances.

5.2.4. Comparison of CCCP and Heuristic Optimization. Because SIM-CCCP is not applicable to the softmax model, we focus on the max model in a comparison of SIM-Heuristic and SIM-CCCP. One point of interest is the convergence of these two algorithms on the nonconvex rank-loss objective. Figure 4 shows the objective h_{RL} vs the number of outer iterations for SIM-Heuristic and SIM-CCCP (recall one outer iteration consists of updating support instances, and solving a convex problem). Observe that CCCP monotonically decreases the objective, while the heuristic occasionally increases the objective [e.g., Figure 4(e)]. CCCP generally achieves lower (better) objective values than the heuristic.

Comparing the accuracy of the two methods over all results in the transductive and inductive modes, with or without a kernel, the count for SIM-CCCP vs SIM-Heuristic

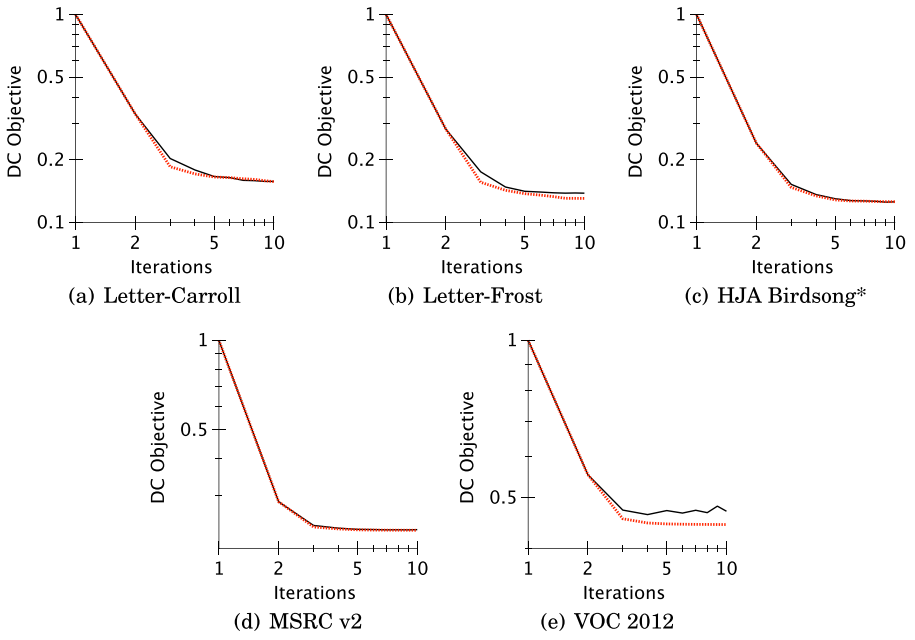

Fig. 4. The objective h_{RL} vs number of iterations of SIM-CCCP and SIM-Heuristic. SIM-CCCP is the dotted line. Transductive mode, `max` model, $\lambda = 10^{-7}$.

is 13-0-7, slightly in favor of CCCP. These results suggest that the lower-objective solutions obtained by CCCP often translate to improved accuracy.

5.2.5. Random Fourier Kernel Features. We now examine accuracy improvements achieved by using random Fourier kernel features [Rahimi and Recht 2007]. Results using this method are listed in Tables III(b) and (d). From these results we see that the random Fourier kernel features often improve accuracy, sometimes by as much as 10% (e.g., transductive SIM-Heuristic with `max` on the Letter-Carroll dataset). More specifically, we compare the results obtained using kernel features (Tables III(b) and (d)) against the results of corresponding linear methods (the first three rows of Tables III(a) and (c) respectively), resulting in a total of 30 comparisons. The aggregated win-tie-loss counts for kernel vs no kernel features is 25-0-5 in favor of kernel features.

5.2.6. Parameter Selection by Cross-Validation. In all of the results discussed so far, we use post-hoc parameter selection. In other words, we run the whole experiment multiple times with different parameters, and report the result with the best instance-level accuracy. This parameter selection is done the same way for all algorithms (including CLPL and M^{3}MIML), so no algorithm gains an unfair advantage. However, this method cannot be used in practice unless some instance labels are known. One might label a relatively small number of instances specifically for this purpose.

In a scenario where no instance labels are known, cross-validation cannot be used to select the regularization parameter λ that gives the best instance-level accuracy. We propose instead to use cross-validated rank loss as a selection criteria for λ as follows:

—*Apply κ-fold cross-validation*. For each fold $l = 1, \ldots, \kappa$, partition the dataset into two sets *Train*(l) and *Test*(l).

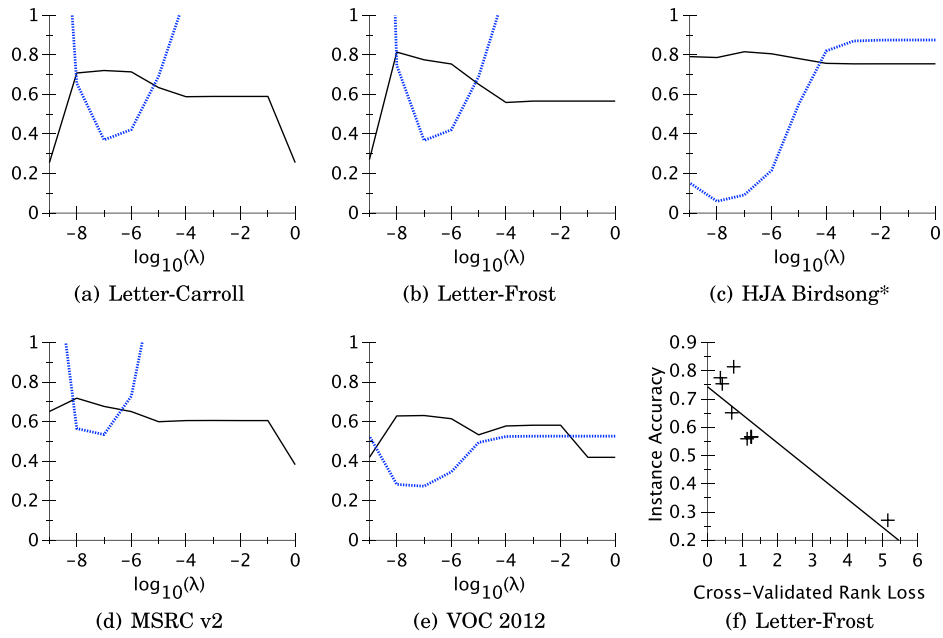

Fig. 5. (a–e) Solid line—accuracy vs λ. Dotted line—$RL(\lambda)$, i.e. cross-validated bag-level rank loss. (f) Scatter plot of accuracy and $RL(\lambda)$ for the Letter-Frost dataset.

— In each fold l, train an SIM on $Train(l)$, then compute unregularized rank loss on $Test(l)$. Compute the average rank loss over all folds as a function of λ,

$$RL(\lambda) = \frac{1}{\kappa}\sum_{l=1}^{\kappa}\sum_{i\in Test(l)}\sum_{j\in Y_i, k\notin Y_i}\beta_i \max\{0, 1-\big(F_j(X_i)-F_k(X_i)\big)\}. \tag{37}$$

— Select the λ that gives the lowest loss averaged over all folds,

$$\hat{\lambda} = \arg\min_{\lambda} RL(\lambda). \tag{38}$$

Note that we did not use this method in other experiments because it increases runtime by an order of magnitude compared to post hoc selection (applying this method in the transductive mode involves cross-validation; in the inductive mode it would be nested cross-validation, e.g., 10-fold CV within each fold of 10-fold CV). Cour et al. [2011] proposed a similar parameter selection scheme for ALC wherein the regularization parameter is selected by cross-validated ambiguous loss.

To test our proposed parameter selection method within a reasonable amount of time, we focus on the transductive mode. Using the SIM-Heuristic algorithm with `softmax`, we vary $\lambda \in \{10^{-9}, \ldots, 10^{0}\}$ and at each value of λ, compute the instance-level accuracy and cross-validated rank loss. Figure 5(a–e) shows the instance-level accuracy and cross-validated rank-loss as a function of λ for each dataset. From these results, we see that the value of λ selected by minimizing cross-validated rank loss is generally close to the value of λ that maximizes instance-level accuracy. Figure 5(f) shows an example scatter plot of accuracy and cross-validated rank loss from the Letter-Frost dataset. Each point corresponds to one value of λ. The line is a linear least-squares fit. This plot shows that lower cross-validated rank loss generally corresponds with higher instance accuracy. The scatter plots are similar for the other

Table IV. Empirical Runtime (Seconds), Transductive, No Kernel, $\lambda = 10^{-7}$

Algorithm	Loss	Model	Carroll	Frost	Birdsong*	MSRC v2	VOC
SIM-Heuristic	Rank	`max`	12.6	9.8	21.0	64.1	139.1
SIM-Heuristic	Rank	`softmax`	13.6	10.2	23.3	68.9	147.1
SIM-CCCP	Rank	`max`	53.7	59.8	132.8	269.9	443.3

datasets. We find experimentally that the proposed method is reasonably effective for selecting the regularization parameter in the absence of any instance labels.

5.2.7. Empirical Runtime. Table IV gives empirical runtimes in seconds[4] for our proposed methods. These results are obtained in the transductive mode, with a fixed regularization parameter $\lambda = 10^{-7}$, and no kernel. In all cases, the runtime is on the order of minutes or seconds. The runtime using the `softmax` model is slightly more than using the `max` model. The runtime for CCCP is longer than the heuristic because it recomputes support instances in every iteration of subgradient descent, and runs more iterations of subgradient descent to ensure monotonicity.

5.2.8. Overall Summary. Comparing our proposed methods SIM-CCCP with `max` and SIM-Heuristic with `softmax`, neither is best all the time. Generally, we observe that SIM-CCCP with `max` achieves higher accuracy when bag label sets are exactly equal to the union of instance labels. When this assumption does not hold, SIM-Heuristic with `softmax` tends to achieve higher accuracy. SIM-Heuristic with `softmax` provides the best balance of runtime and accuracy.

SISL SVM in the inductive mode [Table III(c)] achieves a much higher accuracy than any of the MIML algorithms. This result is expected, as the SISL SVM has access to labeled instances when training, while the MIML algorithms do not. However, this accuracy gap suggests that there is still a lot of room for improvement in the instance annotation algorithms.

6. RELATED WORK

MIML algorithms are developed under multiple frameworks, some of which naturally lend themselves to instance annotation. One such framework is graphical models, which have been previously used to perform bag and instance-level classification. Such models often treat instance labels as latent variables. Inference over such models allows the classification of instances. While a variety of algorithms exists, we highlight some representative examples of recent work. Dirichlet-Bernoulli Alignment [Yang et al. 2009] and the Exponential Multinomial Mixture model [Yang et al. 2010] are topic models for MIML datasets and use variational inference to perform instance labeling. Zha et al. [2008] proposed the MLMIL algorithm, a conditional random field model for MIML image annotation that uses Gibbs sampling to infer instance labels. Du et al. [2009] proposed another application of graphical models to simultaneous image annotation and segmentation.

While graphical models offer intuitive probabilistic interpretation, the computational complexity of inference in such models is one of the standing challenges. In this work, we focus on another class of MIML approaches based on regularized loss minimization. Hamming-Loss SIM can be viewed as an alternative approach for optimizing a similar objective to the M^{3}MIML algorithm [Zhang and Zhou 2008] (which learns an instance-level model, but was not designed for instance annotation). Also note that

[4]These runtimes are measured on a 2010 MacPro with 2.4 GHz Intel Xeon processor and 16 GB of 1066 MHz DDR3 memory. The algorithms are implemented in C++ and compiled with GCC 4.0.

CLPL follows the same framework of regularized loss minimization (but using a loss function designed for ALC).

There are a small number of other works that address MIML instance annotation. Vijayanarasimhan and Grauman [2009] developed an MIML SVM that learns a bag-level model with a set kernel (it does not learn a model of the instance feature space). Their algorithm makes predictions at either the bag or instance level by treating an instance as a bag of one instance. Vezhnevets et al. [2010] proposed an algorithm for instance annotation in MIML data, where images are represented as a bag of pixels. Their algorithm alternates between sampling instance labels from an estimated distribution, and training an ensemble of decision trees on the sampled labels. Similarly, Nguyen [2010] proposed an MIML SVM algorithm that alternates between assigning instance labels and maximizing margin, given the assigned labels (although they did not conduct experiments on instance annotation).

Similar to our approach, the MI-SVM algorithm [Andrews et al. 2002] for multi-instance learning (MIL) uses CCCP to handle nonconvexity (note the interpretation of MI-SVM as an instance of CCCP is due to Cheung and Kwok [2006]). D-MimlSvm [Zhou et al. 2012] also uses CCCP to optimize Hamming loss.

In recent work, Li et al. [2012] consider the problem of identifying the "key instances" that trigger labels. This problem is similar to instance annotation, but differs in that the goal is not to label all instances, but instead to select a set of instances explaining each label.

7. CONCLUSION AND FUTURE WORK

In this work, we proposed rank-loss support instance machines for instance annotation. The goal of instance annotation is to learn an instance level-classifier using an MIML dataset for training data, which does not directly associate instances with labels. We explained why and empirically showed that rank-loss is superior to other loss functions e.g., Hamming or ambiguous loss for the instance annotation problem. The SIM algorithm is based on the observation that for the commonly used `max` model connecting bag-level labels and instance-level labels, the bag-level output can be represented as a linear function of a "support instance", which summarizes the instances in a bag. The SIM model can also be used with the `softmax` model, which is less sensitive to noise. The SIM model poses a nonconvex optimization problem; we offer a heuristic solution that is applicable to either `max` or `softmax` and a CCCP method that can be applied to the `max` model and guarantees monotonic decrease in the nonconvex objective. In either optimization method, the basic process is to alternate between updating support instances, and solving a convex problem to minimize rank loss. We give a primal subgradient descent algorithm for solving these convex problems with linear runtime in the number of bags and instances. We also demonstrate that an approximate kernel method often improves accuracy, while retaining linear runtime.

In future work we will examine deviations from the basic assumptions of MIML instance annotation. For example, the assumption that a bag label set is the union of instance labels, is often violated. This may be the case when the labeler does not provide a complete labeling of the data, and only labels a subset of relevant classes. Another interesting problem is recognizing when an instance does not belong to any of the known classes. This is a common issue in machine vision datasets, where there are often background scenery or novel objects that are not labeled. Finally, one should remember that when MIML instance annotation is applied, the classic SISL approach is also an option, and typically gives higher accuracy at the cost of increased effort to label instances. Learning from a mixed granularity dataset consisting of mostly bag-level label sets and a few unambiguously labeled instances is one potential way to achieve

the higher accuracy of SISL with the reduced labeling effort of MIML. However, this idea has received little study so far [Vijayanarasimhan and Grauman 2009].

APPENDIXES

APPENDIX 1. Proof of Bound on Subgradient

We follow the approach of Shalev-Shwartz and Singer [2007] to establish the convergence for sub-gradient descent. Because the number of iterations to reach an ϵsuboptimal solution is $O(\frac{L}{\lambda\epsilon})$, where L is a bound on the subgradient $||\mathbf{V}||^2 \leq L$, we proceed with a derivation of L. The bound proved in this appendix applies to subgradients (27) and (29). Suppressing all sub- and superscripts except for the class index q, recall that the complete subgradient is $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_q, \ldots, \mathbf{v}_c]$.

We will start with a bound on the qth component of $\mathbf{V}$; using the triangle inequality:

$$||\mathbf{v}_q|| \leq \lambda||\mathbf{w}_q|| + \sum_{ijk} \beta_i I[\cdot] \Big(||\hat{\mathbf{x}}|| I[q = k] + ||\hat{\mathbf{x}}|| I[q = j] \Big). \tag{39}$$

Note that $I[\cdot] \leq 1$ and for any support instance $||\hat{\mathbf{x}}|| \leq R$. Using these properties

$$||\mathbf{v}_q|| \leq \lambda||\mathbf{w}_q|| + R \sum_{ijk} \beta_i \Big(I[q = k] + I[q = j] \Big) \tag{40}$$

$$||\mathbf{v}_q|| \leq \lambda||\mathbf{w}_q|| + R\Big(\sum_{ijk} \beta_i I[q = k] + \sum_{ijk} \beta_i I[q = j] \Big). \tag{41}$$

The first sum on the RHS of (41) can be computed as

$$\sum_{ijk} \beta_i I[q = k] = \sum_{i=1}^{n} \beta_i \sum_{j \in Y_i} \sum_{k \notin Y_i} I[q = k] \tag{42}$$

$$= \sum_{i=1}^{n} \beta_i \sum_{j=1}^{c} \sum_{k=1}^{c} I[q = k]\, I[j \in Y_i]\, I[k \notin Y_i] \tag{43}$$

$$= \sum_{i=1}^{n} \beta_i I[q \notin Y_i] \underbrace{\sum_{j=1}^{c} I[j \in Y_i]}_{|Y_i|} = \sum_{i=1}^{n} \beta_i I[q \notin Y_i]\, |Y_i|. \tag{44}$$

Similarly, the second sum on the RHS of (41) can be simplified as

$$\sum_{ijk} \beta_i I[q = j] = \sum_{i=1}^{n} \beta_i I[q \in Y_i]\, |\bar{Y}_i| \tag{45}$$

Substituting these terms back into (41),

$$||\mathbf{v}_q|| \leq \lambda||\mathbf{w}_q|| + R\Big(\sum_{i=1}^{n} \beta_i I[q \notin Y_i]\, |Y_i| + \sum_{i=1}^{n} \beta_i I[q \in Y_i]\, |\bar{Y}_i| \Big). \tag{46}$$

To condense notation we will rewrite this statement as $||\mathbf{v}_q|| \leq a_q + b_q$, where

$$a_q = \lambda||\mathbf{w}_q|| \tag{47}$$

$$b_q = R \sum_{i=1}^{n} \beta_i \Big(I[q \notin Y_i]\, |Y_i| + I[q \in Y_i]\, |\bar{Y}_i| \Big). \tag{48}$$

LEMMA A.1. $\boxed{||\mathbf{V}||^2 \leq \left(\sqrt{\sum_{q=1}^{c} a_q^2} + \sqrt{\sum_{q=1}^{c} b_q^2}\right)^2}$

PROOF.

$$||\mathbf{v}_q||^2 \leq (a_q + b_q)^2 = a_q^2 + b_q^2 + 2a_q b_q \tag{49}$$

$$||\mathbf{V}||^2 = \sum_{q=1}^{c} ||\mathbf{v}_q||^2 \leq \sum_{q=1}^{c} a_q^2 + \sum_{q=1}^{c} b_q^2 + \sum_{q=1}^{c} 2a_q b_q \tag{50}$$

$$\leq \sum_{q=1}^{c} a_q^2 + \sum_{q=1}^{c} b_q^2 + 2\sqrt{\left(\sum_{q=1}^{c} a_q^2\right)\left(\sum_{q=1}^{c} b_q^2\right)} \text{ by Cauchy-Schwarz} \tag{51}$$

$$\leq \left(\sqrt{\sum_{q=1}^{c} a_q^2} + \sqrt{\sum_{q=1}^{c} b_q^2}\right)^2. \tag{52}$$

□

LEMMA A.2. $\boxed{\sum_{q=1}^{c} a_q^2 \leq 2\lambda}$

PROOF.

$\sum_{q=1}^{c} a_q^2 = \sum_{q=1}^{c} \lambda^2 ||\mathbf{w}_q||^2 = \lambda^2 \sum_{q=1}^{c} ||\mathbf{w}_q||^2 = \lambda^2 ||\mathbf{W}||^2 \leq 2\lambda$ because $\mathbf{W} \in S$. □

LEMMA A.3. $\boxed{\sum_{q=1}^{c} b_q^2 \leq 4R^2}$

PROOF.

$$\sum_{q=1}^{c} b_q^2 = R^2 \sum_{q=1}^{c} \left(\sum_{i=1}^{n} \beta_i (I[q \notin Y_i] \, |Y_i| + I[q \in Y_i] \, |\bar{Y}_i|)\right)^2 \tag{53}$$

$$= R^2 \sum_{q=1}^{c} \sum_{i=1}^{n} \sum_{i'=1}^{n} \beta_i \beta_{i'} \Big(I[q \notin Y_i] \, |Y_i| + I[q \in Y_i] \, |\bar{Y}_i|\Big)\Big(I[q \notin Y_{i'}] \, |Y_{i'}| + I[q \in Y_{i'}] \, |\bar{Y}_{i'}|\Big). \tag{54}$$

Note that this result is obtained from the identity $\Big(\sum_{i=1}^{n} x_i\Big)^2 = \sum_{i=1}^{n}\sum_{i'=1}^{n} x_i x_{i'}$. Expanding the right side of the expression, we will obtain four terms, e.g.,

$$R^2 \sum_{q=1}^{c}\sum_{i=1}^{n}\sum_{i'=1}^{n} \beta_i \beta_{i'} I[q \notin Y_i]\, I[q \notin Y_{i'}]\, |Y_i||Y_{i'}| \tag{55}$$

$$= R^2 \sum_{i=1}^{n}\sum_{i'=1}^{n} \beta_i \beta_{i'} |Y_i||Y_{i'}| \sum_{q=1}^{c} I[q \notin Y_i]\, I[q \notin Y_{i'}] \tag{56}$$

$$\leq R^2 \sum_{i=1}^{n}\sum_{i'=1}^{n} \beta_i \beta_{i'} |Y_i||Y_{i'}| \sqrt{\Big(\sum_{q=1}^{c} I[q \notin Y_i]^2\Big)\Big(\sum_{q=1}^{c} I[q \notin Y_{i'}]^2\Big)} \text{ by Cauchy-Schwarz} \tag{57}$$

$$= R^2 \sum_{i=1}^{n}\sum_{i'=1}^{n} \beta_i \beta_{i'} |Y_i||Y_{i'}| \sqrt{|\bar{Y}_i||\bar{Y}_{i'}|} \tag{58}$$

$$= R^2 \sum_{i=1}^{n}\sum_{i'=1}^{n} \frac{|Y_i||Y_{i'}|\sqrt{|\bar{Y}_i||\bar{Y}_{i'}|}}{n^2 |Y_i||\bar{Y}_i||Y_{i'}||\bar{Y}_{i'}|} = \frac{R^2}{n^2} \sum_{i=1}^{n}\sum_{i'=1}^{n} \frac{1}{\sqrt{|\bar{Y}_i||\bar{Y}_{i'}|}} = \frac{R^2}{n^2}\Big(\sum_{i=1}^{n} \frac{1}{\sqrt{|\bar{Y}_i|}}\Big)\Big(\sum_{i=1}^{n} \frac{1}{\sqrt{|\bar{Y}_i|}}\Big). \tag{59}$$

The other three terms can be derived similarly. It follows that

$$\sum_{q=1}^{c} b_q^2 \leq \frac{R^2}{n^2}\Big(\Big(\sum_{i=1}^{n} \frac{1}{\sqrt{|\bar{Y}_i|}}\Big)^2 + \Big(\sum_{i=1}^{n} \frac{1}{\sqrt{|Y_i|}}\Big)^2 + 2\Big(\sum_{i=1}^{n} \frac{1}{\sqrt{|\bar{Y}_i|}}\Big)\Big(\sum_{i=1}^{n} \frac{1}{\sqrt{|Y_i|}}\Big)\Big) \tag{60}$$

$$= \frac{R^2}{n^2}\Big(\sum_{i=1}^{n}\Big(\frac{1}{\sqrt{|Y_i|}} + \frac{1}{\sqrt{|\bar{Y}_i|}}\Big)\Big)^2 \leq R^2(1 + \frac{1}{\sqrt{c-1}})^2. \tag{61}$$

To obtain the last inequality, observe that a label set Y_i may not be empty or contain all c classes, therefore $1 \leq |Y_i| \leq c-1$ and $1 \leq |\bar{Y}_i| \leq c-1$. Finally, note that $1 + \frac{1}{\sqrt{c-1}} \leq 2$, therefore $\sum_{q=1}^{c} b_q^2 \leq 4R^2$. □

Combining Lemmas A.1, A.2, and A.3, we get

$$||\mathbf{V}||^2 \leq \Big(\sqrt{\sum_{q=1}^{c} a_q^2} + \sqrt{\sum_{q=1}^{c} b_q^2}\Big)^2 \tag{62}$$

$$\leq \Big(\sqrt{2\lambda} + 2R\Big)^2. \tag{63}$$

therefore $L' = \Big(\sqrt{2\lambda} + 2R\Big)^2$ is a suitable bound for the Pegasos analysis. This result applies to either SIM-CCCP or SIM-Heuristic with rank loss.

APPENDIX 2. Implementation Details

In this appendix, we further discuss the design and implementation of of SIM.

Average Support Instance Initialization. The rank loss objective is nonconvex and hence the optimum found may depend on the starting point. In preliminary experiments, we tried starting with random weights, then computing the first support instances with `max` or `softmax` based on the random weights. However, a problem with this approach is that the support instance $\hat{\mathbf{x}}_{ij}$ computed from random weights is unlikely to be a good representation of class j for bag i. This approach is prone to getting stuck in bad local optima. Also note that meaningful support instances cannot be computed from $\mathbf{W} = 0$. Our proposed SIM algorithms instead use an `average` model $\hat{\mathbf{x}}_{ij} = \frac{1}{n_i} \sum_{\mathbf{x} \in X_i} \mathbf{x}$, which does not depend on the weights. After solving the convex problem once with these `average` support instances, the weights are good enough to compute support instances with `max` or `softmax` in subsequent iterations.

Warm Start. Going from one outer iteration of CCCP or the heuristic to the next, we start at the weights with the lowest objective evaluation on the convex problem from the previous iteration. We do this because projected subgradient descent is not guaranteed to monotonically decrease the convex objective (an upper bound on the objective is guaranteed to decrease monotonically). Hence the best solution may occur in some iteration of subgradient descent other than the last.

Using an Approximate Solver within CCCP. To show that CCCP monotonically decreases the objective, it is assumed that the convex problem in each iteration is solved exactly [Sriperumbudur and Lanckriet 2009]. However, we are using an approximate solver in each iteration. We can still ensure monotonic decrease in the original DC problem by running a sufficient number of iterations of subgradient descent. Recall that the convex problem solved in each iteration of CCCP uses a linear upper bound of the concave part of the objective at the current point, which touches the original DC objective at the current point. Hence at iteration t of CCCP, we have $h_{RL}(\mathbf{W}^{(t)}) = h^{(t)}_{RL,cccp}(\mathbf{W}^{(t)})$. At iteration τ of subgradient descent within iteration t of CCCP, the objective value for the convex problem is $h^{(t)}_{RL,cccp}(\mathbf{W}^{(t,\tau)})$. If this objective is less than the DC objective at the start of the iteration of CCCP, i.e. $h^{(t)}_{RL,cccp}(\mathbf{W}^{(t,\tau)}) \leq h^{(t)}_{RL,cccp}(\mathbf{W}^{(t)})$, then τ is a sufficient number of iterations to ensure DC objective is monotonically decreasing, because

$$h_{RL}(\mathbf{W}^{(t+1)}) \leq h^{(t)}_{RL,cccp}(\mathbf{W}^{(t,\tau)}) \leq h^{(t)}_{RL,cccp}(\mathbf{W}^{(t)}) = h_{RL}(\mathbf{W}^{(t)}). \tag{64}$$

In our implementation of CCCP, we start by running K iterations of subgradient descent. If K iterations is enough to decrease the h_{RL} objective, we move on to the next iteration of CCCP. If not, we repeatedly run K more iterations until either the h_{RL} objective decreases, or the total number of iterations reaches K_{max}. In the latter case, we terminate CCCP and return the best solution found so far. We do not use this scheme of increasing the number of iterations of subgradient descent for the heuristic optimization method (it always uses exactly K iterations).

Feature Rescaling. We apply the following preprocessing to the instance features. First, we transform each feature to the range $[0, 1]$. Next, we apply the same feature rescaling process used in the *Convex Learning from Partial Labels Toolbox* (for ambiguous label classification [Cour et al. 2011]), which centers the data and scales each feature by $\frac{1}{\sqrt{\sum_{i=1}^{m} ||\mathbf{x}_i||^2}}$. When the Random Fourier Kernel features are used, we first apply the preceding process, then apply the transform $\mathbf{z}$, then repeat the process from

the CLPL a second time. We have observed that the proposed methods are sensitive to feature scaling, and found these processing steps effective.

Numerical Issues with `softmax`. Due to large exponents, numerical overflow may occur when computing the `softmax` weights as

$$\alpha_{iq}^{j} = \frac{e^{\mathbf{w}_j \cdot \mathbf{x}_{iq}}}{\sum_{\mathbf{x}' \in X_i} e^{\mathbf{w}_j \cdot \mathbf{x}'}}.$$

This problem is more likely to occur when the regularization parameter λ is small, because the weights are constrained to a ball with large radius. An equivalent formula for calculating the `softmax` weights that avoids numerical issues is

$$\alpha_{iq}^{j} = \frac{e^{\mathbf{w}_j \cdot \mathbf{x}_{iq} - b}}{\sum_{\mathbf{x}' \in X_i} e^{\mathbf{w}_j \cdot \mathbf{x}' - b}} \text{ where } b = \max_{\mathbf{x} \in X_i} \mathbf{w}_j \cdot \mathbf{x}. \tag{65}$$

ACKNOWLEDGMENTS

We would like to thank Matthew Betts, Sarah Frey, Adam Hadley, and Jay Sexsmith for their help in collecting HJA data, Iris Koski for labeling the data, Katie Wolf for her work on noise reduction, and Lawrence Neal for his work on segmentation.

REFERENCES

Andrews, S., Tsochantaridis, I., and Hofmann, T. 2002. Support vector machines for multiple-instance learning. In *Proceedings of Advances in Neural Information Processing Systems 15*, 561–568.

Briggs, F., Fern, X., and Raich, R. 2012a. Rank-loss support instance machines for MIML instance annotation. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* 534–542.

Briggs, F., Lakshminarayanan, B., Neal, L., Fern, X., Raich, R., Hadley, S., Hadley, A., and Betts, M. 2012b. Acoustic classification of multiple simultaneous bird species: A multi-instance multilabel approach. *J. Acoust. Soc. Amer. 131*, 4640.

Carroll, L. 1896. *Through the Looking-Glass: And What Alice Found There*. Macmillan UK.

Chang, C.-C. and Lin, C.-J. 2001. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol. 2*, 3.

Cheung, P. and Kwok, J. 2006. A regularization framework for multiple-instance learning. In *Proceedings of the 23rd International Conference on Machine Learning.* 193–200.

Cour, T., Sapp, B., Jordan, C., and Taskar, B. 2009. Learning from ambiguously labeled images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 919–926.

Cour, T., Sapp, B., and Taskar, B. 2011. Learning from partial labels. *J. Mach. Learn. Res. 12*, 1225–1261.

Dalal, N. and Triggs, B. 2005. Histograms of oriented gradients for human detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. Vol. 1.* 886–893.

Demšar, J. 2006. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res. 7*, 1–30.

Dietterich, T., Lathrop, R., and Lozano-Pérez, T. 1997. Solving the multiple instance problem with axis-parallel rectangles. *Artif. Intell. 89*, 1–2, 31–71.

Du, L., Ren, L., Dunson, D., and Carin, L. 2009. A Bayesian model for simultaneous image clustering, annotation and object segmentation. In *Proceedings of Advances in Neural Information Processing Systems 22*, 486–494.

Elisseeff, A. and Weston, J. 2001. A kernel method for multi-labelled classification. In *Proceedings of Advances in Neural Information Processing Systems 14*, 681–687.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. 2010. The Pascal visual object classes (VOC) challenge. *Intern. J. Comput. Vis. 88*, 2, 303–338.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. 2008. LIBLINEAR: A library for large linear classification. *J. Mach. Learn. Res. 9,* 1871–1874.

Frey, P. W. and Slate, D. J. 1991. Letter recognition using Holland-style adaptive classifiers. *Mach. Learn. 6*, 161.

Frost, R. 1916. *Mountain Interval*. Henry Holt and Company.

Hüllermeier, E. and Beringer, J. 2006. Learning from ambiguously labeled examples. *Intell. Data Anal. 10*, 5, 419–439.

Li, Y., Ji, S., Kumar, S., Ye, J., and Zhou, Z., 2009. Drosophila gene expression pattern annotation through multi-instance multi-label learning. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*. 1445–1450.

Li, Y., Hu, J., Jiang, Y., and Zhou, Z. 2012. Towards discovering what patterns trigger what labels. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*.

Nguyen, N. 2010. A new SVM approach to multi-instance multi-label learning. In *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM)*. 384–392.

Rahimi, A. and Recht, B. 2007. Random features for large-scale kernel machines. In *Proceedings of Advances in Neural Information Processing Systems 20*. 1177–1184.

Shalev-Shwartz, S. and Singer, Y. 2007. Logarithmic regret algorithms for strongly convex repeated games. Tech. rep., The Hebrew University.

Shalev-Shwartz, S., Singer, Y., and Srebro, N. 2007. Pegasos: Primal estimated sub-gradient solver for SVM. In *Proceedings of the 24th International Conference on Machine Learning*. 807–814.

Shalev-Shwartz, S., Singer, Y., Srebro, N., and Cotter, A. 2011. Pegasos: Primal estimated sub-gradient solver for SVM. *Math. Prog. 127*, 1, 3–30.

Shen, C., Jiao, J., Wang, B., and Yang, Y. 2009. Multi-instance multi-label learning for automatic tag recommendation. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*.

Sriperumbudur, B. and Lanckriet, G. 2009. On the convergence of the concave-convex procedure. In *Proceedings of Advances in Neural Information Processing Systems 22*, 1759–1767.

Vezhnevets, A., Buhmann, J., and Zurich, E. 2010. Towards weakly supervised semantic segmentation by means of multiple instance and multitask learning. In *Proceedings of Conference on Computer Vision and Pattern Recognition*.

Vijayanarasimhan, S. and Grauman, K. 2009. What's it going to cost you? Predicting effort vs. informativeness for multi-label image annotations. In *Proceedings of CPVR*.

Wang, W. and Zhou, Z. 2012. Learnability of multi-instance multi-label learning. *Chinese Sci. Bull.*, 1–4.

Winn, J., Criminisi, A., and Minka, T. 2005. Object categorization by learned universal visual dictionary. In *Proceedings of the 10th IEEE International Conference on Computer Vision (ICCV)*. 1800–1807.

Xu, X., Xue, X., and Zhou, Z. 2011. Ensemble multi-instance multi-label learning approach for video annotation task. In *Proceedings of the 19th ACM International Conference on Multimedia*. 1153–1156.

Yakhnenko, O. 2009. Learning from text and images: Generative and discriminative models for partially labeled data. Ph.D. thesis, Iowa State University.

Yang, S., Zha, H., and Hu, B. 2009. Dirichlet-Bernoulli alignment: A generative model for multi-class multi-label multi-instance corpora. In *Proceedings of Neural Information Processing Systems*. 2143–2150.

Yang, S., Bian, J., and Zha, H. 2010. Hybrid generative/discriminative learning for automatic image annotation. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

Yuille, A. and Rangarajan, A. 2002. The concave-convex procedure (CCCP). In *Proceedings of Advances in Neural Information Processing Systems 2*, 1033–1040.

Zha, Z., Hua, X., Mei, T., Wang, J., Qi, G., and Wang, Z. 2008. Joint multi-label multi-instance learning for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1–8.

Zhang, M. and Zhou, Z. 2008. M3MIML: A maximum margin method for multi-instance multi-label learning. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM)*. 688–697.

Zhou, Z. 2004. Multi-instance learning: A survey. Tech. rep., AI Lab, Department of Computer Science and Technology, Nanjing University.

Zhou, Z. and Zhang, M. 2007. Multi-instance multi-label learning with application to scene classification. In *Proceedings of Advances in Neural Information Processing Systems*.

Zhou, Z., Zhang, M., Huang, S., and Li, Y. 2012. Multi-instance multi-label learning. *Artif. Intell. 176*, 1, 2291–2320.

Received September 2012; revised December 2012; accepted March 2013