AN ABSTRACT OF THE THESIS OF

Forrest Briggs for the degree of Doctor of Philosophy in Computer Science presented on November 25, 2013.

Title: Multi-Instance Multi-Label Learning: Algorithms and Applications to Bird Bioacoustics

Abstract approved: _____

Xiaoli Z. Fern

We consider the problem of supervised classification of bird species from audio recordings in a real-world acoustic monitoring scenario (i.e. audio data is collected in the field with an omnidirectional microphone, without human supervision). Obtaining better data about bird activity can assist conservation efforts, and improve our understanding of their interactions with the environment and other organisms. However, traditional observation methods are labor-intensive. Most prior work on machine learning for bird song is not applicable to real-world acoustic monitoring, because it assumes recordings contain only a single species of bird, while recordings typically contain multiple simultaneously vocalizing birds. We propose to use the multi-instance multi-label (MIML) framework in machine learning for the species classification problem, where the dataset is viewed as a collection of bags of instances paired with sets of labels. Furthermore, we formalize MIML instance annotation, where the goal is to predict instance labels while learning only from bag label sets. We develop the first MIML representation for audio, and several new algorithms for MIML instance annotation based on support vector machines or classifier chains. The proposed methods classify either the set of species present in a recording, or individual calls, while learning only from recordings paired with a set of species. This form of training data requires less human effort to obtain than individually labeled calls. These methods are successfully applied to audio collected in the field which included multiple simultaneously vocalizing species. The proposed algorithms for MIML classification are general, and are also applied to object recognition in images.

Multi-Instance Multi-Label Learning: Algorithms and Applications to Bird Bioacoustics

by

Forrest Briggs

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented November 25, 2013
Commencement June 2014

Doctor of Philosophy thesis of Forrest Briggs presented on November 25, 2013.

APPROVED:

Major Professor, representing Computer Science

Director of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Forrest Briggs, Author

ACKNOWLEDGEMENTS

This work is partially funded by the Ecosystems Informatics IGERT program via NSF grant DGE 0333257, NSF-CDI grant 0941748 to M.G.B., NSF grant 1055113 to X.F., and the College of Engineering, Oregon State University. We would also 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, Lawrence Neal for his work on segmentation, and Dave Mellinger for his help editing.

I would also like to thank my advisors Xiaoli Z. Fern, Raviv Raich, and Matthew Betts for their guidance in this research, and my mom, Luanna Helfman for her support.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

Figure Page

LIST OF TABLES

Chapter 1: Introduction

This thesis addresses two main ideas: multi-instance and/or multi-label supervised classification, and applications to bird bioacoustics.

Automatic recognition of bird species from audio is an important problem. Habitat loss, declining biodiversity [6], and climate change [116] necessitate the development of better tools to monitor birds, including their ranges, diversity, and phenology. Birds are a good indicator of ecosystem health and diversity because they are relatively easy to detect, may provide information about other organisms such as plants and insects, and respond quickly to environmental change [125]. However, monitoring bird populations and activity is a labor intensive task. Traditional methods involve human experts working in the field. High labor requirements limit the spatial and temporal resolution, and extent of bird data collection [123]. Acoustic population surveys instead use microphones to record bird vocalizations, which can later be analyzed by humans or signal processing and machine learning tools to estimate species presence/absence, abundance, gender, age, and other individual characteristics. Acoustic surveys can provide greater resolution and extent than human observers, and are more practical in remote sites (where limitations for human observers are particularly acute [10]). For example, it is impractical for a human to record observations on the side of a cliff continuously for two weeks, but a microphone can collect audio in the same scenario.

Over the last 5 years, our collaborators have collected more than 10 terabytes of audio recordings of birds using unattended omnidirectional microphones in the the H. J. Andrews Experimental (HJA) Forest in the Cascade Mountains of Oregon (this corresponds to thousands of hours of audio collected at 13 sites). Although there has been over a decade of research on machine learning for bird song [88], most work has focussed on ideal problem setups such a single bird with a person aiming a directional microphone at it to obtain a cleaner sound. In contrast, the audio data collected in HJA is less ideal because unattended omnidirectional microphones pick up all sounds in the environment, particularly wind and stream noise, and there are often several birds vocalizing at once. Prior machine learning and signal processing methods are not sufficient for the HJA audio dataset because they do not handle multiple simultaneously vocalizing birds and strong background noise environments. This thesis contributes the first

steps toward automated machine learning-based analysis of this dataset to infer species activity at the resolution of either 10-second intervals, or individual utterances. We present solutions for noise and multiple simultaneously vocalizing birds of different species.

The most common formulation of supervised classification pairs instances (feature vectors) with single labels, hence it is called single-instance single-label learning (SISL). Well known algorithms such as support vector machines (SVM) and logistic regression are SISL methods. Reducing classification problems with audio and images to SISL often involves a transformation which discards useful structure in the data. Many supervised learning problems in audio and vision domains have a certain structure where documents are naturally decomposed into a collection of parts. For example, an image is naturally represented as a collection of regions. Similarly, an audio recording can be decomposed into a collection of parts in several ways. This structure motivates multiple instance learning (MIL) [49], which departs from basic supervised classification in that the representation is no longer a simple vector. It is often natural to associate a document with multiple labels, for example to indicate all objects present in an image, or all bird species present in an audio recording. Multi-label classification (MLC) addresses this kind of classification problem. Multi-instance multi-label (MIML) classification [172] combines both ideas.

Chapter 2 reviews literature on machine learning for bird sound. In each subsequent chapter, we present a new learning algorithm which focuses on different aspects of the multi-instance and/or multi-label structure in the problem of recognizing bird species from audio. In some cases, we are also able to apply the proposed learning methods to other domains, e.g., object recognition in images.

In chapter 3, we consider the problem of acoustic classification of bird species from a short audio recording of its vocalizations. It is assumed that there is only a single species of bird in each recording. Recordings are represented as a collection of frame-level (i.e. short-duration) acoustic features, and summarized by a codebook histogram. The approach proposed in this chapter can be consider a multi-instance, single-label, multi-class method. We formulate a probability model for sounds in a short recording, and show that the Bayesian maximum a posteriori (MAP) classifier is approximated by a nearest-neighbor classifier using Kullback Leibler divergence between frame-level feature histograms. In this early work, we conduct experiments on audio recordings from the Cornell Macaulay library, which are single-species recordings collected with a directional microphone.

In chapter 4, we propose the first application of MIML to audio. MIML formulations have

previously been used in text [132], image [97, 165], and video [157] domains. In order to handle multiple simultaneously vocalizing birds in the HJA audio data, we formulate the problem of species presence/absence prediction in the MIML framework. A short (e.g. 10 second) audio recording is a bag, the instances are feature vectors describing distinct utterances, and the label set is a list of the species that are audible in the recording (as determined by a human expert). The MIML classifier outputs a set of predicted species, given a new audio recording as input. In this chapter, we use off-the-shelf MIML classifiers; the main contribution is a procedure for transforming an audio signal into a bag-of-instances representation. Our proposed method filters background noise from streams and wind, and can separate distinct bird utterances that overlap in time (but not frequency). This work is one of the first to address bird sound classification with multiple vocalizations that overlap in time.

In chapter 5, we consider a different variation of MIML. In most prior work on MIML, the goal is to predict the label set for a previously unseen bag. A related problem which has received less attention is training a classifier that predicts instance labels on a MIML dataset, which provides only bag-level label sets. This problem is called MIML instance annotation. In this setup, no instance labels are available for learning; bag label sets indirectly provide ambiguous information about instance labels. The advantage of MIML instance annotation compared to SISL is that bag label sets are often cheaper to obtain than instance labels. For example, it is more time consuming to annotate each individual pixel in an image with its class than to provide a set of tags describing the contents of the image. Similarly, it is less time-consuming for a human to label audio recordings of bird song with a set of audible species than to individually label each utterance. Applying MIML instance annotation to bird song, the training data consists of short audio recordings paired with a list of species, and the predictions made by the classifier are species labels for each individual utterance. Hence the MIML instance annotation approach enables a finer temporal resolution analysis of the bird song data with a relatively low labeling cost. In a MIML instance annotation image problem, the goal is to predict the label for regions in an image while training on a dataset consisting of images paired with lists of objects. Images are routinely tagged in online databases, while datasets with pixel-level labels are rare and generally only constructed to facilitate machine vision experiments.

We formalize the MIML instance annotation problem, and propose a new SVM-style algorithm. A common pattern is to train one linear model for each class and predict the instance label as the highest-scoring class. Most prior MIML algorithms are designed to predict label sets for new bags. Some MIML classifiers designed for label-set prediction also build an instance-level

model as a side effect of training on a MIML dataset, however such algorithms have not been explicitly designed to optimize instance-level classification accuracy. For example, M^3MIML [168] is an SVM-style algorithm for MIML classification which builds an instance-level classifier by minimizing a regularized bag-level loss function. While the loss function used in M^3MIML (Hamming loss) is good for label set prediction, it is not good for instance-level classification because it fails to calibrate the models for each class in a way that causes the correct class have the highest-ranked score. We propose a regularized bag-level rank-loss objective, which is designed to improve instance classification accuracy by ranking the class scores in the correct order. A bag-level loss function must be used because only bag label sets are available for training, however the goal is to learn an instance-level classifier. Hence it is necessary to make some assumptions about the relationship between bag label sets and instance labels which allow the loss to be expressed in terms of the instance-level models.

A common assumption is that the bag label set is the union of instance labels. Optimizing regularized bag-level loss functions for MIL and/or MIML is non-convex under a standard approximation of this assumption. A practical problem for applying SVM-style algorithms to MIL and MIML is how to optimize the non-convex problem. We propose heuristic and constrained convex-concave procedure–based methods for MIML rank-loss optimization. Both methods alternate between summarizing each bag with a single instance that is most representative of each class (called a "support instance"), and solving a convex problem by efficient primal sub-gradient descent. Hence we call the proposed methods rank-loss Support Instance Machines (SIM). Experiments on the HJA birdsong data, as well as several machine vision and artificial problems show that rank-loss SIM achieves higher accuracy than similar algorithms using different loss functions.

The rank-loss SIM algorithms can be applied in either transductive or inductive modes. In the transductive mode, the goal is to classify instances for bags with known label sets, whereas in the inductive mode, the goal is to classify instances for new bags, which have unknown label sets. Rank-loss SIM, and several other recent algorithms to which it is compared, all achieve lower accuracy in the inductive mode. This outcome is to be expected because less information is given to the classifier in the inductive mode. Rank-loss SIM classifies each instance in a bag independently from the other instances, and thus it does not take advantage of contextual information provided by other instances in the same bag. In chapter 6, we develop a new algorithm for MIML instance annotation, which achieves better accuracy in the inductive mode by jointly classifying all instances in a bag, in a way that exploits the contextual relationships

between correlated labels. For example, if an image contains cows, it is also likely to contain grass. Similarly, the presence of bird species with similar ecological niches is often correlated.

The proposed method in chapter 6 extends an algorithm for MLC called ensembles of classifier chains (ECC) [122], to MIML instance-annotation, hence we call it MIML-ECC. Classifier chains exploit label correlation by making a binary decision about each class sequentially, and using all previous classes as features that provide context to inform the next prediction. A standard classifier chain for MLC is a chain of SISL classifiers; we extend classifier chains to MIML instance annotation by building a chain of MIL classifiers. MIML-ECC computes probabilities for each instance to belong to each class, and uses these probabilities to form a label set for the whole bag, which in turn provides context for computing more instance-level probabilities. This may seem circular, but it is not because each class is processed sequentially in a chain. We propose a training algorithm that is closely related to EM, and a classification procedure that approximates posterior inference by sampling.

The MIML-ECC algorithm improves on rank-loss SIM (and other SVM-style algorithms that can be applied to instance annotation) in several additional ways. In particular, it is not possible to use cross-validation to select kernel and regularization parameters in MIML instance annotation, because the instance labels are not available to compute accuracy in the held-out folds. Consequently, rank-loss SIM must rely on heuristics to select these parameters. In contrast, MIML-ECC does not have any parameters that need to be tuned (it only has parameters that control a trade-off between runtime and approximation accuracy, which can simply be set as high as time allows). Furthermore, while the training time for rank-loss SIM is linear in the number of bags and instances, it is quadratic in the number of classes. MIML-ECC is scalable in all dimensions of the problem.

The new algorithms proposed in this thesis provide a practical solution to many problems in automatic acoustic monitoring of birds. The MIML representation is a natural fit for audio recordings of bird sounds, and enables classification of multiple simultaneously vocalizing birds. MIML instance annotation enables fine-grained analysis of utterances, while making efficient use of human labeling effort by learning from species label sets. This formulation is a general machine learning framework, useful not only for bird sound, but also in other domains such as images.

Chapter 2: Background & Literature Review

In this chapter, we discuss prior work on machine learning for acoustic classification of bird species. In order to provide a more comprehensive literature review, this chapter also discusses some work that was published after the work presented in later chapters of this thesis. Each chapter includes a self-contained related works section. Refer to these sections for more information about prior work in multi-instance and/or multi-label learning. Other surveys on machine learning for classification and detection of birds are available, e.g., [71, 124]. Here we provide basic background, and argue that prior work on machine learning for bird song does not address several major challenges, and hence the advances in this thesis are necessary and significant.

2.1 Spectrograms

Sound is a wave of pressure. Audio data is stored on a computer as a quantized signal, which is a sequence of numbers that represent the pressure as a function of time. The signal has a particular sampling frequency, which is the number of samples per second. Some methods for sound analysis can be applied to the signal directly, but it is often easier to identify patterns through a spectrogram representation. Spectrograms are one of the most fundamental tools for analysis of bird sound [25]. A spectrogram is a plot with time on the horizontal axis, frequency on the vertical axis, and brightness representing intensity, energy, or loudness. Spectrograms can also be through of as a digital image. More formally, spectrograms are computed by dividing a signal into overlapping *frames*, each of which consists of a block of consecutive samples of the signal. Then a Fourier transform is applied to each frame, which produces a vector of complex Fourier coefficients. Each column of pixels in a spectrogram corresponds to the magnitudes of the coefficients for one frame.

2.2 Common Design Patterns

There are several design patterns that are common to many systems for bird species classification and detection. We will cite specific examples of these patterns in the following sections. One of the most prevalent is to divide the problem into three stages: segmentation, featurization, and classification. In this pattern, first the signal or spectrogram is devided into distinct syllables, calls, or segments, then each is associated with a feature vector. Finally, a supervised classification algorithm is applied to the features. Other variations of this pattern includes associating a whole recording with a feature vector (rather than individual parts), and classifying this feature. In some cases, predictions are made about frames or syllables, while in other cases, predictions are made about a song consisting of multiple frames or syllables (sometimes directly, and other times, by aggregating lower level predictions).

2.3 Segmentation

Many bird sounds have structure at multiple levels of organization, referred to as notes, calls, syllables, and songs. Often, the distinction between these different levels of structure is ambiguous [136], and not all levels are relevant to a given recording or species. From a practical perspective, most systems for audio classification divide a recording into units referred to as syllables or segments. These units are simply whatever is isolated by the segmentation algorithm. Detection is related to segmentation, but instead the goal is to identify specific events (rather than to divide a recording into distinct units).

2.3.1 Energy Threshold

The most commonly used methods for segmentation are based on energy (which is related to loudness), and simply look for intervals of time or 2D regions in a spectrogram where energy is high. Such methods often begin with the computation of a smoothed energy envelope, which is energy as a function of time, convolved with some kernel to smooth out small deviations. Another common pattern is to estimate the energy level of background noise based on intervals outside of segments, and iteratively refine the segmentation.

For example, Härmä and Somervuo [74] start with a smoothed energy envelope, and computes the global maximum energy M_{dB}. A threshold $T_{dB} = M_{dB} - 20\ dB$ is set, and syllables are identified as intervals where the energy is above this threshold. Next, a noise energy level

N_{dB} is estimated from intervals which are not in a syllable. Then the threshold is updated to a new value which depends on the noise level based on the rule $T_{dB} = (M_{dB} - N_{dB})/2$, and the process is iterated to refine the segmentation.

Similarly, in [31, 59, 60, 136], first the noise level N_{dB} is estimated as the global minimum of the smoothed energy envelope. Then a threshold $T_{dB} = N_{dB}/2$ is set (note that division by 2 indicates a louder sound, because the decibel measure is negative). Syllables are identified as intervals above the threshold, and the noise level is recomputed from intervals below the threshold. The threshold is recomputed by the same rule above, and the process is iterated.

Some other works use energy-based segmentation, but provide less details on the method. Fagerlund and Härmä [58] used an energy threshold method with adaptive noise level estimation, similar to the above methods. Selin et al. [126] used an energy threshold that changes over time based on a moving average. In [30], the main goal is to extract spectral peaks, but energy-based segmentation is used as a preprocessing step. In this work, the background noise level is estimated based on a short period of time at the beginning of recordings known to not contain vocalizations. Similarly, Cheng et al. [32] assumes the first part of each recording is noise, and uses an energy threshold equal to a constant multiple of that energy level. Wielgat et al. [152] used an energy threshold combined with manual correction. Tyagi et al. [143] used segmentation based on a combination of energy and zero-crossing rate (ZCR). Vaca-Castano et al. [146] used energy threshold with a minimum and maximum syllable duration to merge short segments that are close in time.

2.3.2 Spectral Peak Tracking

Spectral peak tracking is a method for segmentation which is often used for tonal sounds (i.e. sounds with low bandwidth, which look like narrow lines in a spectrogram) [73], although it is sometimes used for other kinds of sound [92].

Härmä [73], Somervuo and Härmä [134, 135], and Lee et al. [92] use the following spectral peak tracking algorithm: First, the point in the spectrogram which is the time-frequency maxima is identified as $(t_0, f_0) = \arg\max_{(t,f)} S(t, f)$, where $S(t, f)$ is the spectrogram. Next, a start time t_s and end time t_e for the syllable containing the maxima is found by expanding from the maxima and stopping when the peak amplitude (the largest value of $S(t, f)$ over all f at a particular t) is a constant D dB less than at t_0. All elements of the spectrogram between t_s and t_e are then set to 0 to remove the syllable from the spectrogram, and the process is

repeated to find the next loudest syllable.

Chou et al. [36] proposed an algorithm very similar to [73] (maybe identical, although there is no citation in either direction). Their algorithm is an extension of work by Rabiner and Sambur [118] on segmentation for human speech, which starts at an energy maxima, then searches forward and backward for a start/end time based on energy and ZCR. Presumably, the same algorithm is used in [35].

Heller and Pinezich [75] seeks to extract multiple spectral peak tracks (as one might encounter in a harmonic vocalization). This method finds all local maxima in each frame, then connects each maxima with its nearest frame-adjacent neighbor.

2.3.3 Non-Segmentation and/or High-Energy Methods

Some methods do not aim for a precise segmentation of syllable time-boundaries, but instead use high-energy frames in another way. For example, Juang and Chen [84] find the max-energy frame, then use a fixed number of frames before and after as input to a neural net.

Briggs et al. [15] represented short audio recordings as a histogram of frame codewords (chapter 3). The set of frames used to construct the histogram is not required to be an exact segmentation into syllables. Instead, the 10% highest energy frames are used. Similarly, Stowell and Plumbley [138] aims to compare recordings of different lengths with different numbers of vocalizations without segmentation by constructing a histogram of frame-level features (using all frames).

2.3.4 Frequency-Band Methods

One common approach targeting specific species with known frequency ranges is to limit detection to sound in that range (referred to as a band). For example, Farnsworth and Russell [62] and Farnsworth et al. [63] detect where amplitude exceeds a threshold within a specified frequency band. Damoulas et al. [43] used the same detection method as [62]. In some systems that use frequency-band specific detection, subsequent classification is not applied because it is assumed that any detection within that band corresponds to the species of interest.

Bardeli et al. [8] seeks to detect calls from two specific species, in high-noise field recordings containing other animals, human speech, traffic and wind. They proposed novelty functions which measure the interest of a particular time over a manually specified range of frequencies.

Repeated calls are detected based on autocorrelation of the novelty functions, e.g., using Fourier analysis to determine if a repetition at a known characteristic period is present.

The main difficulty with frequency-band detection methods is that prior knowledge of the frequency range for the species of interest is required.

2.3.5 2D Segmentation

Most segmentation algorithms attempt to find the start and end time of a syllable, which can be viewed as a 1D approach. Alternatively, it is possible to identify a 2D region or bounding box in the spectrogram corresponding to a syllable. The advantage of 2D segmentation is that it may be able to separate sounds which overlap in time. Most of the methods presented in this thesis use 2D segmentation.

Brandes [13] seeks to classify sounds by species (for birds, frogs, and crickets). In this work, the focus is on frequency modulated, narrow-bandwidth calls. The frequency range is partitioned into separate bands automatically based on optima in an average spectrum. Then 2D per-pixel segmentation is performed separately within each band using an energy threshold set by a noise-estimate from that band. This method can obtain segments that overlap in time. ARBIMON [2] also used 2D energy-based segmentation (without dividing the spectrum into bands).

Duan et al. [53] proposed several algorithms for detecting common general patterns in spectrograms that describe a wide range of bird sounds. In particular, the categories of pattern are lines, blocks, warbles, stacked harmonics, and oscillations. This approach also generates 2D annotation.

2.3.6 Other Segmentation Methods

There are many other methods (primarily for 1D) segmentation which do not fit cleanly into one of the above categories.

McIlraith and Card [111] used a “leaky-integrator” for segmentation, which acts directly on the signal samples rather than the energy envelope. At each sample, if the absolute magnitude exceeds a threshold, the integrator is incremented by a constant, otherwise it is decremented by a different constant.

Du and Troyer [52] proposed a method for segmentation at the note-level rather than syllable level (for Zebra Finch song). It starts with a RMS amplitude envelope (similar to energy envelope). At each point on the time axis, a least-squares linear fit on the envelope is performed on both sides, using adjacent points. The angle between these two lines is treated as a second envelope which is compared to an energy-modulated threshold to identify note boundaries.

Lakshminarayanan et al. [90] proposed a segmentation method for identifying syllable boundaries. Instead of an energy envelope, each frame is treated as a probability density function, and the KL divergence between that distribution and a uniform distribution is used as an interest function. Segment boundaries are predicted at frames corresponding to local minima in KL divergence. Then for each interval between boundaries, a decision is made whether it is bird sound or background noise based on an iterative energy threshold algorithm similar to [60].

Jancovic and Köküer [83] proposed a heuristic "sine-distance" interest function, and predict frames where this distance exceeds a threshold as tonal bird sound.

Vaca-Castaño and Rodriguez [145] proposed a finite state machine that scans over frames, and updates its state based on the energy envelope, and an adaptive threshold. The purpose of the state machine is to handle syllables that may contain small gaps (i.e. some frames in the middle of a syllable may be below a simple threshold).

Ranjard and Ross [120] segment syllables by thresholding an autocorrelation function. Then syllable time boundaries are refined as the minima of a windowed frequency roll-off function. Segments below a fixed duration are dropped.

2.3.7 Manual Segmentation

In some experiments, syllables are segmented manually, or automatic segmentation is used first, then manually corrected. This approach is more commonly used when segmentation is not the primary focus of the study, and the authors wish to focus on a different part of the system.

Kogan and Margoliash [88] and Trifa et al. [142] use manually segmented templates for training an HMM. Lee et al. [91] proposed a new set of features for classification of species from syllables, and used manually segmented syllables. Acevedo et al. [1] use manually segmented syllables for supervised species classification experiments. Syllables are designated by a 2D bounding box in a spectrogram. Wimmer et al. [153] developed a web-based system for collaborative analysis of bioacoustic data. One part of this system is a labeling tool that allows

users to draw a rectangle around a vocalization in a spectrogram. This system can also be used in conjunction with automatic or semi-automatic segmentation. Other works in which sound is manually segmented into syllables, calls, or "phrases" include [70, 100, 101, 127, 140].

2.4 Acoustic Features for Bird Sound

Classifiers for bird sound can target frames, syllables, or whole recordings/songs. Features to describe sound exist for all of these levels of structure. In some cases, features are computed at one level, then aggregated to form a feature at higher level (i.e. the level where classification is desired).

2.4.1 Frame-Level Features

Frame-level features describe individual frames of a signal, and are often derived from the Fourier coefficients of the frame. The simplest frame-level feature is simply the vector of Fourier coefficient magnitudes. Works using this feature include [15, 89].

Mel-frequency cepstral coefficients (MFCCs) [45] are a frame-level feature originally developed for speech analysis. MFCCs are computed by transforming the spectrum of a frame into the Mel-frequency scale [150], which matches human perception of pitch more closely than linear frequency. This transformation is accomplished by applying a collection of non-uniformly spaced triangle filters. Each triangle filter response gives one Mel-frequency coefficient (MFC). MFCCs are the result of applying the discrete cosine transform (DCT) to the log of the Mel-frequency coefficients (the term 'cepstral' means spectrum of a spectrum). The DCT reduces the dimensionality of the Mel-scale spectrum, and removes correlation from its elements. There are many variations on the implementation details of computing MFCCs, e.g., the number of triangle filters used; some evaluation of different variants has been performed in [68].

MFCCs are the most widely used feature in machine learning for bird sound [9, 23, 30, 35, 36, 60, 88, 92, 136]. However, we argue that the Mel-frequency transform is not well motivated for bird sound. The Mel-frequency transform is appropriate for human speech, because it compresses the high-frequency portion of a spectrum more than the low frequency portion. Human speech is generally concentrated at low frequencies. However, many bird species make high-frequency sounds, which are degenerated by the Mel-frequency transformation. Experimental results presented in chapter 3 support this perspective. MFCCs have also been criticized

as being non-robust to noise [144], although many enchanted versions have been proposed to address this issue.

Other widely-used frame-level features include linear prediction coefficients (LPCs), and linear prediction cepstral coefficients (LPCCs), which were also developed for speech. LPC or LPCC are used in [30, 84, 88, 92, 111].

Many frame-level features are high-dimensional, hence some authors apply dimension-reduction methods such as principal component analysis (PCA) [89, 111].

2.4.2 Syllable-Level Features

One approach to constructing syllable-level features is to compute frame-level features for each frame within the syllable, then take the average feature vector [60, 92, 136]. Spectral peak tracking is used not only as a segmentation method, but also by representing the peak trajectory with a feature vector [73, 74, 136]. Other features include analysis-by-synthesis/overlap-add (ABS/OLA) [30], wavelets [9, 126], and 'descriptive parameters' such as bandwidth, zero-crossing rate and spectral flux [60, 136]. For marine mammals, Mellinger and Bradbury [112] proposed a set of noise-robust features that are computed from a 2D bounding box around a vocalization. These features are based on statistics of the time and frequency profiles within the box.

2.4.3 Recording/Song-Level Features

There are many ways to represent a recording by viewing it as a collection of parts, and summarizing the parts with a feature vector. One pattern for constructing such a feature vector is to use a histogram of frame-level features [15, 138]. Similarly, histograms of syllable types or syllable features may be used [19, 20, 134]. Simple statistics such as the number of segments or mean of segment features may also be used [20]. Other representations of the collection of syllables in a recording have been proposed, e.g., the vector of Hausdorff distances between the syllable features, and a dictionary of syllable clusters [20]. A spectrogram can also be represented as a collection of rectangular patches which are reconstructed as a sparse linear combination of dictionary atoms [20, 93].

A different kind of recording-level representation does not involve a collection of parts, but instead uses (statistical) signal descriptors, sometimes within a collection of frequency bands.

For example, the mean, variance, skewness, kurtosis, min, max and median statistics may be computed within Bark bands [20] (the `rp_extract` package [98] computes these features).

Two of the top-ten ranking teams in the MLSP 2013 Bird Classification Challenge [20] used template matching as a way of producing feature vectors at the recording level. First, a collection of templates were extracted automatically based on segmentation of training data into syllables. Then each template was compared with a target recording while sliding across in time. The value of the normalized cross-correlation at the time where it was maximized was used as a feature (one for each template).

While it is more common to average frame-level features for the purposes of representing syllables, this approach is also used to represent whole recordings in [143], and as a baseline method in [15].

2.5 Classifiers

Supervised classifiers that have been applied to bird sound in prior work fit into the SISL framework (i.e. they associate a single feature vector with a single label). SISL classifiers have been applied at the frame, syllable, and song/whole-recording level. SISL classifiers that have been used include k-nearest neighbors and other distance-based classifiers [9, 15, 73, 92, 100, 136], Naïve Bayes [100], multilayer perceptrons [9, 23, 35, 36, 100, 111, 126], deep-belief nets [20], recurrent neural nets [84], time-delay neural nets [127], support vector machines (SVM) [15, 43, 59, 60, 100], decision trees [60, 100], and Gaussian mixture models (GMM) [89, 136].

Hidden Markov-models (HMM), while not technically part of the SISL framework, still produce an output which is a sequence of single labels. HMMs are used in [9, 13, 30, 88, 136, 142].

This thesis focusses on multi-instance single-label and multi-instance multi-label classification. In recent work, we have also applied the multi-label classification framework (single-instance) to bird sound [19, 20].

Template matching can be applied as a form of classification/detection. One first obtains a collection of templates (often rectangular regions from a spectrogram), then slides them across a target spectrogram, while computing some measure of similarity such as cross-correlation. Points in time where the similarity is high are considered detections. Template matching is sometimes used in conjunction with dynamic time warping (DTW), which non-uniformly stretches either the template or the target spectrogram in time to produce a better match. The software package XBAT has been used extensively for template detection [28, 139].

One of the earliest works using template matching and DTW for bird song is [3]. Anderson et al. [3] manually extracted a set of templates, including some which represent intervals with no bird sound. The one-stage DTW algorithm is used to generate a representation of the target spectrogram as sequence of templates. This can be viewed as simultaneous segmentation and classification. Similarly, Chu and Blumstein [37] used DTW on frame-level features to obtain a distance measure for hierarchical clustering. The clustering is used to obtain a set of training templates (with manually annotated time boundaries). Hidden-Markov models (HMMs) are then used to infer a sequence of state transitions, where the states may correspond to either bird sound or noise.

Damoulas et al. [43] used DTW without template matching to obtain a similarity measure between calls (bypassing call-featurization), which was used in a kernel for SVM classification of warbler flight calls. This approach was applied after a band-specific segmentation algorithm extracted calls.

2.6 Limitations of Prior Work

Prior work on machine learning for bird song has limited applicability to general-purpose automatic acoustic monitoring in the real world for several reasons. Perhaps the most significant limitation is that prior work typically focusses on audio recordings containing only a single species of bird, while audio collected with omnidirectional microphones usually contains multiple simultaneously vocalizing birds. Many recordings used in prior work are obtained from birds in captivity, or with a directional microphone aimed by a person at a bird. Such recordings have a relatively high signal-to-noise ratio. In contrast, recordings obtained in a monitoring scenario often have background noises including wind, streams, rains, and motor vehicles. Some frequency-band methods (e.g, [62, 63]) are applicable to unattended monitoring, but must be crafted exclusively for a particular species (without the aid of machine learning).

Another issue not addressed by prior work is the efficient use of human effort for labeling training data. Through the MIML instance annotation framework, our proposed methods enable classification of syllables while training only on recordings paired with the set of species that are present (which requires less effort to obtain than annotating individual calls).

Quite recently (2013), the ARBIMON system [2] has been successfully applied to automated monitoring of birds, amphibians, mammals, and insects in field conditions. ARBIMON uses a 2D energy-based segmentation system to extract syllables. We demonstrated in earlier work

that a supervised 2D segmentation algorithm achieves better accuracy than a 2D energy-based algorithm [114]. A purely energy-based system cannot address non-stationary background noise such as rain, leading to false-positive detections [2]. This means their methods may have limited use in determining relationships between bird activity and covariates such as precipitation. We address segmentation in rain by extending the supervised classification method in [20]. In the ARBIMON system, segments are described by a 5-dimensional feature vector (minimum/maximum frequency, duration, bandwidth, and maximum intensity), of which the first 4 are a subset of the 38 features we proposed earlier in [18] (chapter 4). With this limited set of features, ARBIMON cannot be expected to scale well to a large number of species (their experiments involve 9 species), or handle subtle differences in calls. Furthermore, ARBIMON uses SISL classifiers (specifically, HMMs) and hence does not make efficient use of human effort in the same way as our proposed methods.

Chapter 3: Audio Classification of Bird Species: a Statistical Manifold Approach[1]

3.1 Introduction

Our goal is to develop algorithms that can predict which species of bird is present in an audio recording, by learning from a collection of labeled examples. Such algorithms will serve as part of a system to automatically collect bird species presence/absence data, which will provide valuable ecological information for species distribution modeling and conservation planning. Existing bird species distribution data are collected by manual surveys, which are labor intensive, and require observers trained in bird recognition [11]. Automated bird population surveys could provide vast amounts of useful data for species distribution modeling, while requiring less effort and expense than human surveys. Other applications of classifying bird sounds include reducing plane crashes caused by collisions with birds [30], and audio classification in general.

Sounds that birds make have a grammatical structure; two important levels of organization in this structure are songs and syllables. Syllables are single distinct utterances by a bird and serve as the basic building blocks of bird song [25]. A song consists of a series of syllables arranged in a particular pattern. In this study, our goal is to classify bird species from an interval of sound (containing one or more syllables), which roughly corresponds to the song level of organization.

Audio classification systems typically begin by extracting acoustic features from audio signals. Such features often pertain to individual frames (i.e., very short segments of signal). For example, one commonly used feature is the spectrum of a signal frame, which describes the intensity of (a short segment of) the signal as a function of frequency. To apply many standard algorithms for classification, it is necessary to represent a sound, which contains multiple frames, using a fixed-length vector. To construct such a fixed-length feature vector to describe a sound as a whole, a common approach is to first identify interesting frames by segmentation, compute features for those frames, then take the average of the features over all frames [60, 92, 136]. For

[1] "Audio classification of bird species: A statistical manifold approach." Forrest Briggs, Raviv Raich, Xiaoli Z. Fern. Ninth IEEE International Conference on Data Mining, 2009. ICDM'09.

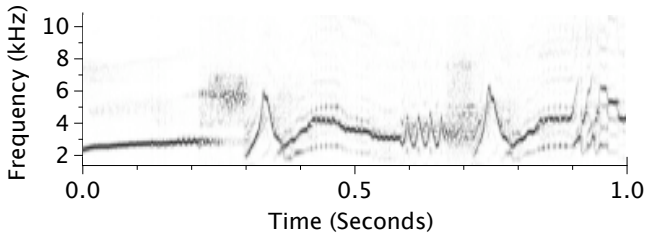


Figure 3.1: The spectrogram for a one-second portion of a recording of a Swainson's Thrush. Darker areas indicate higher energy at the corresponding frequency.

example, in the context of bird species recognition, a recent work by Fagerlund [60] (current state-of-the-art) averages frame-level features and applies support vector machines (SVMs).

Rather than averaging frame-level features, we represent their distributions using histograms (bag-of-codewords) defined by a 'codebook' of clustered frame-level features. Codebook based representations have been successfully applied in computer vision [41, 85, 154], and have also recently achieved success in music genre classification [128].

Our main contribution in this chapter is to establish a theoretical framework that connects nearest-neighbors classifiers using histograms of features, Bayesian risk minimization, and geodesics on statistical manifolds. In particular,

- We propose a probability model for audio, then follow a Bayesian approach to derive the risk-minimizing classifier for this model. The Bayes classifier is closely approximated by a nearest-neighbor classifier using Kullback-Leibler (KL) divergence to compare feature histograms (Sec. 3.3);

- We explain that not only do Kullback-Leibler and the related Hellinger distance follow from a Bayesian probability model, but in the limit of a nearest-neighbor classifier, they can be thought of as approximations to geodesics using the Fisher information metric on statistical manifolds of histograms (Sec. 3.4);

- We experimentally compare the accuracy of nearest-neighbors using L1, L2, KL and Hellinger distances, and SVMs, with averages and histograms of frame-level features, on a data set consisting of 413 thirty-second intervals of sound from six species of bird. Results indicate that classifiers using histograms of frame-level features outperform those

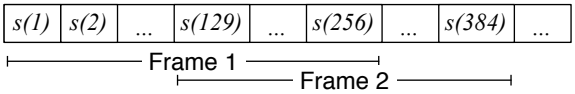


Figure 3.2: An audio signal is made up of samples, which are divided into overlapping frames.

using averages, and that with a manifold geodesic distance between histograms, nearest neighbor can outperform SVM. Several classifiers achieve over 90% accuracy (Sec. 3.5).

3.2 Background and Related Work

We review data representation for audio classification, and related work on species identification from bird sounds.

3.2.1 Data Representation

Our goal is to classify a recording of bird sound as one of several species. A critical initial step toward this goal is to extract meaningful features to describe an interval of sound. This section presents our approach to constructing feature vectors to describe such intervals.

3.2.1.1 Basics: Signals and Spectrograms

Audio signals consists of a time-series of *samples*, which we denote as $s(t)$. It is often easier to recognize patterns in an audio signal when samples are converted to a frequency domain *spectrogram* using the Fast Fourier Transform (FFT) [25], (see Fig. 3.1 for an example spectrogram).

To compute a spectrogram, samples in a sound are divided into overlapping frames (Fig. 3.2), each of which contains a fixed number of consecutive samples. The FFT is applied to each frame to obtain the complex Fourier coefficients. The magnitudes of these coefficients are called the frame's magnitude spectrum and represent the intensity of the sound during that frame at different frequencies. A spectrogram is a plot of the spectrum for each frame in a signal.

3.2.1.2 Frame-Level Features

Many features for audio classification describe individual frames of a signal. In this section, we describe three features that we used in our experiments.

Spectrum Density. The magnitude spectrum of a frame can be normalized to form a probability distribution. If the magnitude spectrum is $(|c_1|, \ldots, |c_l|)$, where l is the number of elements in a spectrum, then the spectrum density is $f(i) = \frac{|c_i|}{\sum |c_i|}$. We can directly use $(f(1), f(2), \ldots, f(l))$ as the feature vector describing a frame.

Mean Frequency and Bandwidth. Consider the spectrogram shown in Fig. 3.1; each vertical slice represents the spectrum of one frame of sound. Bird sounds are usually concentrated at a few frequencies; we can see this phenomenon as horizontal strips in the spectrogram. This suggests that it is possible to condense the information contained in the spectrum density into just two values: the mean frequency and the bandwidth of the spectrum. The mean frequency of a frame indicates the vertical position of the strip, while bandwidth describes the width of the strip. Specifically, the mean frequency is $f_c = \int x f(x) dx$, and bandwidth is $BW = \sqrt{\int (x - f_c)^2 f(x) dx}$.

Mel-Frequency Cepstral Coefficients. Mel-frequency cepstral coefficients [45] (MFCCs) are one of the most widely used features for audio classification. The idea is to first compute Mel-frequency coefficients (MFCs), which are like the magnitude spectrum, but in units of mels rather than Hz (mels correspond more closely with human perception of pitch [150]). MFCs are computed by applying a collection of triangular filters to the magnitude spectrum; the MFCs are the response of each filter. The filters are evenly spaced in the mel scale. MFCCs are the result of applying the discrete cosine transform (DCT) to the log of the MFCs.

3.2.1.3 Aggregating Frame-Level Features

An interval contains a large number of frames, which can be aggregated to produce a single fixed-length feature vector. A common approach that has been used in syllable classification is to average frame-level features [60, 92, 136]. However, by averaging, significant information about the distribution of features is lost, which can be problematic when the distribution of features in an interval is multimodal. For example, Fig. 3.3(b) shows the distribution of the

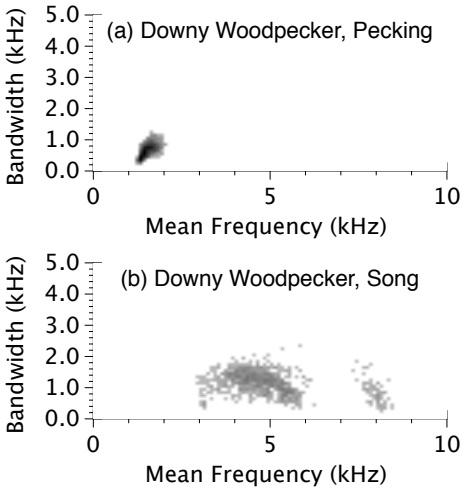


Figure 3.3: 2D histograms of frame mean frequency and bandwidth from two different intervals of audio recordings of the Downy Woodpecker.

features (mean-frequency and bandwidth) of the frames from a 30-second recording of a downy woodpecker (approximated by a 5000-bin histogram). In this case, the distribution is clearly multimodal and its mean will actually be in an area of relatively low probability, making it a poor representation for the overall distribution. We observed that such multimodality is common for bird sound. This observation suggests that aggregation schemes that can capture multimodality in feature distributions may be more successful than averages (our experimental results support this idea; Sec. 3.5.6). Inspired by the use of codebooks for image classification [41, 85, 154], and recent work in music genre classification [128], we consider aggregating frame-level features by representing their distributions with histograms.

Low-Dimensional Feature Histograms. Given an interval (i.e., a set of frames), each of which is described by a d-dimensional feature vector, a natural way to represent the interval is to use the probability distribution of features in this interval. This distribution can be approximated by a d-dimensional histogram, where dimension i is discretized into k_i bins, leading to a total of $\prod_{i=1}^{d} k_i$ bins. Note that since the total number of bins grows exponentially with d, this method can only be applied for small values of d. The vector of frequencies for each histogram bin can be used as a feature vector for classification.

Codebook Feature Histograms. The simple binning approach does not work for higher dimensional frame-level features such as spectra or MFCCs — we would need an infeasible number of bins to cover these high-dimensional spaces. Instead, we take a 'codebook' approach [128] to constructing histograms for high-dimensional features, which amounts to using non-uniform bins. A codebook is a collection of k codewords, each of which is a feature vector that is considered as representative in the feature space. There is one bin associated with each codeword. Given an interval (i.e., a set of frames each described by a feature vector) and a codebook, to compute a feature for the interval, assign each frame to its closest codeword, then count the number of frames assigned to each codeword. The vector of counts, normalized by dividing by their sum, gives the final feature vector, which is a histogram.

3.2.2 Related Work

Bird species can be classified using features extracted from audio recordings. A common approach to bird species classification is to identify distinct syllables, then construct feature vectors for those syllables and apply a standard classifier such as nearest neighbor or support vector machines to predict the species for each syllable [59, 60, 73, 89, 92, 135, 136]. Song-level species prediction has also been investigated using Hidden Markov Models [88, 136], Gaussian Mixture Models [136], based on comparisons of syllable-pair histograms [134], or nearest-neighbor classifiers using a feature constructed by aggregating syllable features [92].

To classify syllables or songs, most prior work relies on segmentation of the input audio into syllables [136]. As such, the classifier accuracy can be strongly dependent upon accurate segmentation [59]. A standard approach to segmentation is to compute the energy of each frame, then adaptively compute a threshold that separates syllables from background noise [60, 126, 126, 136]. It is difficult to obtain reliable segmentation using this method in recordings with low signal-to-noise ratio. In this chapter, we use a simple approach to detect a set of interesting frames within the signal that correspond to bird sound, and do not require that they precisely match syllable time-boundaries (Sec. 3.5).

Audio classification in general has been widely studied, with applications to human speech and music being the most common. Our work is closely related to recent work by Seyerlehner et al. [128] on music genre identification. They follow a codebook approach to constructing audio feature histograms (Sec. 3.2.1.3), and use a nearest-neighbor classifier with L1 distance to classify these features. However, it is not obvious why a nearest-neighbor classifier is ideal

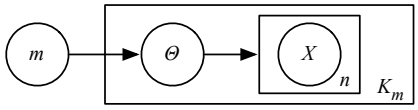


Figure 3.4: The plate diagram for the Interval-IID model.

for classifying histograms of features, or which distance measures are the best for comparing histograms. In this chapter, we show that the Bayes optimal classifier for a probability model for audio is closely related to nearest-neighbor classifiers using histograms of features with appropriate distance measures.

3.3 Probability Model for Sound

In the following section, we present a theoretical justification for the frame-level feature histogram representation through a probability model, namely the Interval-IID model, and show that the corresponding Bayes risk-minimizing classifier can be approximated by a nearest-neighbor classifier with KL divergence.

3.3.1 The Interval-IID model

The Interval-IID model follows the graphical representation in Fig. 3.4. The model suggests that to generate an interval, we first determine its class label m based on the class prior $p(m)$. Given m, we then generate an interval-specific parameterization θ based on $p(\theta|m)$, which parameterizes the the frame feature distribution $p_{x|\theta}(x|\theta)$ of that interval. Given θ, we then generate n independent and identically distritbuted (i.i.d.) frame feature vectors x_i based on $p_{x|\theta}(x|\theta)$ (thus the name Interval-IID, i.e., frames are i.i.d. within an interval).

Given an observed interval represented by its collection of frame features $X = [x_1, x_2, \ldots, x_n]$, using Bayes rule, we write its class-conditional probability as

$$p_{X|m}(X|m) = \int p_{X|\theta}(X|\theta)p(\theta|m)d\mu(\theta), \tag{3.1}$$

where θ is a parametrization determining the interval-conditional feature distribution $p_{x|\theta}(x|\theta)$ and m denotes the class label. Here we marginalized over the interval-conditional features probability model $p_{X|\theta}(X|\theta)$ according to the class conditional histogram parametrization dis-

Table 3.1: Summary of Notation

Variable	Description
m	class label (bird species)
n	number of interesting frames in an interval
θ	frame feature histogram parametrization
x_i	ith test frame feature vector
x_{ik}^t	ith training frame feature vector for the k training interval
X	frame feature vector collection for an interval $X = [x_1, \ldots, x_n]$
X_k^t	frame feature vector collection for the kth training interval
y_k^t	class label associated with the kth training interval
K	number of training intervals
K_m	number of training intervals from class m
$P(m)$	class prior probability
$p(\theta\|m)$	class-conditional histogram probability
$p_{x\|\theta}(x\|\theta)$	interval-conditional frame-feature probability
$p_{X\|\theta}(X\|\theta)$	interval-conditional features probability
$p_{X\|m}(X\|m)$	class-conditional features probability

tribution $p(\theta|m)$. As the Interval-IID model name suggests, conditioned on θ, the frame-level features are assumed i.i.d., and hence $p_{X|\theta}(X|\theta)$ can be written as a product of the marginal distributions of each frame-level feature:

$$p_{X|\theta}(X|\theta) = \prod_{i=1}^{n} p_{x|\theta}(x_i|\theta), \tag{3.2}$$

where x_i denotes the feature vector of the ith frame. Substituting (3.2) into (3.1), the class conditional model for the Interval-IID model is given by

$$p_{X|m}(X|m) = \int \prod_{i=1}^{n} p_{x|\theta}(x_i|\theta) p(\theta|m) d\mu(\theta). \tag{3.3}$$

The integral w.r.t. θ here applies to the product of marginal probabilities. By writing p as $e^{\log p}$ and replacing the integral with the expectation notation, (3.3) becomes

$$p_{X|m}(X|m) = E_\theta\big[e^{\sum_{i=1}^n \log p_{x|\theta}(x_i|\theta)}|m\big]. \tag{3.4}$$

To express $p_{X|m}(X|m)$ in (3.4) in terms of the Kullback-Leibler (KL) divergence, start by introducing the following terms:

$$\theta^* = \arg\max_\theta \frac{1}{n}\sum_{i=1}^n \log p_{x|\theta}(x_i|\theta), \tag{3.5}$$

$$\tilde{H}(\theta^*) = -\frac{1}{n}\sum_{i=1}^n \log p_{x|\theta}(x_i|\theta^*), \tag{3.6}$$

$$D(\theta^*,\theta) = \frac{1}{n}\sum_{i=1}^n \log \frac{p_{x|\theta}(x_i|\theta^*)}{p_{x|\theta}(x_i|\theta)} \tag{3.7}$$

Note that θ^* is the maximum-likelihood estimator of θ. Using (3.5-3.7) and the observation that $\sum_{i=1}^n \log p_{x|\theta}(x_i|\theta) = -n(\tilde{H}(\theta^*) + D(\theta^*,\theta))$, we rewrite (3.4) as

$$p_{X|m}(X|m) = e^{-n\tilde{H}(\theta^*)} E_{\theta|m}\big[e^{-nD(\theta^*,\theta)}\big]. \tag{3.8}$$

By the definition of θ^* in (3.5), we have that $D(\theta^*,\theta) \geqslant 0$ for all θ and is zero for $\theta = \theta^*$. We proceed with the specific case in which the features are discretized into L non-intersecting bins defined by the sets A_l. Hence, we represent the class-conditional distribution of frame-level features using histograms. Each frame-level feature x_i can fall into one of the histogram bins $\{A_1,\ldots,A_L\}$ with probability $\{\theta_1,\ldots,\theta_L\}$, respectively. The vector $\theta = [\theta_1,\ldots,\theta_L]^T$ is a probability mass function (or a histogram), i.e., $\sum \theta_l = 1$ and $\theta_l \geqslant 0$. The interval-conditional probability model for a frame-level feature is given by

$$p_{x|\theta}(x|\theta) = \prod_{l=1}^L \theta_l^{I(x\in A_l)}, \tag{3.9}$$

where $I(\cdot)$ is the indicator function which takes the value one if its argument is true and zero otherwise. We would like to point out that when $p_{x|\theta}(\cdot|\theta)$ is given by (3.9) and $p(\theta|m)$ is the Dirichlet distribution, then (3.3) becomes the Dirichlet-Multinomial model, which is also

referred to as Polya distribution [113] or the Dirichlet compound multinomial (DCM) model [56]. This model is often used as a topic model in text document classification. One criticism concerning the choice of Dirichlet prior is the limited capability of representing multimodal priors [163]. Our experience with bird sounds suggests that the probability model $p(\theta|m)$ is indeed multimodal; as Fig. 3.3 shows, frame-level feature histograms for the same species differ between intervals.

For $p_{x|\theta}(x|\theta)$ given by (3.9), we have

$$\theta_l^* = \hat{p}_l, \tag{3.10}$$

$$\tilde{H}(\theta^*) = H(\theta^*) \tag{3.11}$$

$$D(\theta^*, \theta) = D_{kl}(\theta^* \| \theta), \tag{3.12}$$

where $\hat{p}_l = \frac{1}{n}\sum_{i=1}^{n} I(x_i \in A_l)$ is the lth empirical histogram bin probability estimate based on the observed feature collection $X = [x_1, x_2, \ldots, x_n]$, $H(p) = -\sum_{l=1}^{L} p_l \log p_l$ is the entropy associated with a multinomial parameterized by p (the vector of bin probabilities of a histogram), and

$$D_{kl}(\theta^* \| \theta) = \sum_{l=1}^{L} \theta_l^* \log \frac{\theta_l^*}{\theta_l}$$

is the Kullback-Leibler (KL) divergence between a multinomial parameterized by θ^* and another parameterized by θ. Substituting (3.10)-(3.12) into (3.8), we have

$$p_{X|m}(X|m) = e^{-nH(\hat{p})} E_\theta\big[e^{-nD_{kl}(\hat{p}\|\theta)}|m\big]. \tag{3.13}$$

This form for $p_{X|m}(X|m)$ acts as the likelihood component in the Bayes risk minimizing classifier in the following section. Moreover, it highlights the role of the KL divergence in optimal Bayesian classification for the problem at hand.

3.3.2 Bayes Risk Minimizing Classifier

We start with a brief review of the Bayes risk minimization approach to classification [67]. The probability of error for a given classification rule $\hat{m}(X)$ is

$$Pr(\text{error}) = \sum_{m=1}^{M} P(\hat{m}(X) \neq m|m)P(m). \tag{3.14}$$

The classification rule that minimizes the error in (3.14) is

$$\hat{m} = \arg\max_{m} \; p_{m|X}(m|X). \tag{3.15}$$

This rule is also referred to as maximum a-posteriori (MAP), as it assigns a decision based on the highest class probability given the set of observations. Using Bayes rule $p_{m|X}(m|X) = p_{X|m}(X|m)P(m)/p_X(X)$ and the fact that $p_X(X)$ is constant w.r.t. to the class variable m, yields an equivalent form to the MAP classifier:

$$\hat{m} \;\; = \;\; \arg\max_{m} \;\; \log P(m) + \log p_{X|m}(X|m). \tag{3.16}$$

After replacing the likelihood $P_{X|m}(X|m)$ with (3.13), the MAP classification rule (3.16) for the Interval-IID model in (3.3) is

$$\begin{aligned} \hat{m} \;\; = \;\; & \arg\max_{m} \;\; \log P(m) \\ & + \log E_\theta\big[\exp\big(-nD_{kl}(\hat{p}\|\theta)\big)|m\big]. \end{aligned} \tag{3.17}$$

Note that since $H(\hat{p})$ in (3.13) is independent of m, it is not incorporated into (3.17). It is equivalent to replace the maximization with minimization and divide by n

$$\hat{m} = \arg\min_{m} \;\; -\frac{1}{n}\log\Big(E_\theta\big[e^{-nD_{kl}(\hat{p}\|\theta)}|m\big]P(m)\Big). \tag{3.18}$$

With no exact knowledge about $P(m)$ and the PDF $p(\theta|m)$ used to compute the expectation $E_\theta[\cdot|m]$, we propose estimating these quantities from the training data.

3.3.3 Training

To describe the training process, we start by explaining the format of the training data. Each interval k in the training data contains n training features $X_k^t = [x_{1k}^t, x_{2k}^t, \ldots, x_{nk}^t]$ and is associated with a class label y_k^t. We assume that K training intervals are available, i.e., $(X_1^t, y_1^t), (X_2^t, y_2^t), \ldots, (X_K^t, y_K^t)$. We denote training variables using the superscript t notation.

To train the classification rule in (3.18), we replace $P(m)$ and $E[\cdot|m]$ through their sample estimates

$$\hat{m} = \arg\min_m \ \frac{-1}{n} \log\Big(\sum_{k=1}^{K} I(y_k^t = m) e^{-nD_{kl}(\hat{p}\|\hat{\theta}^{(k)})} \Big), \tag{3.19}$$

where k is the interval number, K is the total number of training intervals, y_k^t is the class label for the kth training interval, and $\hat{\theta}^{(k)}$ is a histogram estimated from the kth training interval given by

$$\hat{\theta}_l^{(k)} = \frac{1}{n} \sum_{i=1}^{n} I(x_{ik}^t \in A_l), \tag{3.20}$$

where x_{ik}^t is the ith feature vector from the kth interval. With a slight abuse of notations, we rewrite (3.19) as

$$\hat{m} = \arg\min_m \ -\frac{1}{n} \log\Big(\sum_{k=1}^{K_m} e^{-nD_{kl}(\hat{p}\|\hat{\theta}^{(k,m)})} \Big), \tag{3.21}$$

where the $\hat{\theta}^{(k,m)}$'s are the sorted version of the $\hat{\theta}^{(k)}$s from class m such that $D_{kl}(\hat{p}\|\hat{\theta}^{(1,m)}) \leqslant D_{kl}(\hat{p}\|\hat{\theta}^{(2,m)}) \leqslant \neg \ldots D_{kl}(\hat{p}\|\hat{\theta}^{(K_m,m)})$, and K_m is the number of training intervals for the mth class. Using the ordered training class histograms, we reorganize (3.21) as

$$\begin{aligned} \hat{m} \ = \ & \arg\min_m \ D_{kl}(\hat{p}\|\hat{\theta}^{(1,m)}) \\ & -\frac{1}{n} \log\Big(1 + \sum_{i=2}^{n_m} e^{-n(D_{kl}(\hat{p}\|\hat{\theta}^{(i,m)}) - D_{kl}(\hat{p}\|\hat{\theta}^{(1,m)}))} \Big). \end{aligned} \tag{3.22}$$

We refer to (3.22) as the Interval-IID MAP classifier. While equivalent to (3.21), (3.22) provides insight into the relation between Bayes risk-minimization, nearest-neighbor classifiers, and man-

ifold geodesics. Identifying the training intervals with their feature histograms, and the test interval with its feature histogram, the first term on the RHS of (3.22) is a KL divergence based nearest neighbor rule in histogram space. Note that if the KL distance to points other than the first nearest neighbor $D_{kl}(\hat{p}\|\hat{\theta}^{(i,m)})$ is sufficiently larger than the distance to the first nearest neighbor $D_{kl}(\hat{p}\|\hat{\theta}^{(1,m)})$ then the second term on the RHS of (3.22) becomes negligible, and (3.22) is simply a nearest neighbor classifier using KL divergence.

3.4 Nearest Neighbors on Statistical Manifolds

The connection between optimal Bayes classification and the histogram KL nearest neighbor rule leads us to extend the approach to nearest neighbor classification on histograms. Note that a collection of probability models (i.e, histograms) can be regarded as a manifold. Denote a model by $p(X|\theta)$ or in short by $p(\cdot|\theta)$. The collection of models given by

$$\mathcal{M} = \left\{ p(\cdot|\theta) \mid \theta \in \Theta \in \mathbb{R}^d \right\}, \tag{3.23}$$

is a d-dimensional statistical manifold if there exist a one-to-one smooth mapping between θ to $p(\cdot|\theta)$. In the geometric approach to statistical models [87], one can measure the geodesic distance between two histograms by using the Fisher information metric (FIM) as the Riemannian metric

$$D_F(p(\cdot|\theta), p(\cdot|\theta')) = \min_{\substack{\theta(\cdot), \\ \theta(0)=\theta, \\ \theta(l)=\theta'}} \int_0^l \sqrt{\dot{\theta}(t)^T \mathcal{I}(\theta(t)) \dot{\theta}(t)} dt, \tag{3.24}$$

where $\mathcal{I}(\theta)$ is the Fisher information matrix given by

$$\mathcal{I}_{ij}(\theta) = E\Big[\frac{d \log p(x|\theta)}{d\theta_i} \frac{d \log p(x|\theta)}{d\theta_j} \Big]. \tag{3.25}$$

The FIM is considered a natural metric for statistical manifolds as it reflect the capability to discriminate between probability models from their samples.

To generalize the nearest neighbor approach discussed in the previous section in the context of statistical manifolds, we consider a geodesic nearest neighbor rule using $D_F(p(\cdot|\theta), p(\cdot|\theta'))$ defined in (3.24). As the precise form of the manifold is unavailable, an exact computation of

the geodesic distance $D_F(p, p')$ is impossible. Since the nearest neighbor approach prompts us to calculate short geodesic distances, local approximations of $D_F(p, p')$ can be used instead. For two close probability models $p \to p'$ it is known [87] that $\sqrt{2D_{kl}(p\|p')} \to D_F(p, p')$. The KL divergence provides a computable approximation to the FIM manifold geodesic distance.

Note that other approximations for the FIM are available (e.g., certain Ali-Silvey divergences, and specifically, Hellinger divergence). In this chapter, we use the Hellinger divergence given by

$$D_H(p, q)^2 = \sum(\sqrt{p_i} - \sqrt{q_i})^2, \tag{3.26}$$

which is a metric as opposed to the KL divergence. The approximation of the FIM using Hellinger distance for close models is $2D_H(p, p') \to D_F(p, p')$ [87].

For the purpose of comparison, we experimentally evaluate nearest neighbor classifiers using L1 and L2 distances as well as KL and Hellinger. L2 is the standard Euclidean distance, which is widely used, but not theoretically justified for the comparison of probability distributions. L1 is fairly common for comparing probability distributions. It is a member of the Ali-Silvey family, but due to non-differentiability, it is not an approximation to the FIM. However, it is related to Hellinger by the inequality, $\frac{1}{2}D_H(p, q)^2 \leqslant D_{L1}(p, q) \leqslant D_H(p, q)$ [44]. This relation between L1 and Hellinger hints at why classifiers using these distances achieve similar results (Sec. 3.5.6).

3.5 Experiments

In this section, we describe the experimental setup used to measure the accuracy of the proposed methods for bird species classification, and to compare with SVMs [60]. We consider various frame-level features (mean frequency and bandwidth, MFCCs, and spectral density), interval-level features (averages vs. histograms), and metrics for nearest-neighbor classification (L1, L2, KL, and Hellinger). We also empirically verify that a nearest-neighbor classifier using Kullback-Leibler closely approximates the Interval-IID MAP classifier (3.22) as suggested in Sec. 3.3.

3.5.1 Data

We have 1.13 GB of recordings from the Cornell Macaulay library, of 6 species: Black Throated Blue Warbler, Hermit Warbler, Downy Woodpecker, Swainson's Thrush, Western Tanager, and Winter Wren. All of these recordings are at least 30 seconds long, and most are less than 10 minutes. We divide each recording into intervals of 30 seconds, resulting in 413 intervals. Our goal is to classify these intervals according to species.

The recordings were collected over several decades, mainly in the western United States. Most are made using a directional microphone in the field. The amount of noise in the recordings varies widely. In addition to static and wind, some recordings contain cars sounds, human speech, and other non-bird sounds. We manually removed most portions of sound with human voices. Although each recording is labeled with just one species, some recordings contain multiple birds, sometimes of different species; usually the loudest bird present corresponds to the label for the recording. The sampling frequency for all recordings is 44.1 kHz. The audio data is stored as mono-channel WAV files.

3.5.2 Preprocessing

Section 3.2.1 covers the process of converting a sequence of samples from an audio interval into interval-level features. We proceed by further elaborating on the specific details of our experimental setup.

When dividing a signal into frames, we use 256 samples per frame, and successive frames overlap by 50%. To reduce noise and decrease processing time in later stages, we discard the lowest 8 and highest 64 elements of each frame's spectrum, leaving 56 elements from the original 128 (equivalent to removing all sound below 1.378 kHz and above 10.852 kHz).

Instead of syllables, we detect a subset of interesting frames (which are more likely to contain bird sound) in an interval. To find these interesting frames, we compute the total magnitude of each frame, $\sum |c_i|$, and retain only the 10% of frames with highest total magnitude in all subsequent calculations. Note that the total magnitude is similar to, but not the same as the energy of a frame.[2]

Our implementation of MFCCs (Sec. 3.2.1.2), is based on the description provided by

[2]Parseval's theorem states that the energy of a frame can be computed from its spectrum via the formula $E = \sum |c_i|^2$, were c_i is the ith FFT coefficient.

Ganchev et al. [68] of the MFCCs computed in the Cambridge Hidden Markov Models Toolkit (for MATLAB), known as HTK [162]. We use 24 filters,[3] resulting in 24 MFCs, then take only the first 12 elements of the output of the DCT as the frame-level feature.

For constructing 2D histograms, we divide the range of values for mean frequency and bandwidth into square bins 100 Hz wide, with 100 bins on the mean frequency axis, and 50 bins for the bandwidth axis (for a total of $50 \times 100 = 5000$ bins, covering a range of 0 Hz to 10,000 Hz for f_c and 0 Hz to 5000 Hz for BW). There is one element in the feature vector for each histogram bin, so this representation results in a 5000-dimensional feature vector.[4]

3.5.3 Clustering for Codebooks

For constructing codebooks, we apply the k-means++ clustering algorithm [5] to the frame-level features from a training data set. Note that there are several hundred-thousand frames to cluster in our data set. To speed-up codebook construction, we follow a two-staged clustering proceedure suggested by Seyerlehner et al. [128]. In particular, we first cluster features within each 30-second interval, then cluster the resulting cluster centers to obtain the final codewords. In the first stage of clustering, the feature vectors are either the spectrum density, or MFCCs for the interesting frames. In the second stage, the examples are the cluster centers from the first stage. We use $k = 10$ clusters for the first stage and $k = 100$ for the second. Thus, the final interval-level features constructed using this method are 100-dimensional. In our preliminary experiments, this approach to clustering yielded an order-of-magnitude speedup over clustering all frame-level features at once, because the first stage of clustering does not need to be repeated in each fold of cross-validation.

3.5.4 Classifiers

There are many combinations of frame-level features and methods of aggregating them. The combinations we consider in this study are: averages of f_c and BW, spectrum density, and

[3] In our implementation, the filters span a range of frequencies from $f_{low} = 1000Hz$ to $f_{high} = 22050Hz$. Following an exact implementation of the filters described by Ganchev et al. [68], we got aliased triangle filters because some were narrower than a single spectrum bin, which caused artifacts in the MFCs. To fix this problem, we numerically integrate the triangle filter function over the range of each bin. Many other implementations of MFCCs work with lower sampling frequencies [68], so we suspect this problem is related to working with sound sampled at 44.1 kHz, as well as our choice of values for f_{low} and f_{high}.

[4] We apply Laplace smoothing to the histogram estimation by starting with a count of 1 for each bin.

MFCCs, 2D histograms of f_c and BW, and codebook histograms of spectrum density and MFCCs.

Using the above features extracted from the data described in Sec. 3.5.1, we compare several classification algorithms: nearest neighbor with L1, L2, KL and Hellinger distances, and the Interval-IID MAP classifier proposed in Sec. 3.3.2 (3.22), as well as support vector machines. Of these classifiers, Interval-IID Map, KL and Hellinger are our proposed methods, and the others are included for comparison.

3.5.4.1 Support Vector Machines

Support vector machines [38] (SVMs) are a family of algorithms for supervised classification that find a linear decision boundary by maximizing the margin between two classes. In cases where linear classification is insufficient, the kernel trick is applied to non-linearly project features into a higher dimensional space where linear separability is possible. The implementation of SVMs that we used is WLSVM [54], which integrates LIBSVM [26] into the Weka [155] machine learning system. Following Fagerlund [60], and the recommendations of Hsu, Chang and Lin [79], we use a radial basis function kernel, and optimize the SVM parameter C and the kernel parameter γ, by grid search. We evaluate the SVM at all combinations of C and γ in $\{10^{-1}, 10^0, 10^1, 10^2\}$, and report the best accuracy achieved with any set of parameters. To handle multiple classes (in our case, species), LIBSVM use the one-against-one voting scheme [78].

3.5.5 Cross Validation and Multiple Trials

To measure the accuracy of the proposed classifiers, we use them to predict the species in each of 413 thirty-second intervals of sound. Each classifier is trained using all of the intervals that do not come from the same recording as the interval being classified (the data set consists of longer recordings that are split into intervals). We use this setup so the classifier must identify species without already having example recordings of the individual bird being classified. Fagerlund [60] used a similar 'individual independent' setup for cross-validation.

Classifiers that use a codebook to construct feature histograms depend on a randomized clustering algorithm. To account for the randomness, we ran five trials with different random seeds, and report average accuracy, $\pm$ average deviation.

3.5.6 Results

Table 3.2 lists the accuracy of each classifier on the species recognition problem. We make the following key observations.

- Regardless of which frame-level features we use, histograms of features achieved better accuracy than averages. One possible explanation of this result is that feature distributions may be multimodal (Fig. 3.3(b)), so the mean alone may not be enough to discriminate between distributions from different species.

- Using the 2D histogram of f_c and BW, the Interval-IID MAP classifier (3.22) produced identical results to a nearest-neighbor classifier with KL divergence. This result confirms our theoretical argument that a nearest neighbor classifier using KL divergence is a close approximation to the Interval-IID MAP classifier. Accordingly, we recommend using the more efficient nearest neighbor with KL as opposed to (3.22), when audio data is believed to be generated as in the Interval-IID model.

- Comparing different distance functions when using histograms, we observe that L1, Hellinger and KL were generally more accurate than using L2, with the performance of Hellinger being the most robust across different settings. Interestingly, while L1 is not an approximation to the FIM, its performance is highly competitive to KL and Hellinger. For histograms of spectrum density, L1 slightly outperformed Hellinger (although not statistically significantly). This is possibly due to the close relationship between L1, Hellinger and KL, as explained in Sec. 4. Note that MFCCs are essentially a compressed version of the spectrum (from 56 elements to 12), so it is not surprising that classifiers using them are slightly less accurate than those using spectrum density.

- Despite their relative simplicity, classifiers using 2D histograms of mean frequency and bandwidth provide remarkably accurate predictions. Being able to visualize a 2D histogram as an image provides insight into the structure of bird sound (for example, we can see that interval feature histograms may be multimodal).

- Finally, we note that the proposed methods achieved accuracy similar to or better than SVMs. In particular, using codebook histograms of MFCCs, SVMs are slightly more accurate than a nearest neighbor classifier with Hellinger, although the difference is not statistically significant. On codebook histograms of spectrum density, nearest-neighbor

classifiers using statistical divergence measures (i.e. L1, Kullback-Leibler and Hellinger) outperform SVM. We want to emphasize that unlike SVMs, which require significant parameter tuning, the proposed methods also offer additional advantages in terms of their simplicity and scalability, making them more usable in practice.

3.6 Conclusion & Future Work

In this chapter, we addressed the problem of bird species classification from audio recordings. Following a Bayesian approach to classification, we introduced the interval-IID model to describe the distribution of feature vectors within an interval consisting of frames, and derived the corresponding MAP classifier. The MAP classifier suggests aggregating features into histograms and using KL nearest neighbor to classify. This connection to nearest neighbor classification on statistical manifolds led us to extended the classifier by proposing different metrics (e.g., Hellinger). To use the MAP classifier with high-dimensional frame-level features, we employ codebook histograms.

Our study suggests that **1)** using histograms of frame-level features in an audio classifier can produce better results than using averaged frame-level features **2)** nearest-neighbor classifiers using Kullback-Leibler and Hellinger distance to compare feature histograms results are competive with state-of-the-art method such as SVM and **3)** metrics appropriate for histograms such as Hellinger, KL, and L1 perform better than the Euclidean L2 metric.

The classifiers in this study make predictions from intervals based on the collection of frames within the interval. A common alternative is to instead focus on individual syllables. We are working on an experimental survey of methods for classifying bird species from syllables, as well as probability models that are specialized for this purpose.

The experiments and algorithms presented here are a preliminary step toward analyzing a large (terabyte scale) data set of bird sounds that our collaborators collected in field conditions, using an array of omnidirectional microphones. We intend to apply algorithms for bird species classification to these recordings to extract information about patterns of bird activity at an unprecedented spatial and temporal resolution.

Table 3.2: The accuracy of each classifier in predicting bird species based on 413 thirty-second intervals of sound. NN means nearest neighbor. The values listed for classifiers using a codebook are average accuracy over 5 trials, ± average deviation. Our proposed methods are listed in bold.

Frame Feature	Representation	Classifier	% correct
f_c, BW	Average	NN-L1	42.85
f_c, BW	Average	NN-L2	42.85
MFCCs	Average	NN-L1	81.11
MFCCs	Average	NN-L2	81.11
MFCCs	Average	SVM	84.50
Spectrum Density	Average	NN-L1	79.42
Spectrum Density	Average	NN-L2	81.35
Spectrum Density	Average	SVM	84.75
f_c, BW	2D Histogram	**Interval-IID MAP**	87.40
f_c, BW	2D Histogram	**NN-Kullback-Leibler**	87.40
f_c, BW	2D Histogram	**NN-Hellinger**	88.13
f_c, BW	2D Histogram	NN-L1	86.44
f_c, BW	2D Histogram	NN-L2	83.05
MFCCs	Codebook	NN-L1	84.41 ± .89
MFCCs	Codebook	NN-L2	83.49 ± .62
MFCCs	Codebook	**NN-Kullback-Leibler**	85.42 ± .62
MFCCs	Codebook	**NN-Hellinger**	86.59 ± .50
MFCCs	Codebook	SVM	87.17 ± .58
Spectrum Density	Codebook	NN-L1	92.54 ± .44
Spectrum Density	Codebook	NN-L2	88.28 ± .99
Spectrum Density	Codebook	**NN-Kullback-Leibler**	90.70 ± .40
Spectrum Density	Codebook	**NN-Hellinger**	92.10 ± .27
Spectrum Density	Codebook	SVM	88.14 ± .58

Chapter 4: Acoustic Classification of Multiple Simultaneous Bird Species: a Multi-Instance Multi-Label Approach[1]

4.1 Introduction

Current and projected declines in biodiversity as a function of habitat loss [6] and climate change [116] necessitate the development of efficient and accurate estimates of species' diversity, habitats, and phenology. Birds have been used widely as indicators of biodiversity because they provide critical ecosystem services, respond rapidly to change, are relatively easy to detect, and may reflect changes at lower trophic levels (e.g., insects, plants) [125]. Birds have thus been proposed as "canaries in the coal mine" with respect to anthropogenic environmental changes at both local and global scales.

Unfortunately, collection of data on trends in birds populations has been plagued by problems of poor sample representation in remote regions, observer bias [10], imperfect detectability [106], and particularly the prohibitive costs of sampling over large spatial and temporal scales at sufficiently fine resolutions [123]. These problems could be ameliorated to some degree with the use of automated acoustic surveys. However, the complexity of bird song, the noise present in most habitats, and the simultaneous song that occurs in many bird communities [104, 105] make automated species identification a challenging task.

Many authors have proposed methods for acoustic bird species classification, but more work is needed to address the problem of identifying all species present in noisy recordings containing multiple simultaneously vocalizing birds [124]. It is common to classify species under the assumption that there is a single bird species present in a recording [60, 88, 111]. This assumption is reasonable for audio collected with hand-held directional microphones aimed at a target individual [33, 142, 149], or for audio collected from birds in captivity [3], but not for audio collected by unattended omnidirectional microphones [37]. A related problem is detection of one or a few specific species [8, 37] (possibly amidst other sources of noise, including other birds), or detection of birds that make a particular type of call (e.g., tonal sounds [83]).

[1] "Acoustic Classification of Multiple Simultaneous Bird Species: a Multi-Instance Multi-Label Approach." Forrest Briggs, Balaji Lakshminarayanan, Lawrence Neal, Xiaoli Z. Fern, Raviv Raich, Sarah J.K. Hadley, Adam S. Hadley, Matthew G. Betts. The Journal of the Acoustical Society of America, 2012 volume 131(6), p4640-50.

Unlike prior work in automatic bird sound detection and classification, we consider the following problem: given an audio recording (e.g., ten-seconds), predict the *set* of all species present in that recording.

We formulate this problem in the multi-instance multi-label (MIML) framework for supervised classification [171]. The main idea of MIML is that the objects to be classified are represented as a collection of parts (referred to as a "bag-of-instances"), and associated with multiple class labels. In this application, the objects to be classified are recordings, the parts are segments of the spectrogram corresponding to syllables of bird sound described by a feature vector of acoustic properties, and the labels are the species present. All supervised classification algorithms require some labeled training data to build a predictive model. A major advantage of the MIML formulation is that the only training data required is a list of the species present in a recording, rather than a detailed annotation of each segment, or training recordings containing only a single species (which is required in most prior work) [30, 92, 126, 136]. For recordings containing multiple simultaneously vocalizing species of bird, it is less labor intensive to construct the former type of labels.

In order to apply MIML classification algorithms, it is necessary to transform the data from its original representation into a suitable bag-of-instances representation. An algorithm to do this is called a "bag generator". In prior work, MIML bag generators for images and text have been proposed, but MIML has not previously been applied to audio. We propose a MIML bag generator for audio, which makes it possible to apply existing MIML classifiers to the species set prediction problem.

We experimentally evaluate MIML acoustic species classification on 548 ten-second recordings containing 13 species. These experiments demonstrate that our methods accurately predict the set of species present in noisy, multi-bird recordings collected in the field by unattended omnidirectional microphones.

4.2 Background & Related Work

In this section, we discuss segmentation, features and classifiers used in prior work on acoustic species classification. Then we review the multi-instance multi-label framework for supervised classification.

4.2.1 Acoustic Bird Species Classification

Brandes provides a survey of methods for acoustic bird species classification [124]. There are three main stages in most bird species classification systems: segmentation, feature construction, and supervised classification.

A syllable is a single short utterance by a bird, which may be a call, or part of a song. Methods for acoustic classification of bird species can broadly be grouped into those that classify individual syllables, and those that classify recordings containing multiple syllables. In both cases, segmenting audio into distinct syllables is a crucial step. The accuracy of any classifier that relies on segmentation is sensitive to the quality of the segmentation [59].

Most algorithms for segmentation operate in the time domain, and are based on energy. It is common to compute the energy of the signal in each frame, then consider intervals with high energy to be syllables [60, 84, 126, 136].

Energy-based, time-domain segmentation is not well suited for audio with high-noise or multiple simultaneous birds. Energy-based segmentation accuracy degrades in high-noise recordings (e.g., from wind, stream noise, or motor vehicles), and also cannot differentiate other loud non-bird sounds. Vocalizations from multiple birds may overlap in time, making time-domain segmentation ineffective. Further work is needed to extract measurements from syllables that overlap in time but not frequency, and in high-noise environments [124].

There has been some prior work on 2D time-frequency segmentation, which is better suited to audio with multiple simultaneous birds. Mellinger and Bradbury [112] used 2D segmentation in the form of bounding boxes for vocalizations of marine mammals, but this algorithm requires a human to provide a rough box first. Brandes divides the frequency range of a recording into several automatically determined bands, then applies a 2D energy threshold within each band [13]. We showed in earlier work that a random forest [14] classifier applied to each pixel of a spectrogram achieves higher segmentation accuracy than a 2D energy threshold on field-collected recordings [114].

After running a segmentation algorithm to identify syllables, systems for bird species recognition extract acoustic features to characterize the syllables in a way that can be used with machine learning algorithms for classification. Linear Predictive Coding (LPC) [30, 84, 111] and Mel-frequency cepstral coefficients (MFCCs) [45, 150], are common in analysis of speech and music and are amongst the most widely used features to describe bird sound [23, 60, 89, 92]. Features such as LPCs and MFCCs describe individual frames of sound; to characterize a

syllable as a whole, a common approach is to average the frame-level features [60, 92, 136]. Other features that have been used to characterize syllables include spectral peak tracking [73, 74, 136], analysis-by-synthesis/overlap-add [30], wavelets [126], and 'descriptive parameters' such as bandwidth, zero-crossing rate and spectral flux [60, 136].

The algorithms that have been applied to acoustic bird species classification either at the syllable or interval level (or both) all follow the standard single-instance, single-label framework (SISL) in machine learning, i.e. they associate a single feature vector with a single class label. SISL algorithms that have been applied to bird species classification include nearest-neighbor and distance based classifiers [30, 73, 92, 136], neural nets [23, 84, 111, 126], self-organizing maps [126], decision trees [149], support vector machines [60], hidden Markov models [13, 88, 136, 142], and Gaussian mixture models [136]. We elaborate on the differences between SISL and MIML in the following section.

4.2.2 Multi-Instance Multi-Label Learning

In traditional supervised classification, we are given a collection of training examples, each of which consists of a feature vector and a class label. The goal is to learn from the training examples how to assign a class label to a previously unseen feature vector. However, in some applications, it is natural for the objects of interest to be represented as a collection of parts (referred to as a bag-of-instances), where each part is described by a fixed-length feature vector. Multi-instance learning [49] incorporates such structure into the classification model. For example, in multi-instance image classification, an image is a bag, and the instances are features describing pixels, patches or regions [172]; in multi-instance text classification, a document is a bag, and the instances correspond to paragraphs or sub-windows of text [160, 171]. In this study, a short audio recording is a bag, and the instances correspond to 2D segments in the time-frequency domain described by a vector of their acoustic properties (these segments roughly correspond to syllables).

The original formulation of multi-instance learning [49] concerns problems where bags have single binary labels. Zhou [170] and Foulds and Frank [64] provide surveys on multi-instance learning, mainly focussing on the binary label case. Recently, Zhou and Zhang [171] proposed multi-instance multi-label (MIML) learning, where there a multiple classes and bags have a set of multiple labels.

Numerous algorithms for MIML have been proposed, and achieve superior accuracy in

image and text domains to prior approaches that do not model the multi-instance or multi-label aspects of a problem explicitly [97, 132, 165, 172]. Practical applications include labeling anatomical structures in images of *Drosophila* embryogenesis [97], and predicting tags for web pages on a social bookmarking site [132].

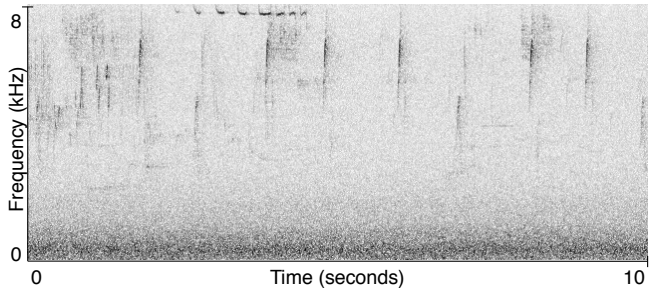
MIML has not previously been applied to audio, but Mandel and Ellis [108] recently applied multi-instance learning to classify music clips. Our proposed representation is at a different temporal scale, and is designed for bird sound rather than music.

A major advantage of MIML is that it is often easier or less costly to obtain labels at the bag-level. To the best of our knowledge, Brandes [13] is the only prior author to address acoustic species classification with multiple simultaneous species. In his work, the goal is to classify individual bird, frog, or cricket calls; this requires training data in the form of individually labeled calls. In contrast, because we use a MIML formulation, we predict a set of species present rather than the species for each vocalization. However, we only require a list of the species in each recording (bag) for training data.

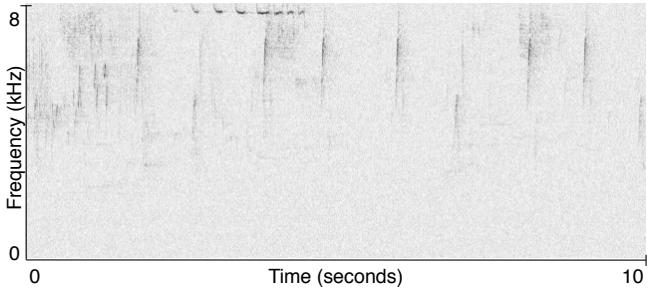
The MIML framework is formalized as follows: Suppose we have a feature space $\mathcal{X}$ (usually $\mathcal{X} = \mathbb{R}^d$), and set of labels $\mathcal{Y} = \{1, \ldots, c\}$. In SISL learning, the training dataset is $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, where $\mathbf{x}_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. A SISL classifier is a function $f : \mathcal{X} \to \mathcal{Y}$, i.e. it maps feature vectors to single class labels. In MIML, the dataset is $(X_1, Y_1), \ldots, (X_m, Y_m)$, where $X_i = \{\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,n_i}\}$ is a bag of n_i instances (i.e. feature vectors), and $Y_i \subseteq \mathcal{Y}$ is a set of labels. A bag can be considered a subset of the feature space, i.e. $X_i \subset \mathcal{X}$. A MIML classifier is a function $f : 2^{\mathcal{X}} \to 2^{\mathcal{Y}}$, i.e. it maps sets of feature vectors (bags) to sets of labels.

4.3 Methods

We formulate the species identification problem in the MIML framework as follows: audio recordings are bags, segments in the spectrogram are instances, and the set of species in a recording are a bag's label set. We propose a bag generator to convert an audio recording into a bag-of-instances representation, then use a MIML classifier to predict the set of species present in the recording. The bag generator transforms an audio signal into a spectrogram, applies noise reduction, segments the spectrogram into 2D regions, then associates each region with a feature vector. After applying the bag generator, any MIML classifier can be used.



(a) Before noise reduction



(b) After noise reduction

Figure 4.1: An example showing noise reduction in a recording wind and stream noise.

4.3.1 Bag Generator

This section describes the noise reduction, segmentation, and features in our proposed bag generator.

4.3.1.1 Preprocessing & Noise Reduction

Starting from a ten-second recording sampled at 16 kHz, we transform it into a spectrogram by dividing the input signal into frames of 512 samples with 50% overlap, then computing the 256-element magnitude spectrum of each frame using the Fast Fourier Transform (FFT) with a Hamming window. We will denote the elements of the spectrogram as $S(t, f)$, where t indexes a frame and f corresponds to frequency (note f indexes an element of the discrete spectrum, i.e. $f \in \{1, 2, \ldots, f_{max}\}$ where $f_{max} = 256$; it is not in units of Hz).

To reduce noise and improve the contrast of bird sound, we first normalize $S(t, f)$ to the range $[0, 1]$, then compute $S_1(t, f) = \sqrt{S(t, f)}$ for all elements of the spectrogram. Then we

apply two iterations of a whitening filter. The main idea is to estimate the frequency profile of noise from low energy frames, and then attenuate each row of the spectrogram according to this profile. The filter is:

1. Compute a quantity similar to the energy of each frame t, as $E(t) = \frac{1}{f_{max}} \sum_{f=1}^{f_{max}} S_1(t,f)^2$. Sort the frames by E. Let the noise frames $N = \{t : \text{frame } t \text{ is one of the lowest } 20\% \text{ energy frames}\}$.

2. For each frequency $f \in \{1, \ldots, f_{max}\}$, compute $P(f) = \sqrt{\epsilon + \sum_{t \in N} S(t,f)^2}$, where $\epsilon = 10^{-10}$ (we add ϵ to avoid dividing by 0).

3. For all (t,f) compute the noised reduced spectrogram as $S_2(t,f) = S_1(t,f)/P(f)$.

We will refer to the spectrogram resulting from two iterations of this process as $\hat{S}(t,f)$. Figure 4.1 shows a spectrogram before and after noise reduction.

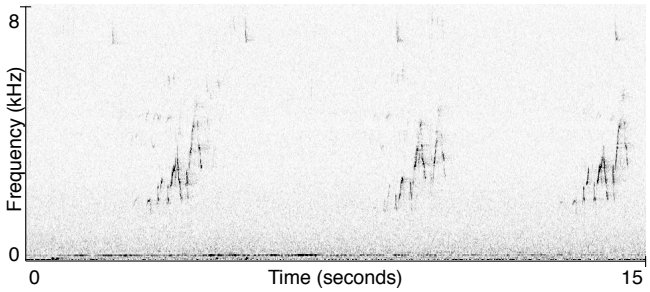
There are some differences in how we compute $\hat{S}(t,f)$ for segmentation and feature construction; for segmentation, we apply the whitening filter once, define $P(f)$ as $\frac{1}{|N|} \sum_{t \in N} S(t,f)$, and do not apply the square root in S_1.

4.3.1.2 Segmentation

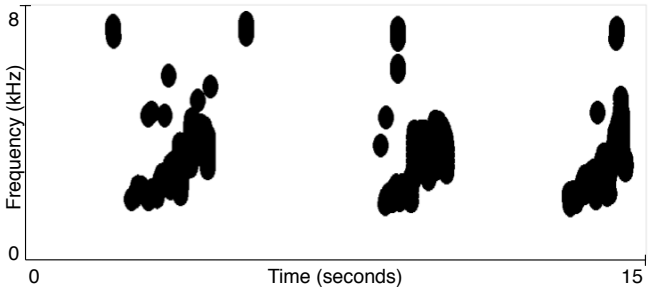
We use 2D time-frequency segmentation to separate syllables which may overlap in time. Rather than a 2D energy threshold [13], we use a supervised SISL classifier to label each pixel in a spectrogram as bird sound or noise [114]. To do so, we associate each pixel in a spectrogram with a feature vector that describes a rectangular patch surrounding it. So for a particular (t,f) in the spectrogram, we compute its feature vector $\mathbf{x}(t,f)$ as:

- The spectrum-bin index f.
- The value of the elements of the spectrogram in a rectangle surrounding (t,f), i.e. $\hat{S}(i,j), i \in [t - t_w, t + t_w], j \in [f - f_w, f + f_w]$, where in our setup $t_w = 6$ and $f_w = 12$ (these values are manually tuned for the sampling frequency and window size used in our study).
- The variance of $\hat{S}$ in the same rectangle as above.

In order to train the classifier used for segmentation, we manually annotate a collection of spectrograms as examples of correct segmentation (Fig. 4.2). The mask $M(t,f)$ for spectrogram



(a) A spectrogram in the segmentation training set.



(b) The corresponding manually generated time-frequency mask.

Figure 4.2: An example of the manual segmentation that is used to train our supervised segmentation algorithm.

$\hat{S}(t, f)$ is defined as $M(t, f) = 0$ (white) if element (t, f) is background noise and $M(t, f) = 1$ (black) if it is bird sound. Recall that a SISL classifier (such as a random forest) takes as training data a list of pairs $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$. We form these pairs by selecting 500,000 points (t_i, f_i) at random within the manually annotated spectrograms. These points are sampled so there are 90% negative examples and 10% positive examples. For each point we compute the feature vector as described above, $\mathbf{x}_i = \mathbf{x}(t_i, f_i)$. The label for each training example is $y_i = M(t_i, f_i)$ (i.e. we have a two-class problem with labels 0 and 1). Then we train a random forest classifier [14] with 40 trees on this data (a random forest is an ensemble of decision trees).

Given an input $\mathbf{x}$, a random forest generates a probability $P(y|\mathbf{x})$ for the instance to belong to each class y, which is the fraction of trees in the forest that vote for label y given input $\mathbf{x}$. We use the random forest to compute the probability for each pixel (t, f) in the spectrogram to be bird sound, i.e. $P(y = 1|\mathbf{x}(t, f))$. Then we smooth these probabilities by convolving with a Gaussian kernel to obtain $g(t, f) = P(y = 1|\mathbf{x}(t, f)) * K$, where K is Gaussian kernel with $\sigma = 3$ over a 17×17 box. Finally, we obtain a predicted segmentation mask $M(t, f)$ for a spectrogram by applying a threshold of $\theta = 0.2$ to $g(t, f)$ (chosen by visual inspection of results with varying θ). Figure 4.3b shows an example of the predicted segmentation for one recording.

The random forest classifier discussed in this section is only used for segmentation; it is not directly involved in predicting the set of species in a recording (a MIML classifier does that instead). The collection of manually annotated spectrograms used to train the segmentation algorithm is disjoint from recordings for which we predict sets of species, i.e., training and testing data are separate.

4.3.1.3 Features

To compute features for each segment, we first crop the mask and spectrogram to contain just that segment. Figure 4.3 shows how one segment is cropped. Figure 4.3a shows the original spectrogram, and Fig. 4.3b shows the binary mask produced by our segmentation algorithm. For the sake of illustration, we highlight one segment. Figures 4.3c and 4.3d show a cropped image of the mask and spectrogram based on the highlighted segment.

The mask and spectrogram are cropped to the minimum number of frames to contain the whole segment in time, but not cropped at all in frequency. Note in Fig. 4.3b, several other segments overlap in time with the highlighted segment. These overlapping segments are removed from the cropped mask (Fig. 4.3c). The portions of the cropped spectrogram that are

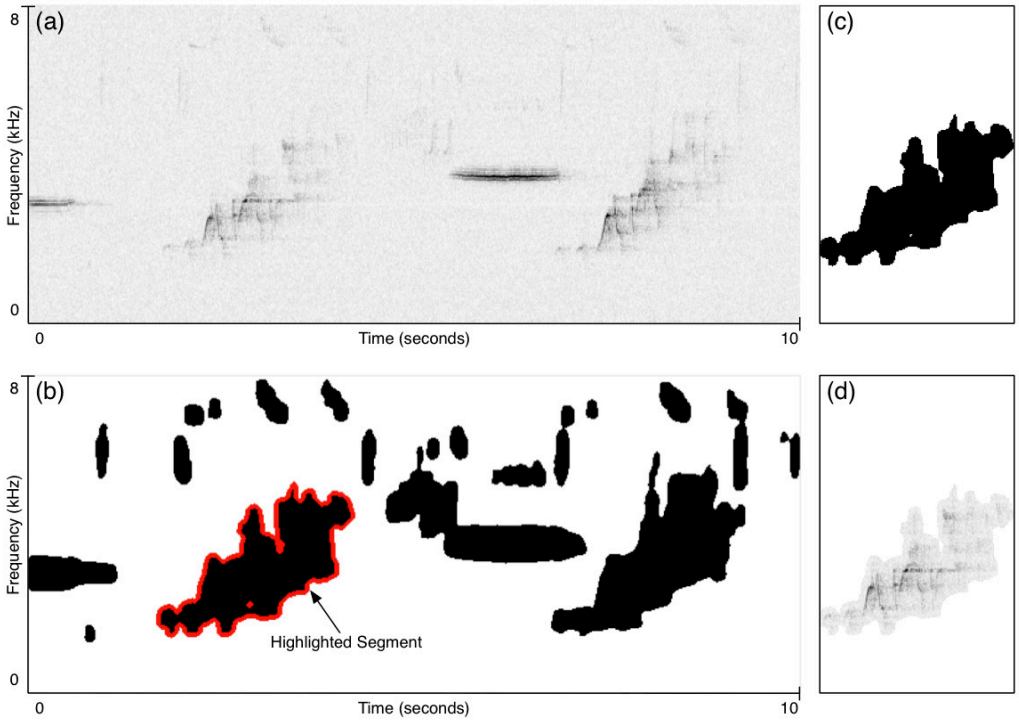


Figure 4.3: Extracting a syllable from the segmentation results. (a) The original spectrogram, (b) The binary mask generated by our segmentation algorithm. The highlighted segment will be further processed in this example. Note that several other segments overlap in time. (c) A cropped mask of the highlighted segment. (d) The masked and cropped spectrogram corresponding to the highlighted segment.

outside the mask are set to 0 (Fig. 4.3d). The purpose of these two changes is to eliminate any contribution in the segment features from other segments that overlap in time, or noise which is outside of the segment mask.

We use the following notation in describing the segment features: Let $M_c(t, f)$ be the cropped, binary mask for a segment and let $\hat{S}_c(t, f)$ be the cropped, noise-reduced spectrogram. Note that t ranges from 1 to the duration of the segment in frames, T.

Three types of features describe a segment: mask descriptors, profile statistics, and histogram of gradients (HOG) [42]. We depart from more commonly used audio features such as MFCCs because we are using 2D segmentation. The shape of the segment alone provides a lot of useful information, which the mask-based features capture. The profile statistics are similar to features that have previously been used for bioacoustics in noisy environments based on 2D segmentation [112].

Mask Descriptors The first set of features that we compute for a segment are based on only the mask (i.e. not the contents of the spectrogram), and describe the shape of the segment. These features are:

- *min-frequency* $= \min\{f : M_c(t, f) = 1\}$
- *max-frequency* $= \max\{f : M_c(t, f) = 1\}$
- *bandwidth* $=$ *max-frequency* $-$ *min-frequency*
- *duration* $= T$
- *area* $= \sum_{t,f} M_c(t, f)$
- *perimeter* $= \frac{1}{2} \times$(# of pixels in M_c such that at least one pixel in the surrounding 3×3 box is 1 and at least one pixel is 0)
- *non-compactness* $=$ *perimeter*^{2}/*area*
- *rectangularity* $=$ *area* / (*bandwidth* $\times$ *duration*)

Profile Statistics The next set of features that describe segments are based on statistical properties of the time and frequency profiles of the segment. To compute the time or frequency profile, we sum the columns or rows of the spectrogram. The time profile is $p_t(t) = \sum_f \hat{S}_c(t, f)$ and the frequency profile is $p_f(f) = \sum_t \hat{S}_c(t, f)$. We normalize the profiles to sum to 1, so they can be interpreted as probability mass functions. The normalized profile densities are $\hat{p}_t$ and $\hat{p}_f$. Two features measure the uniformity of these densities according the Gini index [69]:

- *freq-gini* $= 1 - \sum_f \hat{p}_f(f)^2$
- *time-gini* $= 1 - \sum_t \hat{p}_t(t)^2$

We obtain several more features by computing the kth central moments of the time and frequency profiles. However, because each segment may have a different duration, we compute these features in a re-scaled coordinate system where time goes from 0 to 1 over the duration of the segment, and frequency goes from 0 to 1.

- *freq-mean* $= \mu_f = \sum_{f=1}^{f_{max}} \hat{p}_f(f) \frac{f}{f_{max}}$
- *freq-variance* $= \sum_{f=1}^{f_{max}} \hat{p}_f(f)(\mu_f - \frac{f}{f_{max}})^2$
- *freq-skewness* $= \sum_{f=1}^{f_{max}} \hat{p}_f(f)(\mu_f - \frac{f}{f_{max}})^3$
- *freq-kurtosis* $= \sum_{f=1}^{f_{max}} \hat{p}_f(f)(\mu_f - \frac{f}{f_{max}})^4$
- *time-mean* $= \mu_t = \sum_{t=1}^{T} \hat{p}_t(t) \frac{t}{T}$
- *time-variance* $= \sum_{t=1}^{T} \hat{p}_t(t)(\mu_t - \frac{t}{T})^2$
- *time-skewness* $= \sum_{t=1}^{T} \hat{p}_t(t)(\mu_t - \frac{t}{T})^3$
- *time-kurtosis* $= \sum_{t=1}^{T} \hat{p}_t(t)(\mu_t - \frac{t}{T})^4$

In the same relative coordinate system, we compute the maxima of the time and frequency profiles:

- *freq-max* $= (\arg\max \hat{p}_f(f))/f_{max}$
- *time-max* $= (\arg\max \hat{p}_t(t))/T$

We also include the mean and standard deviation of the spectrogram within the masked region.

- *mask-mean* $= \mu_{tf} = \frac{1}{area} \sum_{tf} \hat{S}_c(t,f)$
- *mask-stddev* $= \sqrt{\frac{1}{area} \sum_{tf} (\mu_{tf} - \hat{S}_c(t,f))^2}$

Histogram of Gradients To further characterize the shape and texture of each segment, we include a HOG feature similar to the work of Dalal and Triggs [42]. As input, we take the cropped spectrogram and mask for a segment $\hat{S}_c(t,f)$ and $M_c(t,f)$. First, the spectrogram is blurred by convolving with a 7×7 Gaussian kernel G with $\sigma^2 = 4$ to obtain $S_b(t,f) = \hat{S}_c(t,f) * G$. The gradients at a point (t,f) are computed by convolving a Sobel kernel with with S_b, i.e. $\frac{d}{dx} S_b(t,f) = S_b(t,f) * D_x$ and $\frac{d}{dy} S_b(t,f) = S_b(t,f) * D_y$, where $D_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$ and $D_y = D_x^T$. Then, for each pixel of the spectrogram that is in the mask (i.e. $M_c(t,f) = 1$), we compute $\nabla S_b(t,f) = [\frac{d}{dx} S_b(t,f), \frac{d}{dy} S_b(t,f)]$. Only pixels such that $||\nabla S_b(t,f)||^2 \geqslant 0.01$ contribute to the histogram. The histogram consists of 16 bins evenly spaced over the range of angles $[0, 2\pi]$. The feature vector for a segment consists of the normalized count, for each bin, of the number of gradients belonging to that bin. Hence we obtain a 16 dimensional HOG feature for each segment.

Feature Rescaling All of the features described above are concatenated to form a single feature vector describing each segment. These features differ widely in the range of values they can have. This property of the features can bias distance-based classifiers such as MIML-kNN to place more weight on features with larger magnitudes. To prevent this bias, we rescale each feature independently to the range [0,1].

4.3.2 MIML Classifiers

Using our bag generator, we experimentally evaluate three MIML algorithms: MIMLSVM [171], MIMLRBF [167], and MIML-kNN [166]. These algorithms reduce the MIML problem to a single-instance multi-label problem by associating each bag with a bag-level feature, which aggregates information from the instances in the bag. Hence the MIML dataset $\{(X_i, Y_i)\}_{i=1}^m$ is

transformed into a single-instance multi-label dataset $\{(\mathbf{z}_i, Y_i)\}_{i=1}^m$ where $\mathbf{z}_i \in \mathbb{R}^d$ is the bag-level feature for bag i. Each algorithm constructs a different bag-level feature, but all use some form of bag-level distance measure. The maximal and average Hausdorff distances between two bags $X = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ and $X' = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ are defined as

$$\begin{aligned} D_H^{max}(X, X') &= \max\{\max_{\mathbf{a}\in X} \min_{\mathbf{b}\in X'} ||\mathbf{a}-\mathbf{b}||, \max_{\mathbf{b}\in X'} \min_{\mathbf{a}\in X} ||\mathbf{b}-\mathbf{a}||\} \\ D_H^{avg}(X, X') &= \Big(\sum_{\mathbf{a}\in X} \min_{\mathbf{b}\in X'} ||\mathbf{a}-\mathbf{b}|| \\ &\quad + \sum_{\mathbf{b}\in X'} \min_{\mathbf{a}\in X} ||\mathbf{b}-\mathbf{a}|| \Big) / (|X| + |X'|) \end{aligned}$$

MimlSvm applies k-medoids clustering to the training dataset of bags using D_H^{max}. This clustering produces k medoid bags $M_1, \ldots, M_k$. For each bag (X_i, Y_i), a bag-level feature is computed as $\mathbf{z}_i = (D_H^{max}(X_i, M_1), \ldots, D_H^{max}(X_i, M_k))$. The resulting multi-label classification problem is solved using the MlSvm algorithm, which consists of building one support vector machine (SVM) for each class.

MimlRbf runs k-medoids clustering once for each class using D_H^{avg} on the set of bags including that class as a label (the parameter k is different for each class). Concatenating the medoids obtained in each clustering, there are q medoid bags, $B_1, \ldots, B_q$. Then each bag X_i is associated with a feature $\mathbf{z}_i = (1, K(X_i, B_1), \ldots . K(X_i, B_q))$, where $K(X, X') = exp(-D_H^{avg}(X, X')^2/2\sigma^2)$. The resulting multi-label classification problem is solved using one linear model per class, trained by minimizing sum squared error.

Miml-kNN also assigns bag-level features, but does so using an approach inspired by nearest-neighbors rather than clustering. For each training bag X_i, Miml-kNN finds its k nearest neighbors, and k' citers (other bags that consider X_i to be one of their k' nearest neighbors), using D_H^{avg}. Then each bag X_i is associated with a bag-level feature vector $\mathbf{z}_i = (t_1, \ldots, t_c)$, where t_j is the number of bags in the neighbors and citers of X_i that include class j in their label set. The resulting multi-label problem is solved using the same approach as MimlRbf.

Table 4.1: The number of ten-second recordings containing each species in our labeled dataset.

Code	Name	#
PSFL	Pacific-slope Flycatcher *(Empidonax difficilis)*	165
HAFL	Hammond's Flycatcher *(Empidonax hammondii)*	103
OSFL	Olive-sided Flycatcher *(Contopus cooperi)*	90
HETH	Hermit Thrush *(Catharus guttatus)*	15
VATH	Varied Thrush *(Ixoreus naevius)*	89
SWTH	Swainson's Thrush *(Catharus ustulatus)*	79
GCKI	Golden-crowned Kinglet *(Regulus satrapa)*	197
PAWR	Pacific Wren *(Troglodytes pacificus)*	109
RBNU	Red-breasted Nuthatch *(Sitta canadensis)*	82
DEJU	Dark-eyed Junco *(Junco hyemalis)*	20
CBCH	Chestnut-backed Chickadee *(Poecile rufescens)*	117
HEWA	Hermit Warbler *(Setophaga occidentalis)*	63
WETA	Western Tanager *(Piranga ludoviciana)*	46

4.4 Experiments

We apply the proposed methods to field-collected audio from the H. J. Andrews (HJA) Experimental Forest. These experiments demonstrate that our methods accurately predict the set of species present in an unattended acoustic monitoring scenario.

4.4.1 Data Collection & Labeling

To collect audio in HJA, we use 13 Wildlife Acoustics Song Meter SM1 recording devices. These devices have two omnidirectional microphones enclosed in wind shields protruding from a weather resistant enclosure that houses batteries, a computer, and 32 Gb flash-memory for data storage.

The audio is recorded at 16 kHz. The result of applying the FFT is a spectrogram with frequencies from 0–8 kHz. This range is sufficient to capture most bird sounds in HJA. For example, the Hermit Warbler is one of the highest pitched species in HJA, and is generally below 8 kHz [117]. It is possible that some bird sounds are omitted due to this sampling frequency, but the proposed methods still work well for the species that we identified.

In order to train and evaluate algorithms to predict which species of birds are present in a

recording, it is necessary to have some labeled examples. We have months of audio in total, so it would not be feasible to manually label all of it. Accordingly, we focus on a representative sample of 548 ten-second recordings from six sites, all within the range of 5:00 am to 5:20 am (birds are highly active at this time of day), on 5/31/2009. Many of the recordings include multiple bird species vocalizing simultaneously. We manually identified the set of species that are present in each ten-second recording.

There are 13 bird species in the recordings examined (Table 4.1). Each recording contains between 1 and 5 species. There are 2.144 species per recording on average.

For the purpose of MIML experiments, we assume that recordings that do not contain any bird sounds can be detected during segmentation, hence we only include recordings that contain at least 1 species vocalizing. We evaluated segmentation on recordings that do not contain bird sound in prior work [114].

Note that a small fraction of the recordings contain only a single species (but still multiple syllables). Two out of the 13 species have recordings of this kind. It is not necessary for MIML to have multiple labels associated with every bag in the training set. Bags with a single label provide less ambiguous information, and should therefore be expected to improve accuracy.

There are 10,232 instances (audio segments) in our dataset. It is considerably more laborious to label the instances than the 548 bags. For the purpose of comparison to SISL methods, we have manually labeled 4998 of these instances. Of the remaining 5,234 unlabeled instances, a substantial fraction are segmentation errors or noise, or faint sounds that are very difficult for a human to identify. These instance labels are only used for evaluation of SISL; they are not used by the MIML algorithms.

In addition to the 548 ten-second recordings that we labeled with species sets, we also manually segmented 625 disjoint fifteen-second recordings that are used as examples to train the segmentation algorithm. These recordings are selected from 13 sites, over a period from 5/2/2009 to 7/4/2009, with some examples from each hour of the day. Within these 625 examples, 334 contain some bird sound, and 291 contain only noise.

It is not crucial that the training recordings be ten or fifteen seconds. We can provide some intuition for the choice of duration. Increasing the duration of the recordings used for training the segmentation algorithm reduces the number of syllables in the training data that are cut off by the boundary of the recording. However, as the duration of recordings used for MIML is increased, it becomes more likely for a bag to include all of the species at the recording site. In the extreme, a bag is labeled with every species, in which case no learning is possible because

the labels are completely ambiguous.

All of the data collection sites are within 1 km of a stream. Hence, all recordings contain some stream noise, as well as wind or insects in some cases.

4.4.2 Evaluation

Cross-Validation We use five-fold cross-validation to evaluate each MIML algorithm on the collection of 548 species-labeled recordings. The recordings/bags are randomly partitioned into five disjoint sets (each set contains some examples from every species). For each fold, four of the sets are used as training data for the MIML algorithms, and the remaining set is used for testing. Test performance is aggregated over the five folds.

Because we use data from six sites over a 20-minute interval of time, it is likely that the MIML classifier is trained and tested on vocalizations from the same individual birds. We expect that prediction accuracy would decrease in an experiment where the classifier is applied to individuals that do not appear in the training set.

Accuracy Measures Several measures common in multi-label and MIML experiments characterize the accuracy of each algorithm, namely hamming loss, rank loss, one-error, coverage [172], and micro-AUC [50]. A MIML classifier outputs a set of classes, but many implementations first output a score for each class, which is compared to a threshold to obtain the set. Several of the accuracy measures use these scores. We denote the score for class j given by a MIML classifier f on input bag X as $f_j(X)$. The set of predicted labels which is obtained from the scores is denoted $f(X)$. Also let $I[\cdot]$ denote the indicator function. Recall that the number of classes (species) is c, and the number of bags is n. The accuracy measures are defined as follows:

Hamming Loss does not rely on the scores for each class, but instead directly evaluates the predicted set. It is the number of false positives and false negatives, averaged over the number of classes and bags,

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

Rank loss captures the number of label pairs that are incorrectly ordered by the scores of the MIML classifier (i.e. classes that are in the 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

$$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{|Y_i||\bar{Y}_i|}\sum_{j\in Y_i,k\in\bar{Y}_i} I[f_j(X_i) \leqslant f_k(X_i)]$$

One-error is the fraction of bags for which the top scoring label is not in the true label set,

$$\frac{1}{n}\sum_{i=1}^{n} I[\big(\arg\max_{j\in\mathcal{Y}} f_j(X_i)\big) \notin Y_i]$$

The scores for all classes can be ranked, so that rank 1 is the most likely to be present (highest score), rank 2 is the next most likely, and rank c is the least likely. We denote the rank of class j given input bag X as $rank(X, j)$. Coverage measures the how far down the ranking one must go to get all of the true labels,

$$\frac{1}{n}\sum_{i=1}^{n}\max_{j\in Y_i}\{rank(X_i, j) - 1\}$$

MimlSvm, MimlRbf, and Miml-kNN output signed scores for each class, with a positive score indicating a class is present, and a negative score indicating it is absent. To compute hamming loss, we use a threshold of 0. However, varying this threshold can be used to control the tradeoff between predicting species which are not present (i.e. false positives), or failing to detect species which are present (i.e. false negatives). The Receiver Operating Characteristic (ROC) curve captures this tradeoff. Let the predicted label set for a bag X_i using a threshold t be $f(X_i, t)$. Define true/false positives/negatives as

$$TP = \sum_{i=1\ldots n, j\in Y_i} I[j \in f(X_i, t)], \quad FP = \sum_{i=1\ldots n, j\in\bar{Y}_i} I[j \in f(X_i, t)]$$
$$TN = \sum_{i=1\ldots n, j\in\bar{Y}_i} I[j \notin f(X_i, t)], \quad FN = \sum_{i=1\ldots n, j\in Y_i} I[j \notin f(X_i, t)]$$

The true positive rate is $TPR = TP/(TP + FN)$ and the false positive rate is $FPR = FP/(FP + TN)$. Each point on the ROC curve (Fig. 4.4) corresponds to a pair (TPR, FPR) for one threshold. The area under this ROC curve is called micro-AUC (in contrast with macro-AUC, which is the average AUC of the separate ROC curves for each class) [94].

Table 4.2: Accuracy measures for MIML classifiers and baselines (– indicates the result cannot be calculated).

Algorithm	Hamming Loss ↓	Rank Loss ↓	One-Error ↓	Coverage ↓	micro-AUC ↑
MimlSvm	0.054	0.033	0.067	1.844	0.966
Miml-kNN	**0.039**	**0.019**	0.036	**1.589**	0.962
MimlRbf	0.049	0.022	0.034	1.632	**0.978**
Non-Informative	0.165	0.5	0.8351	8.068	0.5
Frequency Order	–	0.318	0.698	5.901	–
SISL Random Forest	0.125	0.050	0.084	2.201	0.949
SISL Random Forest Filtered	0.049	0.023	**0.022**	1.708	0.974

Parameter Tuning Each MIML algorithm has several parameters that can be tuned to improve accuracy. We evaluate each algorithm over all combinations of parameter values in a range, and report results corresponding to the parameter setting of each algorithm that minimizes hamming loss. The parameters and ranges are:

- For MimlSvm, the parameters are (C, γ, r). The parameter C controls SVM regularization. The SVM uses a Gaussian kernel $k(\mathbf{x}, \mathbf{y}) = exp(-\gamma||\mathbf{x} - \mathbf{y}||^2)$. The parameter k for k-medoids clustering is set to nr, where n is the number of bags. We evaluate all combinations of $(C, \gamma, r) \in \{10^{-2}, 10^{-1}, 10^{0}, 10^{1}\}^2 \times \{0.2, 0.4, 0.6, 0.8\}$.

- For MimlRbf, the parameters are (r, μ). k-medoids clustering repeats once for each class; in the clustering for class i, k is set to $\nu_i r$, where ν_i is the number of bags including label i. MimlRbf constructs bag-level features by applying a function $K(X, X') = exp(-D_H^{avg}(X, X')^2/2\sigma^2)$. The parameter σ in this expression is set to μ times the average Hausdroff distance between clusters [167]. We evaluate all combinations of $(r, \mu) \in \{10^{-2}, 10^{-1}, 10^{0}, 10^{1}\}^2$.

- For Miml-kNN, we vary the number of neighbors k and the number of citers k' over $(k, k') \in \{5, 10, 20, 30\}^2$.

4.5 Results

Table 4.2 lists the accuracy measures for each MIML algorithm. To better interpret these results, we discuss the ranges of values the accuracy measures can take, and compare to values

Table 4.3: Example predictions with MIML-kNN.

Ground Truth	Predicted Labels
PAWR,PSFL	GCKI,PAWR,PSFL
VATH,SWTH	VATH,HEWA,SWTH
OSFL,CBCH	GCKI,OSFL,CBCH
CBCH	GCKI,CBCH
HAFL	HAFL
VATH,HEWA	VATH,HEWA
GCKI,PSFL,RBNU,DEJU	GCKI,PSFL
GCKI,PAWR,PSFL	GCKI
GCKI,OSFL	GCKI,OSFL
GCKI,PAWR,PSFL	GCKI,PAWR,PSFL
SWTH	SWTH
VATH,HEWA,SWTH	VATH,HEWA,SWTH
GCKI,OSFL,HETH	GCKI
GCKI,OSFL	GCKI,OSFL
GCKI,PAWR,PSFL	GCKI,PAWR
SWTH	
GCKI,PAWR,PSFL	GCKI,PSFL
GCKI,PSFL,OSFL	GCKI,PSFL
HAFL	HAFL
CBCH,SWTH	CBCH,SWTH

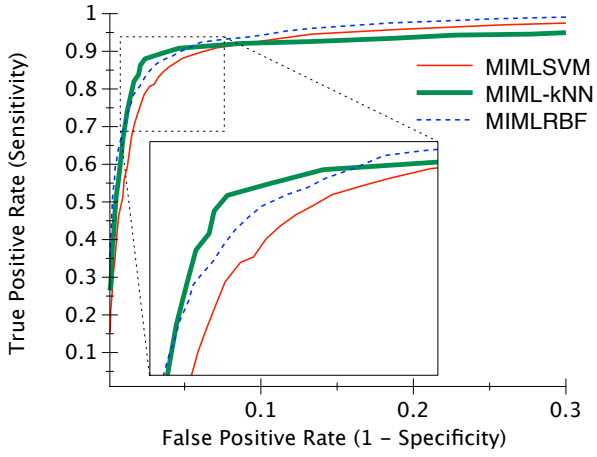


Figure 4.4: ROC curves for each algorithm.

for baseline classifiers.

Hamming loss, rank loss, and one-error have values in the range $[0, 1]$ with 0 corresponding to perfect prediction. Let the average number of labels per bag be $m = \frac{1}{n}\sum_{i=1}^{n}|Y_i|$. Then the best possible coverage is $m - 1$, and the worst possible coverage is $c - 1$ (for our dataset, this gives the range $[1.144, 12]$). However, in order to achieve the worst possible values for these measures, it is necessary to make predictions that are worse than random.

To obtain a baseline for hamming loss, consider a non-informative classifier that always predicts the empty set. The hamming loss is $\frac{m}{c} = 0.1649$, because each label in the true label set is a false-negative. The other measures are based on class ranks, so consider a non-informative classifier that outputs uniformly random scores for each class, or equivalently, ranks classes in a random order. The probability that the top-scoring class will be one of the $|Y_i|$ labels for bag i is $\frac{|Y_i|}{c}$, so the expected one-error is $1 - \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i|}{c} = 1 - \frac{m}{c} = 0.8351$. Because $P(f_j(X_i) \leqslant f_k(X_i)) = \frac{1}{2}$, the expected rank loss is $\frac{1}{2}$. The AUC for a random classifier is $\frac{1}{2}$ as well [12]. We approximate the expected coverage for a non-informative classifier by averaging the coverage for 10,000 random orders.

We also compute the rank loss, one-error, and coverage for a classifier that ignores its input, and outputs the ranking from most frequent class to least frequent, which is a stronger baseline than random ranking (Table 4.2).

All of the MIML classifiers are closer to perfect prediction than to non-informative or frequency order baselines. For example, the hamming loss for MIML-kNN is 4.23 times lower than non-informative. With a rank loss of 0.019, MIML-kNN is 26.31 times less likely to incorrectly rank a present/absent species pair than a random classifier. A one-error of 0.034 means that if we only predict the highest scoring species in each recording, it will truly be present 96.6% of the time. MIML-kNN achieves a hamming loss of 0.039, which is equivalent to a true positive/negative rate of 96.1% (the fraction of true positive/negatives is 1 - hamming loss). To give a concrete view of the predictions, we show results for 20 randomly selected recordings using MIML-kNN in Table 4.3.

Recent work [106] has highlighted the importance of accounting for imperfect detectability of species in wildlife surveys. Due to the massive amount of survey time enabled by continuous recordings, our proposed methods can help to reduce false negatives typical of manual bird surveys. The ROC curves (Fig. 4.4) show that we can set a threshold which achieves a low false positive rate, while still retaining a relatively high true positive rate, thus meeting critical assumptions for occupancy analysis.

Comparison to SISL It is difficult to make a direct comparison between MIML and SISL, because MIML and SISL algorithms make different types of predictions, and are evaluated according to different performance measures. We compare to a model based on a random forest SISL classifier because the random forest achieves high accuracy in many domains, and has only one parameter (the number of trees), to which it is not very sensitive. Using the 4998 labeled instances in the dataset, we train a SISL random forest with 100 trees. We use the same folds for 5-fold cross validation as the MIML algorithms. For each fold, the labeled instances in four of the sets are used to train a random forest, then the instances in the remaining set are used to compute MIML performance measures.

We need to compute bag-level outputs to evaluate the random forest using MIML performance measures. To do so, we compute the probabilities for every instance in a bag to belong to each class, then define the bag level score for each class as the maximum instance probability for that class, i.e. the bag-level score for class j given input bag X is $f_j(X) = \max_{\mathbf{x} \in X} P(j|\mathbf{x})$. This formulation of the bag-level model is similar to MIML algorithms including M^{3}MIML [168], D-MimlSvm [172] and Tmiml [81]. We compute the hamming loss for the random forest using a threshold of 0.5 on the bag-level scores. The other performance measures are computed directly from the scores.

This SISL-trained model is worse in every performance measure than the MIML algorithms (Table 4.2; SISL Random Forest). The bag-level scores are computed using all of the instances in the bag, including the unlabeled instances which are more likely to correspond to noise that is mislabeled as bird sound in the segmentation stage. Such instances bias the bag-level scores to generate many false positives (with a threshold of 0.5 there are 740 false positives and 148 false negatives). We can improve the results for the random forest by filtering out all of the unlabeled instances so they do not influence the bag-level scores (Table 4.2; SISL Random Forest Filtered). The MIML algorithms in this study do not require this filtering to produce accurate results because they do not depend on instance-level predictions (this is not true of all MIML algorithms). However filtering would improve the performance of the MIML algorithms.

Even when we give the SISL model the advantages of having instance labels, and of filtering out the unlabeled instances, the best MIML results are better than the filtered SISL results in all measures except for one-error. This result suggests that it is non-trivial to incorporate instance labels into a bag-level model.

4.6 Conclusion & Future Work

We formulate the problem of detecting the set of bird species present in an audio recording using the MIML framework, and propose a method to transform an audio recording into a representation suitable for using with MIML algorithms. Using data collected in the field with omnidirectional microphones, we showed that the proposed methods achieve high accuracy.

This work is a step toward automatic unattended acoustic surveys of bird populations. In future work, we seek to classify all bird sounds in a much larger collection of audio (over 4 terabytes), representing two years of recordings in the field. This will effectively generate a presence/absence population survey at the sites where we have deployed recording devices. In contrast with manual surveys, automated acoustic surveys can provide high temporal resolution over the long term. For example, it would not be reasonable for a person to count birds once per minute, 24 hours a day, for three months, but we aim to obtain similar results with acoustic surveys. Such data is likely to provide new insights into bird behavior, and their interaction with the environment [133].

MIML classifiers can only predict labels that appear in their training data, and cannot detect when something does not belong to one of the training classes. Hence it is not clear how to handle unexpected sounds. Due to the high-noise environment, and birds vocalizing far from the microphone, it is often difficult for a human labeler to determine all of the species present in a recording. Consequently, some of the segments/instances may come from species that are not present in the training label set. Furthermore, some of the instances are segmentation errors capturing noise rather than bird sound. Further work is needed on classes not present in the training data, incomplete label sets, and noise instances.

One may wish to predict the species of each individual segment/instance, rather than just the set of species in a recording. If individually labeled segments are available for training data, this is the standard SISL supervised classification problem. However, we instead focus on the situation where it is difficult or expensive to obtain such labels. Learning to predict instance labels from MIML training data is a different problem, known as instance annotation, which has received little study so far. One might also wonder if the accuracy of MIML predictions could be improved by including individually labeled instances in the training data (e.g., recordings containing only a single species and syllable). This problem of mixed-granularity training has also received little study. In prior work on MIML, this issue has been handled by using an unmodified MIML algorithm with a bag containing a single instance [148].

Although we focus on birds, MIML may also be applicable to analysis of other bioacoustic signals from animals including grasshoppers [33], crickets, frogs [13], and marine mammals [112], and computational acoustic scene analysis in general. Aside from its ecological applications, this work broadens the scope of MIML domains from text and images to include audio.

Chapter 5: Instance Annotation for Multi-Instance Multi-Label Learning[1]

5.1 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) [49]. The original formulation of MIL concerns problems where bags are associated with a single binary label. Zhou and Zhang [171] 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 [97, 132, 165, 171, 172], and video [157] domains. Recently, Wang and Zhou [151] analyzed PAC-learnability in MIML.

Learning to predict instance labels from MIML training data is a useful problem that has received little study [170]. 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

[1] "Instance Annotation for Multi-Instance Multi-Label Learning." Forrest Briggs, Xiaoli Z. Fern, Raviv Raich, Qi Lou. ACM Transactions on Knowledge Discovery from Data, TKDD, 2012.

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 chapter, we examine two different aggregation models, which are equivalent to defining a "support instance" for each bag. Therefore, we name our methods Support Instance Machines (SIM).

In this chapter, we make the following contributions:[2]

- We propose a regularized rank-loss objective for instance annotation that can be instantiated with different aggregation models (Sec. 5.4.1).

- The rank-loss objective is non-convex. 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) [164].

- 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 sub-gradient 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 (Sec. 5.4.3.4).

[2] A preliminary version of this work appeared in [16]. We present new materials in Sec. 5.4.3, 5.4.4, and the two appendices. Section 6.5 also contains additional experiments including a new dataset and a comparison to the M^{3}MIML algorithm.

- Experiments show that SIM with rank loss achieves higher accuracy than Hamming loss or ambiguous loss (a comparable state of the art approach [39, 40]), 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 (Sec. 5.5.2.4).

- We suggest a method of extending our proposed classifiers from linear to non-linear using a fast approximate kernel trick [119]. Experiments show that this method often improves accuracy significantly, while still achieving linear runtime in the number of bags and instances (Sec. 5.4.4).

- 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) (Sec. 6.3).

5.2 Problem Statement

In this section we formalize the instance annotation problem and contrast it with several related problems. Table 6.2 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 $f_{IA} : \mathcal{X} \rightarrow \mathcal{Y}$ 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 [171] and many others, where the goal is to learn a bag-level classifier $F_{MIML} : 2^{\mathcal{X}} \rightarrow 2^{\mathcal{Y}}$. Nonetheless, the design principles of many traditional MIML

Table 5.1: 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 sub-gradient 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 sub-gradient descent, iteration t of CCCP
β_i	$\frac{1}{n\lvert Y_i\rvert\lvert\bar{Y}_i\rvert}$
$\sum_{ijk}$	$\sum_{i=1}^{n} \sum_{j \in Y_i} \sum_{k \notin Y_i}$

algorithms can be used to learn instance-level classification models, which is the approach we take in this chapter.

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) [40, 80] 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

$f_{ALC} : \mathcal{X} \rightarrow \mathcal{Y}$. A MIML instance annotation problem can be transformed into a ALC problem by 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.

5.3 Background

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

5.3.1 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 [168], and D-MimlSvm [172]. The following formulation approximates this assumption. Let $f_j(\mathbf{x}) : \mathcal{X} \rightarrow \mathbb{R}$ be a function which 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 $F_j(X) = \max_{\mathbf{x} \in X} f_j(\mathbf{x}) \geqslant f_j(\mathbf{x}^*) > 0$. 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.

5.3.2 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 [172]. Hamming Loss is

the number of false positives and false negatives, averaged over all classes and bags,

$$\frac{1}{nc}\sum_{i=1}^{n}\sum_{j=1}^{c} I[j \in F(X_i), j \notin Y_i] + I[j \notin F(X_i), j \in Y_i] \tag{5.1}$$

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

$$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{|Y_i||\bar{Y}_i|}\sum_{j\in Y_i, k\in\bar{Y}_i} I[F_j(X_i) \leqslant F_k(X_i)] \tag{5.2}$$

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 [172] and M^{3}MIML [168] optimize variations of the following loss function (with different regularization terms)

$$\frac{1}{nc}\sum_{i=1}^{n}\sum_{j=1}^{c}\max\{0, 1 - Y_{ij}F_j(X_i)\} \tag{5.3}$$

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

The hinged Hamming-loss objective (5.3) 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 [55]. However, we are not aware of any MIML algorithms that learn an instance-level model by minimizing rank loss (i.e. rank loss has only been used as a performance measure in MIML, not as an objective).

5.4 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 non-convex. One is a heuristic, and the other is based on CCCP, however both exploit the support instance structure in the optimization problem.

5.4.1 Rank-Loss Objective

Because we are learning from a 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 a bag-level loss function, which is a regularized surrogate for rank loss (5.2):

$$h_{RL}(\mathbf{W}) = \frac{\lambda}{2}||\mathbf{W}||^2 + \sum_{i=1}^{n} \sum_{j \in Y_i} \sum_{k \notin Y_i} \frac{1}{n|Y_i||\bar{Y}_i|} \max\{0, 1 - \big(F_j(X_i) - F_k(X_i)\big)\} \tag{5.4}$$

To shorten our notation, let $\beta_i = \frac{1}{n|Y_i||\bar{Y}_i|}$ 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

$$h_{RL}(\mathbf{W}) = \frac{\lambda}{2}||\mathbf{W}||^2 + \sum_{ijk} \beta_i \max\{0, 1 - \big(F_j(X_i) - F_k(X_i)\big)\} \tag{5.5}$$

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 sub-gradient descent and CCCP optimization methods, which we discuss in Sec. 5.4.3.

5.4.2 Aggregation Models and Support Instances

The objective (5.4) 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 [168, 172], with a linear instance classifier is

$$F_j(X_i) = \max_{\mathbf{x}_{iq} \in X_i} f_j(\mathbf{x}_{iq}) = \max_{\mathbf{x}_{iq} \in X_i} \mathbf{w}_j \cdot \mathbf{x}_{iq} \tag{5.6}$$

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

$$\hat{\mathbf{x}}_{ij}(\mathbf{W}) = \arg\max_{\mathbf{x}_{iq} \in X_i} \mathbf{w}_j \cdot \mathbf{x}_{iq} \tag{5.7}$$

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:

$$F_j(X_i) = \sum_{\mathbf{x}_{iq} \in X_i} \alpha_{iq}^j f_j(\mathbf{x}_{iq}) = \sum_{\mathbf{x}_{iq} \in X_i} \alpha_{iq}^j \mathbf{w}_j \cdot \mathbf{x}_{iq} \tag{5.8}$$

The weights are defined according to a softmax rule,

$$\alpha_{iq}^j = \frac{e^{\mathbf{w}_j \cdot \mathbf{x}_{iq}}}{\sum_{\mathbf{x}' \in X_i} e^{\mathbf{w}_j \cdot \mathbf{x}'}} \tag{5.9}$$

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

$$\hat{\mathbf{x}}_{ij}(\mathbf{W}) = \sum_{\mathbf{x}_{iq} \in X_i} \alpha_{iq}^j \mathbf{x}_{iq} \tag{5.10}$$

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

$$\hat{h}_{RL}(\mathbf{W}) = \frac{\lambda}{2}||\mathbf{W}||^2 + \sum_{ijk} \beta_i \max\{0, 1 + \mathbf{w}_k \cdot \hat{\mathbf{x}}_{ik}(\mathbf{W}) - \mathbf{w}_j \cdot \hat{\mathbf{x}}_{ij}(\mathbf{W})\} \tag{5.11}$$

5.4.3 Optimization Methods for Rank-loss Support Instance Machines

SIM could be instantiated with any aggregation model that fits the pattern of (5.11). 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 non-convex. We propose two optimization strategies to handle this non-convex 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.

5.4.3.1 Constrained Convex Concave Procedure (CCCP)

The bag-level surrogate loss functions that arise in multi-instance learning are often non-convex, but can sometimes be manipulated into a difference-of-convex (DC) form which allows the use of CCCP [137, 164]. 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

$$\min_{\mathbf{x}} \quad f_0(\mathbf{x}) - g_0(\mathbf{x}) \tag{5.12}$$

$$s.t. \quad f_i(\mathbf{x}) - g_i(\mathbf{x}) \leqslant 0, i = 1, \ldots, k \tag{5.13}$$

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

$$\min_{\mathbf{x}} \quad f_0(\mathbf{x}) - \left[g_0(\mathbf{x}^{(t)}) + \nabla g_0(\mathbf{x}^{(t)}) \cdot (\mathbf{x} - \mathbf{x}^{(t)}) \right] \tag{5.14}$$

$$s.t. \quad f_i(\mathbf{x}) - \left[g_i(\mathbf{x}^{(t)}) + \nabla g_i(\mathbf{x}^{(t)}) \cdot (\mathbf{x} - \mathbf{x}^{(t)}) \right] \leqslant 0, \quad i = 1, \ldots, k \tag{5.15}$$

If g_i is non-differentiable, a sub-gradient $\mathbf{u} \in \partial g_i(\mathbf{x}^{(t)})$ 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.

5.4.3.2 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 (5.11) is not a DC function, but we can still use CCCP with a simple transformation of the problem. The unconstrained problem (5.11) is equivalent to a constrained problem

$$\min_{\mathbf{W},\xi} \quad \frac{\lambda}{2}||\mathbf{W}||^2 + \sum_{ijk} \beta_i \xi_{ijk} \tag{5.16}$$

$$s.t. \quad F_j(X_i) - F_k(X_i) \geqslant 1 - \xi_{ijk}, \text{ for } i = 1, \ldots, n, j \in Y_i, k \notin Y_i \tag{5.17}$$

$$\xi_{ijk} \geqslant 0 \tag{5.18}$$

The constraint (5.17) 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

$$\underbrace{1 - \xi_{ijk} + \max_{\mathbf{x} \in X_i} \mathbf{w}_k \cdot \mathbf{x}}_{f_{ijk}} - \underbrace{\max_{\mathbf{x} \in X_i} \mathbf{w}_j \cdot \mathbf{x}}_{g_{ijk}} \leqslant 0 \tag{5.19}$$

We define the support instances at iteration t of CCCP as

$$\hat{\mathbf{x}}_{ij}^{(t)} = \hat{\mathbf{x}}_{ij}(\mathbf{W}^{(t)}) \tag{5.20}$$

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

$$1 - \xi_{ijk} + \max_{\mathbf{x} \in X_i} \mathbf{w}_k \cdot \mathbf{x} - \mathbf{w}_j \cdot \hat{\mathbf{x}}_{ij}^{(t)} \leqslant 0 \tag{5.21}$$

Hence CCCP solves the following sequence of convex problems

$$\min_{\mathbf{W},\xi} \quad \frac{\lambda}{2}||\mathbf{W}||^2 + \sum_{ijk} \beta_i \xi_{ijk} \tag{5.22}$$

$$s.t. \quad 1 - \xi_{ijk} + \max_{\mathbf{x} \in X_i} \mathbf{w}_k \cdot \mathbf{x} - \mathbf{w}_j \cdot \hat{\mathbf{x}}_{ij}^{(t)} \leqslant 0, \text{ for } i = 1, \ldots, n, j \in Y_i, k \notin Y_i \tag{5.23}$$

$$\xi_{ijk} \geqslant 0 \tag{5.24}$$

The above 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 sub-gradient descent)

$$h^{(t)}_{RL,cccp}(\mathbf{W}) = \frac{\lambda}{2}||\mathbf{W}||^2 + \sum_{ijk} \beta_i \max\{0, 1 + \max_{\mathbf{x} \in X_i} \mathbf{w}_k \cdot \mathbf{x} - \mathbf{w}_j \cdot \hat{\mathbf{x}}^{(t)}_{ij}\} \tag{5.25}$$

We discuss methods for solving (5.25) in Sec. 5.4.3.4. In summary, the trick we use to efficiently solve the original non-convex 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.

5.4.3.3 Heuristic for Non-Convex Aggregation Models

If $F_j(X_i)$ is not a convex function of $\mathbf{W}$, then the constraint (5.17) 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

$$h^{(t)}_{RL,h}(\mathbf{W}) = \frac{\lambda}{2}||\mathbf{W}||^2 + \sum_{ijk} \beta_i \max\{0, 1 + \mathbf{w}_k \cdot \hat{\mathbf{x}}^{(t)}_{ik} - \mathbf{w}_j \cdot \hat{\mathbf{x}}^{(t)}_{ij}\} \tag{5.26}$$

In contrast with the CCCP algorithm, this heuristic is not proven to decrease the objective (5.4) 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.

5.4.3.4 Sub-gradient Descent

To solve the convex problem in each step of CCCP or the heuristic [i.e. (5.25) or (5.26)], we use a sub-gradient descent method similar to Pegasos, an algorithm for training linear two-class SVMs [130]. The Pegasos algorithm is based on a general framework for optimizing regularized convex objectives [129]. This framework can be applied to convex optimization problems in the

form:

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

ALGORITHM 1: Projected Sub-gradient with Pegasos Learning Rate

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

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. $S = \{\mathbf{W} : ||\mathbf{W}|| \leqslant r\}$, 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 $h^{(t)}_{RL,cccp}(\mathbf{W})$ (5.25) or $h^{(t)}_{RL,h}(\mathbf{W})$ (5.26) respectively. We will show how to apply sub-gradient descent to these objectives. We will start by summarizing relevant notation and stating a sub-gradient 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 sub-gradient 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 sub-gradient descent, and q indicates the class. The full sub-gradient 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 sub-gradient 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 sub-gradient descent the sub-gradients

depend only on $\hat{\mathbf{x}}_{ij}^{(t)}$. The sub-gradient for the heuristic is

$$\mathbf{v}_{RL,h}^{(t,\tau),q} = \lambda \mathbf{w}_q^{(t,\tau)} + \sum_{ijk} \beta_i I[1 + \mathbf{w}_k^{(t,\tau)} \cdot \hat{\mathbf{x}}_{ik}^{(t)} - \mathbf{w}_j^{(t,\tau)} \cdot \hat{\mathbf{x}}_{ij}^{(t)} > 0] \begin{cases} \hat{\mathbf{x}}_{iq}^{(t)} & \text{if } q = k \\ -\hat{\mathbf{x}}_{iq}^{(t)} & \text{if } q = j \\ 0 & \text{otherwise} \end{cases} \tag{5.27}$$

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

$$\hat{\mathbf{x}}_{ik}^{(t,\tau)} = \hat{\mathbf{x}}_{ik}(\mathbf{W}^{(t,\tau)}) \tag{5.28}$$

This notation indicates the support instance computed from the weights at iteration τ of sub-gradient 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 sub-gradient for CCCP at inner iteration τ is

$$\mathbf{v}_{RL,cccp}^{(t,\tau),q} = \lambda \mathbf{w}_q^{(t,\tau)} + \sum_{ijk} \beta_i I[1 + \mathbf{w}_k^{(t,\tau)} \cdot \hat{\mathbf{x}}_{ik}^{(t,\tau)} - \mathbf{w}_j^{(t,\tau)} \cdot \hat{\mathbf{x}}_{ij}^{(t)} > 0] \begin{cases} \hat{\mathbf{x}}_{iq}^{(t,\tau)} & \text{if } q = k \\ -\hat{\mathbf{x}}_{iq}^{(t)} & \text{if } q = j \\ 0 & \text{otherwise} \end{cases} \tag{5.29}$$

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

Runtime of Sub-gradient Descent Let $\mathbf{W}^*$ be the solution, i.e. $\mathbf{W}^* = \arg\min_{\mathbf{W} \in S} h(\mathbf{W})$. Shalev-Shwartz and Singer [129] showed the following convergence rate for Algorithm 1,

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

where L is a constant bounding the magnitude of the sub-gradient, i.e. $\forall \tau, ||\mathbf{V}||^2 \leqslant L$. 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 above algorithm [129].

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

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

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

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

□

Now we will briefly derive a bound $L = c\Big(\sqrt{2\lambda} + R\Big)^2$. The $h^{(t)}_{RL,cccp}(\mathbf{W})$ and $h^{(t)}_{RL,h}(\mathbf{W})$ objectives satisfy the criteria of Lemma 1, so we can replace unconstrained minimization with minimization restricted to the set $S = \{\mathbf{W} : ||\mathbf{W}||^2 \leqslant \frac{2}{\lambda}\}$ without changing the solution. It is also necessary to bound the magnitude of an instance feature vector, $||\mathbf{x}|| \leqslant R$. Note that $\mathbf{W} \in S$ implies $||\lambda \mathbf{w}_q|| \leqslant \lambda\sqrt{\frac{2}{\lambda}} = \sqrt{2\lambda}$. By the triangle inequality, $||\mathbf{v}^{(t,\tau),q}_{RL,cccp}|| \leqslant \sqrt{2\lambda} + R$. Summing over classes, $||\mathbf{V}^{(t,\tau)}_{RL,cccp}||^2 = \sum_{q=1}^{c} ||\mathbf{v}^{(t,\tau),q}_{RL,cccp}||^2 \leqslant c\Big(\sqrt{2\lambda} + R\Big)^2$.

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

To compute $\mathbf{V}^{(t,\tau)}_{RL,h}$ or $\mathbf{V}^{(t,\tau)}_{RL,cccp}$, one could compute each component according to equation (5.27) or (5.29), but this will take $O(nc^3)$ time. Algorithms 2 and 3 compute the sub-gradients 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 sub-gradient descent. In contrast, the CCCP method must recompute the support instances in each iteration of sub-gradient descent, so the runtime of one iteration of sub-gradient descent is $O(m + nc^2)$.

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

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: Sub-gradient $\mathbf{V}_{RL,cccp}^{(t,\tau)}$

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 $\mathbf{V}_{RL,cccp}^{(t,\tau)} = [\mathbf{v}_1, \ldots, \mathbf{v}_c]$

Running $T = O(\frac{L'}{\lambda\epsilon})$ iterations of sub-gradient 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.

5.4.3.5 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 re-computed in each iteration of sub-gradient descent, and the sub-gradient depends on these different support instances. The heuristic also runs a constant number of iterations of sub-gradient 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 non-convex.

Because the general rank-loss objective is non-convex, the starting weights may affect the optimum that is found by SIM-CCCP or SIM-Heuristic. We discuss the 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
    $\mathbf{W} = \mathbf{W}^{(t,\tau)} - \dfrac{1}{\lambda\tau}\mathbf{V}$
    $\mathbf{W}^{(t,\tau+1)} = \min\{1, \sqrt{\dfrac{2}{\lambda}}/||\mathbf{W}||\}\mathbf{W}$
  end
  $\tau^* = \arg\min_{\tau} h_{RL,h}^{(t)}(\mathbf{W}^{(t,\tau)})$
  $\mathbf{W}^{t+1} = \mathbf{W}^{(t,\tau^*)}$
end
return $\mathbf{W}^{(T+1)}$

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
      $\mathbf{W} = \mathbf{W}^{(t,\tau)} - \dfrac{1}{\lambda \tau_{total}} \mathbf{V}$
      $\mathbf{W}^{(t,\tau+1)} = \min\{1, \sqrt{\dfrac{2}{\lambda}} / ||\mathbf{W}||\} \mathbf{W}$
    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 $\mathbf{W}^{(T+1)}$

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 $\mathbf{z}(\mathbf{x}) = \sqrt{\frac{1}{D}} [cos(\omega_1 \cdot \mathbf{x}), sin(\omega_1 \cdot \mathbf{x}), \ldots, cos(\omega_D \cdot \mathbf{x}), sin(\omega_D \cdot \mathbf{x})]$

5.4.4 Non-linear Classification via Kernels

So far we have only considered linear classifiers. There are several ways to extend our proposed methods to non-linear classification via kernels. One way is to use the standard dual kernel trick. Shalev-Shwartz et al. [131] 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 a disadvantage of increasing the asymptotic complexity to solve the optimization problem.

We instead use the random Fourier feature method of Rahimi and Recht [119] (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 i.i.d. 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.

5.5 Experiments

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

5.5.1 Experimental Setup

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

Table 5.2: MIML datasets

Dataset	Classes	Dimension	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

5.5.1.1 Datasets

Table 5.5.1.1 summarizes the properties of each dataset used in our experiments. All of these datasets are available online.[3]

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 5.1 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 4,998 are labeled, and the rest are unlabeled. The available instance labels were provided by a human expert. The spectrograms were originally labeled by manually

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

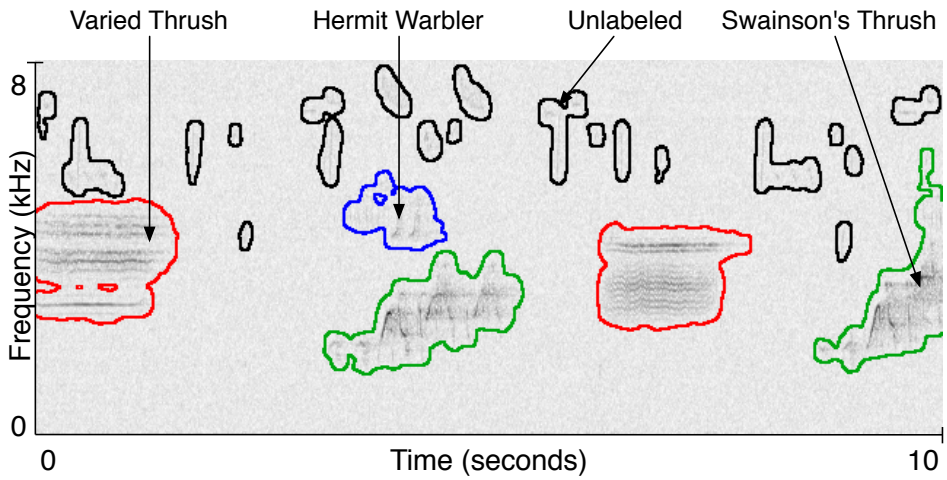


Figure 5.1: An example spectrogram from the HJA Birdsong dataset. This spectrogram corresponds to one bag. Each outlined region is an instance.

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 [18].

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

Figure 5.2: 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.

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.

Image Dataset: MSRC v2 A subset of the Microsoft Research Cambridge (MSRC) image dataset[4] [154] 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 (Fig. 5.2). Several authors used MSRC v2 in MIML experiments [148, 159, 165].

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 (i.e. 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 [42], and a 32-dimensional histogram of colors.

PASCAL Visual Object Challenge 2012 The PASCAL Visual Object Challenge (VOC) 2012 Segmentation dataset [57] 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 (Fig. 5.3). We construct a MIML

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

Figure 5.3: An image from PASCAL VOC 2012, the corresponding segmentation into instances, and the instance labels.

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 colors 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 1,053 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 [65] 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" [24], and "The Road Not Taken" [66]. 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}.

5.5.1.2 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 1,053 images, and for the inductive mode we learn from the 'train' subset and evaluate accuracy on the 'val' subset. That is why the VOC column in Table 5.5.2(c) and (d) does list a standard deviation.

5.5.1.3 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. [40]. Finally, as a reference we evaluate a SISL SVM classifier, which has the unfair advantage of learning directly from instance labels. Below 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

$$h_{Ham}(\mathbf{W}) = \frac{\lambda}{2}||\mathbf{W}||^2 + \frac{1}{nc}\sum_{i=1}^{n}\sum_{j=1}^{c}\max\{0, 1 - Y_{ij}F_j(X_i)\} \tag{5.30}$$

or in terms of support instances,

$$\hat{h}_{Ham}(\mathbf{W}) = \frac{\lambda}{2}||\mathbf{W}||^2 + \frac{1}{nc}\sum_{i=1}^{n}\sum_{j=1}^{c}\max\{0, 1 - Y_{ij}\mathbf{w}_j \cdot \hat{\mathbf{x}}_{ij}\} \tag{5.31}$$

To optimize this objective, we use a variation of the SIM-Heuristic algorithm with the sub-gradient[5]

$$\mathbf{v}_{Ham}^{(t,\tau),q} = \lambda \mathbf{w}_q^{(t,\tau)} - \frac{1}{nc}\sum_{i=1}^{n} I[Y_{iq}\mathbf{w}_q^{(t,\tau)} \cdot \hat{\mathbf{x}}_{iq}^{(t)} < 1] Y_{iq}\hat{\mathbf{x}}_{iq}^{(t)} \tag{5.32}$$

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,

$$Y_{ij}F_j(X_i) \geqslant 1 \text{ for } i = 1, \ldots, n, j = 1, \ldots, c \tag{5.33}$$

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

$$-\max_{\mathbf{x} \in X_i} \mathbf{w}_j \cdot \mathbf{x} \quad \geqslant \quad 1 \tag{5.34}$$

$$\forall \mathbf{x} \in X_i : -\mathbf{w}_j \cdot \mathbf{x} \quad \geqslant \quad 1 \tag{5.35}$$

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

$$\max_{\mathbf{x} \in X_i} \mathbf{w}_j \cdot \mathbf{x} \geqslant \frac{1}{n_i}\sum_{\mathbf{x} \in X_i} \mathbf{w}_j \cdot \mathbf{x} \geqslant 1 \tag{5.36}$$

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

[5]Note that (5.30) satisfies the conditions for Lemma 1 and a bound on the magnitude of the sub-gradient 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 Sec. 5.4.3.4.

solved rather than a sequence.

Ambiguous Label Classification (ALC) Cour et al. [39] 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* [40]. 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 [61] 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 [26] with a linear kernel. Note that LIBSVM uses one linear model for each pair of classes rather than one for each class.

5.5.1.4 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 [16].

Where the random Fourier kernel features are used, the parameter $D = 50$ (as in [119]), 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 sub-gradient descent. For the CCCP algorithm, we use $T = 10$, $K = 100$, and $K_{max} = 1000$. Figure 5.4 shows that most improvement in the rank loss objective occurs within $T = 10$ outer iterations. We presented empirical results in [16] showing that $K = 100$ is a reasonable number of iterations of sub-gradient 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.

5.5.2 Results

Accuracy results are listed in Table 5.5.2. In particular, Tables 5.5.2 (a) and (b) present the accuracy results for the transductive mode (with no kernel, and the RBF kernel respectively). Tables 5.5.2 (c) and (d) present the results for the inductive mode.

Following the recommendations of Demšar [48] 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.

5.5.2.1 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

Table 5.3: 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

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^{3}MIML. Since M^{3}MIML uses the `max` aggregation model, we compare it with 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^{3}MIML, 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.

5.5.2.2 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 (i.e., 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.

5.5.2.3 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 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.

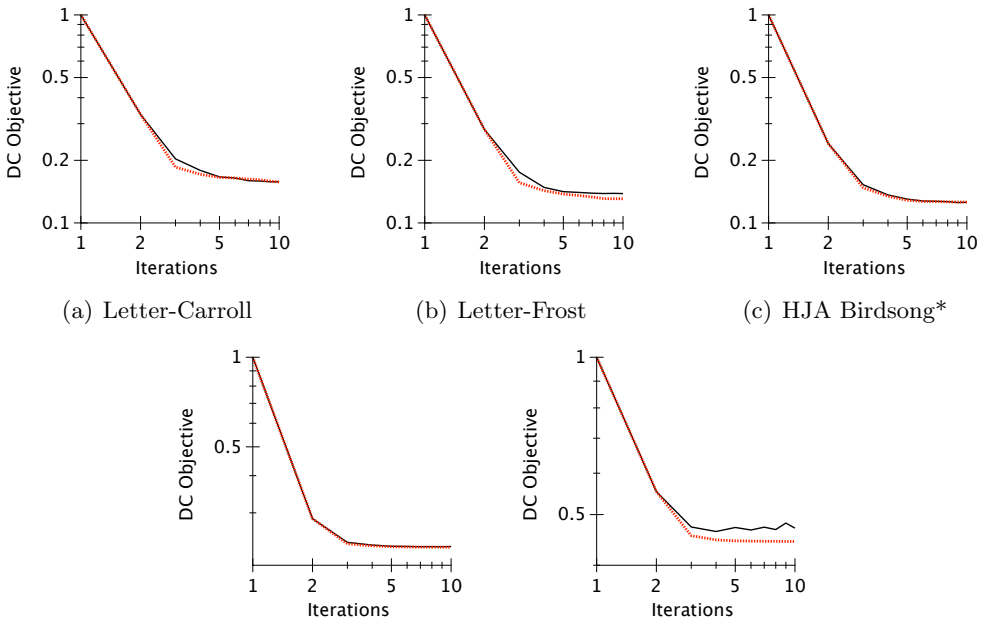
5.5.2.4 Comparison of CCCP and Heuristic Optimization

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 non-convex rank-loss objective. Figure 5.4 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., Fig. 5.4 (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 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.

5.5.2.5 Random Fourier Kernel Features

We now examine accuracy improvements achieved by using random Fourier kernel features [119]. Results using this method are listed in Tables 5.5.2 (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 5.5.2 (b) and (d)) against the results of corresponding linear methods (the first three rows of Tables 5.5.2 (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.



(a) Letter-Carroll (b) Letter-Frost (c) HJA Birdsong*

(d) MSRC v2 (e) VOC 2012

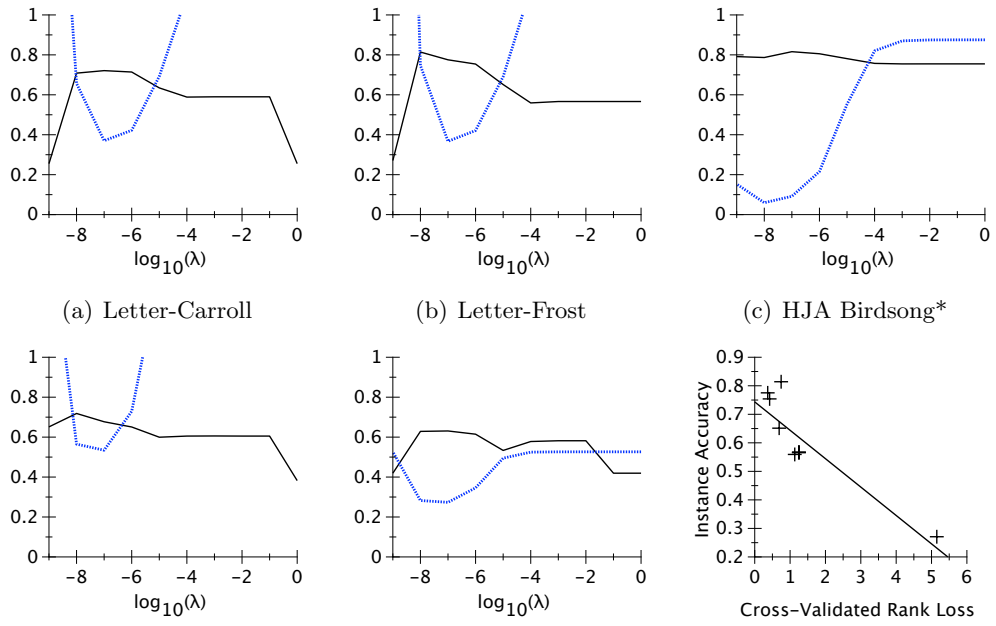
Figure 5.4: The objective h_{RL} vs. number of iterations of SIM-CCCP and SIM-Heuristic. SIM-CCCP is the dotted line. Transductive mode, `max` model, $\lambda = 10^{-7}$.

5.5.2.6 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 $Test(l)$.
- In each fold l, train an SIM on $Train(l)$, then compute un-regularized rank loss on $Test(l)$.



(a) Letter-Carroll (b) Letter-Frost (c) HJA Birdsong*

(d) MSRC v2 (e) VOC 2012 (f) Letter-Frost

Figure 5.5: (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.

Compute the average rank loss over all folds as a function of λ,

$$RL(\lambda) = \frac{1}{\kappa} \sum_{l=1}^{\kappa} \sum_{i \in Test(l)} \sum_{j \in Y_i, k \notin Y_i} \beta_i \max\{0, 1 - \big(F_j(X_i) - F_k(X_i)\big)\} \tag{5.37}$$

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

$$\hat{\lambda} = \arg\min_{\lambda} RL(\lambda) \tag{5.38}$$

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. [40] proposed a similar parameter selection scheme for ALC wherein the regularization parameter is selected by cross-validated

Table 5.4: 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

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 5.5 (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 5.5 (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. Scatter plots are similar for the other datasets. We find experimentally that the proposed method is reasonably effective for selecting the regularization parameter in the absence of any instance-labels.

5.5.2.7 Empirical Runtime

Table 5.5.2.7 gives empirical runtimes in seconds[6] 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 sub-gradient descent, and runs more iterations of sub-gradient descent to ensure monotonicity.

[6]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.

5.5.2.8 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 run-time and accuracy.

SISL SVM in the inductive mode [Table 5.5.2 (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.

5.6 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 [160] and the Exponential Multinomial Mixture model [161] are topic models for MIML datasets and use variational inference to perform instance labeling. Zha et al. [165] proposed the MLMIL algorithm, a conditional random field model for MIML image annotation that uses Gibbs sampling to infer instance labels. Du et al. [51] 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 alternative approach for optimizing a similar objective to the M^3MIML algorithm [168] (which learns an instance-level model, but was not designed for instance annotation). Also note that 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 [148] developed a 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. [147] 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 [115] proposed a MIML SVM algorithm which 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 [4] for multi-instance learning (MIL) uses CCCP to handle non-convexity (note the interpretation of MI-SVM as an instance of CCCP is due to Cheung and Kwok [34]). D-MimlSvm [172] also uses CCCP to optimize Hamming loss.

In recent work, Li et al. [96] 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.

5.7 Conclusion & 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 a 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 non-convex optimization problem; we offer a heuristic solution which is applicable to either `max` or `softmax` and a CCCP method which can be applied to the `max` model and guarantees monotonic decrease in the non-convex 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 sub-gradient 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 is 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 [148].

Chapter 6: Context-Aware MIML Instance Annotation[1]

6.1 Introduction

Instance annotation for multi-instance multi-label (MIML) data is a recent and little-studied problem for supervised classification. A MIML dataset consists of bags of instances paired with sets of labels. For example, in an image domain, a bag is an image, the instances in the bag are feature vectors describing regions, and the label set for a bag indicates which objects or categories the image contains. There are many algorithms that train a classifier on a MIML dataset to predict the label set for a new bag (e.g., the original formulation of MIML by [172]). In contrast, MIML instance annotation aims to train a classifier on a MIML dataset to **predict the instance labels**. For example, we train a classifier on images paired with sets of objects they contain, then predict the class label for each region of a new image.

MIML instance annotation differs from the traditional MIML problem of label set prediction (e.g., M^{3}MIML [168]), and multi-label classification (MLC, e.g., binary relevance). In particular, it is commonly assumed that each instance only belongs to one class, thus the predictions to be made are single labels for instances, not label sets. An appropriate objective for MIML instance annotation is to maximize instance-level accuracy (the fraction of correctly classified instances). However, it is not possible to train a model that directly optimizes accuracy on the training data, because instance labels are not available for training. Sometimes it is possible to modify a MIML or MLC algorithm that is designed for label set prediction, to predict instance labels. The problem with this approach is that the model is optimized for label set accuracy, not instance accuracy. Domain-specific instance annotation problems (e.g., for images) have been widely explored, however, to our knowledge only two prior studies have specifically considered the general domain-independent MIML instance annotation problem [16, 17]. Briggs et al. [16, 17] proposed rank-loss Support Instance Machines (SIM), a collection of SVM-style algorithms that learn a linear instance classifier by minimizing a rank loss objective on bag-level labels.

Prior work [16, 17] has observed that the rank-loss SIM algorithms, as well as several other baseline methods, achieve lower accuracy for inductive classification of instances (predicting

[1] "Context-Aware MIML Instance Annotation." Forrest Briggs, Xiaoli Z. Fern, Raviv Raich. Thirteenth IEEE International Conference on Data Mining, 2013. ICDM'13.

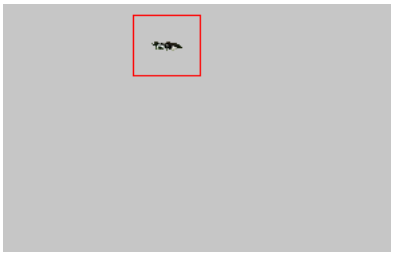


Figure 6.1: Inductive instance annotation without context – "What class is the region of pixels inside the red box?"

instance labels for previously unseen bags) in comparison to transductive classifications (predicting instance labels for bags with known label sets). We hypothesize that one way to improve the performance of inductive classification is to exploit the contextual information provided by other instances in the same bag.

Figure 6.1 illustrates the importance of using context in inductive instance annotation. The region of pixels inside the red box is an instance. A MIML instance annotation classifier might be asked to predict the class label of this instance. Without the context provided by the rest of the image, it is hard to classify, even for a human. Figure 6.2 shows the rest of the image. With this context available, it is much easier to recognize the instance. The situation illustrated by Fig. 6.1 is how inductive MIML instance annotation is posed in prior work [16, 17]. It is not as important to use the context provided by other instances in the same bag for transductive classification, because the bag label set is already known, and provides a similar kind of context. Consider the same example in Fig. 6.1. If we know that the image contains labels "cow" and "grass," we do not need to see the rest of the image to conclude that the label for this instance should be "cow."

Figure 6.2: Inductive instance annotation with context – "What class is the region of pixels inside the red box?" This image is from the VOC12 data.

This chapter proposes a new algorithm for MIML instance annotation designed to improve inductive instance classification accuracy by exploiting the context provided by other instances in the same bag. In particular, we capture the context by modeling label correlations in the bag label set. The proposed algorithm is a multi-instance multi-label ensemble of classifier chains, called MIML-ECC (Sec. 6.4). MIML-ECC has no "tuning" parameters (which are necessarily selected by a heuristic in prior work) (Sec. 6.5.4), and is asymptotically efficient in all dimensions of a problem (number of bags, instances, classes, and feature dimension) (Sec. 6.4.4). The training algorithm is closely related to EM (Sec. 6.4.3), and the classification algorithm selects the maximum a posteriori (MAP) instance label as estimated by the ensemble (Sec. 6.4.2). Experiments show that MIML-ECC achieves higher accuracy than several recent methods and baselines, including Hamming, rank, and ambiguous-loss SVMs, and comparable accuracy to a recent graphical model (Sec. 6.5.5). Further experiments show that the chain structure outperforms binary relevance (Sec. 6.5.6), and an ensemble of chains outperforms a single chain (Sec. 6.5.7).

6.2 Problem Statement

Our goal is to learn an instance-level classifier by training on a MIML dataset consisting of n bags paired with their corresponding label sets $\{(B_1, Y_1), \ldots, (B_n, Y_n)\}$, where B_i is a bag, $Y_i \subseteq \mathcal{Y} = \{1, \ldots, c\}$ is its label set, and c is the total number of classes. Each bag B_i contains n_i instances, i.e., $B_i = \{\mathbf{x}_{i1}, \ldots, \mathbf{x}_{in_i}\}, \mathbf{x} \in \mathcal{X} = \mathbb{R}^d$.

We assume that each instance $\mathbf{x}$ in B_i has a single label $y \in \mathcal{Y}$. The instance labels are not available in the training data; and we only have ambiguous information about them provided through the bag label sets.

We consider instance annotation in both transductive and inductive modes, which differ in what information is available at the classification stage. The transductive classifier is defined as:

$$y = f(\mathbf{x}, B, Y) : \mathcal{X} \times 2^{\mathcal{X}} \times 2^{\mathcal{Y}} \rightarrow \mathcal{Y} \tag{6.1}$$

The notation $f(\mathbf{x}, B, Y)$ indicates that we are given all of the instances in a bag B, its label set Y, and the goal is to predict the label y for a specific instance $\mathbf{x}$ in B.

The inductive mode classifies an instance without the bag label set given. Prior work [16] on

Table 6.1: Frameworks for supervised classification

Framework	Training Dataset	Classifier
SISL	$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$	$y = f(\mathbf{x}) : \mathcal{X} \to \mathcal{Y}$
MIL	$(B_1, y_i), \ldots, (B_n, y_n)$	$y = F(B) : 2^{\mathcal{X}} \to \{0, 1\}$
MLC	$(\mathbf{x}_1, Y_1), \ldots, (\mathbf{x}_n, Y_n)$	$Y = f(\mathbf{x}) : \mathcal{X} \to 2^{\mathcal{Y}}$
MIML	$(B_1, Y_1), \ldots, (B_n, Y_n)$	$Y = F(B) : 2^{\mathcal{X}} \to 2^{\mathcal{Y}}$
ALC/SLL	$(\mathbf{x}_1, Y_1), \ldots, (\mathbf{x}_n, Y_n)$	$y = f(\mathbf{x}) : \mathcal{X} \to \mathcal{Y}$

MIML instance annotation formulates the inductive classifier as $f(\mathbf{x}) : \mathcal{X} \to \mathcal{Y}$, which ignores any contextual information from the bag containing $\mathbf{x}$. We instead formulate the inductive classifier as

$$y = f(\mathbf{x}, B) : \mathcal{X} \times 2^{\mathcal{X}} \to \mathcal{Y} \tag{6.2}$$

The difference is that when classifying an instance $\mathbf{x}$, we know that it is part of a bag B, and can use the contextual information of B to improve the prediction.

6.2.1 Related Problems

There are many other formulations of supervised classification that are related to MIML instance annotation. The main difference between these frameworks is the structure of training data (instance or bag, single- or multi-label), and the input to and type of prediction made by the classifier (instance-level or bag-level, single or multi-label). Refer to Table 6.1 for a statement of the training data and inductive classifier in each framework.

The most common supervised classification formulation is single-instance single-label (SISL). Most standard methods such as support vector machines, decision trees, and logistic regression are for SISL. Multiple-instance learning (MIL) is a framework where the training data consists of bags of instances paired with a single binary label, and the classifier maps bags to binary labels. Multi-label classification (MLC) [122] pairs single instances with sets of labels, and the goal is to predict a label set given a new instance.

Ambiguous label classification (ALC) [40] and superset label learning (SLL) [99] have the same structure of training data as MLC, but assume only one label in the set is correct and

the rest are "distractors." The goal is to learn a classifier to predict a single label for a new instance. MIML instance annotation can be reduced to ALC/SLL by pairing each instance with its bag label set. However, this reduction can be undesirable as it discards the context of the bag.

6.3 Background

A key observation motivating our approach is that the context provided by a bag's label set is useful for classifying instances. In the previous example, knowing that there is "grass" in the image can help for predicting the label "cow" for the given instance, because the labels "cow" and "grass" are correlated. A natural way to exploit such context is to follow a classifier-chain approach, which has been previously developed for MLC to exploit label correlation. Below we begin with a review of classifier chains for MLC. We then discuss some design patterns in MIL and MIML algorithms that learn an instance-level model from bag-level labels, which provide inspiration for our algorithm.

6.3.1 Classifier Chains for Multi-Label Classification

Originally introduced for MLC, classifier chains [122] exploit label correlation by building a chain of binary classifiers. Given an instance $\mathbf{x}$, we denote its label set Y as a binary vector: $Y = [Y^1, \ldots, Y^c]$, where $Y^j = 1$ if the label set for instance $\mathbf{x}$ contains class j. We use $Y^{1:j-1} = [Y^1, \ldots, Y^{j-1}]$ to refer to the first $j-1$ elements of Y. The key idea of classifier chains is to use a chain factorization of the conditional joint distribution of Y:

$$P(Y|\mathbf{x}) = P(Y^1|\mathbf{x}) \prod_{j=2}^{c} P(Y^j|\mathbf{x}, Y^{1:j-1}) \tag{6.3}$$

During training, one binary model $P(Y^j|\mathbf{x}, Y^{1:j-1})$ is learned for each class j, which depends on $\mathbf{x}$, and all of the preceding classes $1, \ldots, j-1$. Let $\oplus$ denote vector concatenation. The basic training algorithm is (Algorithm 7):

ALGORITHM 7: MLC Probabilistic Classifier Chain – Train

for $j = 1, \ldots, c$ **do**
  $\mathcal{D}_j = \{\ldots, (\mathbf{x}_i \oplus Y_i^{1:j-1}, Y_i^j), \ldots\}_{i=1}^n$
  train classifier $P(Y^j|\mathbf{x}, Y^{1:j-1})$ on $\mathcal{D}_j$
end

For each class j, a binary supervised classification problem $\mathcal{D}_j$ is created (this is a standard SISL problem, not an MLC problem). This 2-class problem has n instances like the original MLC problem. Each instance consists of the original feature vector $\mathbf{x}_i$ concatenated with part of the corresponding label vector $[Y_i^1, \ldots, Y_i^{j-1}]$, and paired with the binary label Y_i^j. The binary model for class j, namely $P(Y^j|\mathbf{x}, Y^{1:j-1})$, can be learned using any binary probabilistic classifier, e.g., logistic regression or Random Forest (RF) [14].

To classify a new instance $\mathbf{x}$ with a probabilistic classifier chain, one can evaluate $P(Y|\mathbf{x})$ for all 2^c possible label vectors Y, and pick one that minimizes a set-level loss function. However, this approach may be intractable unless c is small. An alternative is to greedily construct a single value of Y. A basic greedy algorithm [47] is:

ALGORITHM 8: MLC Probabilistic Classifier Chain – Classify

$Y = []$
for $j = 1, \ldots, c$ **do**
  $Y = Y \oplus I[P(Y^j|\mathbf{x} \oplus Y) > 0.5]$
end
return Y

In ensembles of classifier chains (ECC) [122], there are multiple chains, each of which is learned as above, but factorizing the classes in a different random order. When classifying with ECC, each chain votes. ECC reduces the sensitivity to the specific order of the chain and is generally observed to improve accuracy over a single chain.

6.3.2 From Instance to Bag Labels

A central problem in MIL and MIML is that labels are only provided at the bag level. Learning an instance classifier from bag label sets requires an assumption about the relationship between the observed label sets and the hidden instance labels. A common assumption in MIL is that if any instance is positive, the bag label is positive, otherwise it is negative. The corresponding

assumption in MIML is that the bag label set is equal to the union of instance labels. Prior algorithms approximate these assumptions using different formulations, e.g., the max model.

In the MIL setting, the max model is: $F(B) = \max_{\mathbf{x} \in B} f(\mathbf{x})$, i.e. the bag-level output F is the max over the instance-level outputs f on all instances in the bag.

For probabilistic MIL classifiers, the max model has also been called the "most-likely-cause estimator" [110],

$$P(y = 1|B,\theta) = \max_{\mathbf{x} \in B} p(y = 1|\mathbf{x}, \theta) \tag{6.4}$$

The equivalent formulation for MIML [16, 168] applies the same principle for each class $j = 1, \ldots, c$:

$$F_j(B) = \max_{\mathbf{x} \in B} f_j(\mathbf{x}) \tag{6.5}$$

Given a model for connecting bag labels with instance labels, the output of a bag-level classifier can sometimes be expressed as a function of a single instance in the bag or representing the bag. For example, assuming the max model for MIL we have:

$$F(B_i) = \max_{\mathbf{x} \in B_i} f(\mathbf{x}) = f(\hat{\mathbf{x}}_i) \tag{6.6}$$

$$\hat{\mathbf{x}}_i = \arg\max_{\mathbf{x} \in B_i} f(\mathbf{x}) \tag{6.7}$$

where $\hat{\mathbf{x}}_i$ is referred to as the support instance (or "witness instance" [4]) for bag B_i. We can define support instances similarly for MIML, except that one support instance is defined for each class and each bag.

Many existing algorithms for MIL (e.g., MI-SVM [4] and EM-DD [169]) and MIML (e.g., SIM [17]) alternate between computing support instances based on a current classifier, and training a SISL classifier on the support instances. Our proposed algorithm follows the same pattern.

6.4 Proposed Methods

Our goal is to learn a classifier that predicts the label of a given instance, using its feature vector $\mathbf{x}$ and the context provided by the bag B containing $\mathbf{x}$. We propose the MIML-ECC

Table 6.2: Summary of Notation

Notation	Meaning
$\oplus$	vector concatenation operator
B_i	i'th bag of instances in the training data
Y_i	label set for bag B_i, $Y_i \subseteq \{1, \ldots, c\}$
n	number of bags in the training set
n_i	number of instances in bag B_i
$\pi(j)$	the j'th class in some permutation π
$\pi_l(j)$	the j'th class in the permutation for chain l
$Y_i^{\pi_l(j)}$	the j'th bit (0 or 1) of the label set Y_i in order π_l
$Y_i^{\pi_l(1):\pi_l(j-1)}$	the first $j-1$ bits of the label set Y_i in order π_l
$\mathbf{x} \in B_i$	an instance in bag B_i, a vector in $\mathbb{R}^d$
f_{jl}	instance-level score function for class $\pi_l(j)$
F_{jl}	bag-level score function for class $\pi_l(j)$
$\hat{\mathbf{x}}_{ijl}$	support-instance for bag i, chain l, class $\pi_l(j)$
y^k	indicator variable for instance $\mathbf{x}$ in class k

algorithm, which is motivated by the observation that the prediction of whether an instance belongs to a particular class can be influenced by the presence/absence of some other classes in the bag. To capture the label correlation, we assume an ordered chain structure such that whether an instance belongs to a particular class depends on whether the bag contains classes earlier in the chain. Table 6.2 summarizes notation for the proposed method.

6.4.1 Training

A classifier chain for MLC is a chain of SISL classifiers. At a high level, our method can be viewed as building an ensemble of L chains of MIL classifiers. Each chain $l = 1, \ldots, L$ in the ensemble views the classes $1, \ldots, c$ in a different order π_l, such that $\pi_l(j)$ is the j'th class in the order for chain l. We will use F_{jl} to denote the MIL classifier for the j-th class in chain l, which predicts the presence/absence of class $\pi_l(j)$ in the label set of a bag given the bag and $Y^{\pi_l(1):\pi_l(j-1)}$, the presence/absence information of the first $j-1$ classes in chain l. The training algorithm viewed in terms of MIL classifiers is (Algorithm 9):

ALGORITHM 9: MIML-ECC – Train (Bag-Level View)

Input: MIML dataset $\{(B_1, Y_1), \ldots, (B_n, Y_n)\}$

Output: MIL classifiers F_{jl}

for $l = 1, \ldots, L$ **do**
  π_l = random–permutation($[1, \ldots, c]$)
  for $j = 1, \ldots, c$ **do**
    $\mathcal{D}_{jl} = \{\ldots, (B_i \oplus Y_i^{\pi_l(1):\pi_l(j-1)}, Y_i^{\pi_l(j)}), \ldots\}_{i=1}^{n}$
    train MIL Classifier F_{jl} on $\mathcal{D}_{jl}$
  end
end

Each MIL dataset $\mathcal{D}_{jl}$ constructed in the algorithm pairs the bag B_i (and the context $Y_i^{\pi_l(1):\pi_l(j-1)}$) with one bit of the label vector $Y_i^{\pi_l(j)}$. In a standard MIL formulation, there are only bags of instances, so it is a modification of MIL to allow the context $Y_i^{\pi_l(1):\pi_l(j-1)}$, which is a vector in $\mathbb{R}^{j-1}$, to be associated with the bag rather than an instance. However, in practice we simply append this vector to the end of all of the instance features.

Because our goal is ultimately to predict instance labels, we instantiate this template with a MIL classifier that internally builds an instance-level model. The instance-level models are SISL probabilistic classifiers f_{jl} for $j = 1, \ldots, c$ and $l = 1, \ldots, L$. We assume f_{jl} maps the input $\mathbf{x} \oplus Y^{\pi_l(1):\pi_l(j-1)}$ to an output in $[0, 1]$ (as is the case for a RF). Recall that $\mathcal{Y} = \{1, \ldots, c\}$; we encode the label $y \in \mathcal{Y}$ of instance $\mathbf{x}$ with c binary indicator variables $y^1, \ldots, y^c$ where $y^j = I[y = j]$, and interpret $f_{jl} : \mathbb{R}^{d+j-1} \to [0, 1]$ as the posterior probability $P(y^{\pi_l(j)} | \mathbf{x}, Y^{\pi_l(1):\pi_l(j-1)})$. MIL classifiers F_{jl} can be obtained from the instance-level classifiers using the max model, taking into account the context $Y^{\pi_l(1):\pi_l(j-1)}$:

$$F_{jl}(B_i \oplus Y^{\pi_l(1):\pi_l(j-1)}) = \max_{\mathbf{x} \in B_i} f_{jl}(\mathbf{x} \oplus Y^{\pi_l(1):\pi(j-1)}) \tag{6.8}$$

Similar to the MIL algorithm EM-DD, and rank-loss SIM for MIML, we define the bag-level model in terms of a support instance. In MIML-ECC, there is a different support instance for each bag, class, *and chain*. The bag-level model in terms of support instances is

$$\begin{aligned} F_{jl}(B_i \oplus Y^{\pi_l(1):\pi_l(j-1)}) &= f_{jl}(\hat{\mathbf{x}}_{ijl} \oplus Y_i^{\pi_l(1):\pi_l(j-1)}) \\ \hat{\mathbf{x}}_{ijl} &= \arg\max_{\mathbf{x} \in B_i} f_{jl}(\mathbf{x} \oplus Y_i^{\pi_l(1):\pi_l(j-1)}) \end{aligned}$$

The support instance $\hat{\mathbf{x}}_{ijl}$ is the instance in bag B_i that is most representative of class $\pi_l(j)$, according to the classifiers in chain l.

The MIML-ECC training algorithm alternates K times between updating support instances according to the max model, then training SISL classifiers on binary datasets that pair support instances with bits of the label set. In the first iteration, there are no instance classifiers f_{jl} to compute support instances from, so we start by setting the support instances to the average of the instances in each bag, as in [16, 17]. The instance-level view of the training algorithm is:

ALGORITHM 10: MIML-ECC – Train (Instance-Level View)

Input: MIML dataset $\{(B_1, Y_1), \ldots, (B_n, Y_n)\}$
Output: SISL classifiers f_{jl}
for $l = 1, \ldots, L$ **do**
  π_l = random–permutation($[1, \ldots, c]$)
  for $k = 1, \ldots, K$ **do**
    if $k = 1$ **then**
      for $i = 1, \ldots, n, j = 1, \ldots, c$ **do**
        $\hat{\mathbf{x}}_{ijl} = \frac{1}{n_i} \sum_{\mathbf{x} \in B_i} \mathbf{x}$
      end
    else
      for $i = 1, \ldots, n, j = 1, \ldots, c$ **do**
        $\hat{\mathbf{x}}_{ijl} = \arg\max_{\mathbf{x} \in B_i} f_{jl}(\mathbf{x} \oplus Y_i^{\pi_l(1):\pi_l(j-1)})$
      end
    end
    for $j = 1, \ldots, c$ **do**
      $\mathcal{D}_{jl} = \{\ldots, (\hat{\mathbf{x}}_{ijl} \oplus Y_i^{\pi_l(1):\pi_l(j-1)}, Y_i^{\pi_l(j)}), \ldots\}_{i=1}^{n}$
      train SISL classifier f_{jl} on $\mathcal{D}_{jl}$
    end
  end
end

6.4.2 Classification

In the training phase, instance-level binary classifiers $f_{jl}(\mathbf{x} \oplus Y^{\pi_l(1):\pi_l(j-1)})$ are obtained for every class j and chain l. The output of f_{jl} can be considered an estimate of the posterior $P(y^{\pi_l(j)} | \mathbf{x}, Y^{\pi_l(1):\pi_l(j-1)})$, so we consider a probabilistic framework for instance classification based on the maximum a posteriori (MAP) approach. This is how MIML-ECC approximately

optimizes instance accuracy, the desired performance measure for MIML instance annotation.

6.4.2.1 Transductive Mode

In the transductive mode, we condition on the bag and its label set, and predict instance labels according to

$$f(\mathbf{x}, B, Y) = \arg\max_{j \in Y} P(y^j | \mathbf{x}, B, Y) = \arg\max_{j \in Y} P(y^j | \mathbf{x}, Y)$$

This prediction rule assumes that bag label set Y provides all of the contextual information that is relevant to predicting the label for $\mathbf{x}$, i.e. the label is conditionally independent of the other instances in the bag B given Y.

During training we introduced random orders π for the purpose of constructing an ensemble. Now we take a Bayesian approach and assume that π is random variable from a uniform prior $P(\pi)$, so each chain in the ensemble corresponds to one i.i.d. sample $\pi_l \sim P(\pi)$ for $l = 1, \ldots, L$. We estimate the probability for instance $\mathbf{x}$ to have label $y = k$ as $P(y^k | \mathbf{x}, Y) = E_\pi[P(y^k | \mathbf{x}, Y, \pi)]$ using L samples, one for each chain in the ensemble.

$$P(y^k | \mathbf{x}, Y) \approx \frac{1}{L} \sum_{l=1}^{L} \sum_{\{j : \pi_l(j) = k\}} P(y^{\pi_l(j)} | \mathbf{x}, Y^{\pi_l(1):\pi_l(j-1)}, \pi_l)$$

The algorithm for classification in the transductive mode is:

ALGORITHM 11: MIML-ECC – Classify (Transductive)

Input: instance $\mathbf{x}$, label set Y
Output: label y
for $j = 1, \ldots, c$ **do**
  $y^j = 0$
end
for $l = 1, \ldots, L$ **do**
  for $j = 1, \ldots, c$ **do**
    $y^{\pi_l(j)} = y^{\pi_l(j)} + f_{jl}(\mathbf{x} \oplus Y^{\pi_l(1):\pi_l(j-1)})$
  end
end
$y = \arg\max_{j \in Y} y^j$

6.4.2.2 Inductive Mode

In the inductive setting, the bag label set is not given, so the posterior required for classification conditions only on the instances from bag B (and not the bag label set). Therefore, we predict the instance label as the class with the highest posterior probability

$$y = f(\mathbf{x}, B) = \arg\max_{j=1,\ldots,c} P(y^j|\mathbf{x}, B) \tag{6.9}$$

The probability $P(y^j|\mathbf{x}, B)$ is not directly modeled by the instance-level classifiers f_{jl}; instead we estimate this probability by marginalizing $P(y^j|\mathbf{x}, Y, B)$ over the latent variable Y. This process requires a probabilistic model for Y given B, which we develop below.

Assumption 1: Given an order π, an instance $\mathbf{x}$ and the bag-level labels $Y^{\pi(1):\pi(j-1)}$, $y^{\pi(j)}$ is conditionally independent of any other instances in the same bag B,

$$P(y^{\pi(j)}|\mathbf{x}, B, Y^{\pi(1):\pi(j-1)}, \pi) = P(y^{\pi(j)}|\mathbf{x}, Y^{\pi(1):\pi(j-1)}, \pi)$$

For training, we defined the relation between instance labels and bag label sets according to the max model. The max model is also part of our assumptions for inference, although we will rewrite it in probability notation.

Assumption 2: Bag label sets and instance labels are linked via the max model,

$$P(Y^{\pi(j)}|B, Y^{\pi(1):\pi(j-1)}, \pi) = \max_{\mathbf{x}\in B} P(y^{\pi(j)}|\mathbf{x}, Y^{\pi(1):\pi(j-1)}, \pi)$$

Similar to a classifier chain for MLC, the conditional distribution of the bag label set is factored as a chain in the order π as

$$P(Y|B,\pi) = P(Y^{\pi(1)}|B,\pi) \prod_{j=2}^{c} P(Y^{\pi(j)}|B, Y^{\pi(1):\pi(j-1)}, \pi)$$

Recall that Assumption 2 defines the conditional probability for $Y^{\pi(j)}$ in terms of the instance-level probabilities for $y^{\pi(j)}$, while Assumption 1 defines the instance-level probabilities for $y^{\pi(j)}$ in terms of $Y^{\pi(1):\pi(j-1)}$.

We estimate $P(y^j|\mathbf{x}, B)$ by sampling as follows. For a given π, we apply Assumption 1 to

obtain $P(y^{\pi(j)}|\mathbf{x}, B, \pi)$

$$\begin{aligned} &= E_{Y^{\pi(1):\pi(j-1)}|B,\pi}\big[P(y^{\pi(j)}|\mathbf{x}, Y^{\pi(1):\pi(j-1)}, B, \pi)\big] \\ &= E_{Y^{\pi(1):\pi(j-1)}|B,\pi}\big[P(y^{\pi(j)}|\mathbf{x}, Y^{\pi(1):\pi(j-1)}, \pi)\big] \end{aligned} \tag{6.10}$$

Because π is a permutation, computing $P(y^{\pi(j)}|\mathbf{x}, B, \pi)$ for $j = 1, \ldots, c$ implies computing $P(y^j|\mathbf{x}, B, \pi)$ for all j.

Finally, we average the posterior estimates over multiple samples from a uniform prior on π:

$$P(y^j|\mathbf{x}, B) = E_\pi\big[P(y^j|\mathbf{x}, B, \pi)\big] \tag{6.11}$$

As in the transductive mode, each chain in the ensemble gives one sample of $\pi_l \sim P(\pi)$ to estimate the expectation. The inductive classification algorithm is:

ALGORITHM 12: MIML-ECC – Classify (Inductive)

Input: bag $B = \{\mathbf{x}_1, \ldots, \mathbf{x}_{n_i}\}$

```
1  for i = 1,...,n_i, j = 1,...,c do
2  |  y_i^j = 0
3  end
4  for l = 1,...,L do
5  |  Y = []
6  |  for j = 1,...,c do
7  |  |  for i = 1,...,n_i do
8  |  |  |  y_i^{π_l(j)} = y_i^{π_l(j)} + f_jl(x_i ⊕ Y)
9  |  |  end
10 |  |  p_j = max_{i=1,...,n_i} f_jl(x_i ⊕ Y)
11 |  |  Y = Y ⊕ Bernoulli(p_j)
12 |  end
13 end
14 for i = 1,...,n_i : do
15 |  y_i = arg max_{j=1,...,c} y_i^j
16 end
```

Output: instance labels $y_1, \ldots, y_{n_i}$

Line 8 updates the estimate of $y_i^{\pi_l(j)}$ based on one sample of the expectation (6.10). Line 10 applies the max model (Assumption 2). In lines 5 through 10, the pseudocode variable Y stores $Y^{\pi_l(1):\pi_l(j-1)}$. Line 11 samples $Y^{\pi_l(j)}$ from a $Bernoulli(p_j)$ distribution, and appends it

to the current label vector.

6.4.3 Similarities with EM

The proposed training algorithm is a heuristic, and is not proven to converge over multiple support instances updates. However, it is closely related to prior work using support instances with expectation maximization (EM). We discuss this similarity with prior algorithms to provide evidence that MIML-ECC can be expected to improve in accuracy as the number of support instance updates K increases.

EM-DD [169] is a widely used algorithm for MIL (single-labeled bags of instances), in the spirit of EM (a formal proof is not given). The "E-step" consists of computing support instances, and the "M-step" maximizes likelihood in a model involving the support instances. EM-DD also uses the max model to define the support instances. The main difference in how support instances are treated in MIML-ECC is that each bag has a different support instance for each class and chain. Recall that MIML-ECC trains SISL classifiers f_{jl} in each iteration. If the base SISL classifier maximizes log-likelihood (e.g., logistic regression), there is a direct correspondence with the M-step of EM-DD. In our implementation of MIML-ECC, f_{jl} is a RF using the Gini split criteria, which greedily minimizes squared-loss $\mathcal{L}_2(y, p) = (y - p)^2$ on the training data [22]. If the entropy split criteria were used instead, the RF would greedily maximize likelihood. Gini and entropy are very similar for binary problems.

6.4.4 Asymptotic Complexity

MIML-ECC implemented with RF as the base SISL classifier is asymptotically efficient in all important dimensions of problem size. The size of a MIML dataset is determined by the number of bags n, the total number of instances in all bags m, the number of classes c, and the instance feature dimension d. MIML-ECC has several parameters which affect its runtime: the number of chains L, the number of trees in each RF T, and the number of support-instance updates K. Note that the runtime to train a RF on a SISL dataset of n instances with feature dimension d is $O(T(\log d)(n \log n))$, and to classify it is $O(\log n)$. It follows from the loop-structure of the pseudocode that the training time for MIML-ECC is

$$O\Big(LKT\big(m(\log n)(\log d) + cn \log n \log(d + c)\big)\Big) \tag{6.12}$$

Table 6.3: MIML datasets used in our experiments

Dataset	Classes	Dimension	Bags	Instances
MSRCv2	23	48	591	1,758
VOC 2012	20	48	1,053	4,142
Birdsong	13	38	548	4,998
Carroll	26	16	166	717
Frost	26	16	144	565

An efficient implementation of MIML-ECC classifies all instances in a bag at once, rather than treating each instance classification problem separately, in order to share redundant work. Using this optimization, the classification time is $O(LTc \log n)$ per instance. In Section 6.5.9 we provide empirical run time of our algorithm.

6.5 Experiments

Our experiments compare MIML-ECC to prior and baseline methods on two vision datasets, an audio dataset, and two artificial datasets. Our experimental setup is identical to the setup used in [17] and [99], hence results are directly comparable (e.g., the same features and folds for cross validation are used). Therefore we report new results for MIML-ECC and baseline methods, and compare to previously reported results from the aforementioned prior work.

6.5.1 Datasets

The datasets used in our experiments are summarized in Table 6.3. Datasets have been pre-processed through feature rescaling (which does not affect RF), to improve results for SVM style-classifiers, by the same process in [16, 17, 40].

6.5.1.1 Vision Datasets

We consider two vision datasets, Microsoft Research Cambridge v2 (MSRCv2) [154], and PASCAL Visual Object Recogntion Challenge (2012 "Segmentation") [57]. Both datasets contain images of objects with pixel-level labeling of regions. MSRCv2 provides a single class label

for each pixel. VOC provides a segmentation of each image into objects and a label for each object. Here bags are images labeled with a list of objects, instances are objects / regions of pixels, described by a 48-D feature vector. Single-label images are removed to make the learning problem more challenging.

6.5.1.2 Bioacoustics Dataset

This dataset was introduced by [18], applying a MIML formulation for label set prediction to a real-world application of classifying bird song collected in field conditions. Each bag is a 10 second audio recording labeled with the set of species it contains. Each instance is an utterance of bird sound obtained by an automatic segmentation algorithm. This dataset has also been used in work on MIML instance annotation and superset label learning [16, 17, 99]. For instance annotation, [16] introduced two variants of this dataset, "filtered" and "unfiltered." Our experiments use the filtered variant, as does [99].

6.5.1.3 Artificial Datasets

We use the same artificial MIML datasets as [16, 17], which are generated to simulate correlations between labels by using letter correlation in English words. The datasets are generated based on the words in two poems, "Jabberwocky" by Lewis Carroll [24], and "The Road Not Taken" by Robert Frost [66], hence they are referred to as Carroll and Frost. Each bag is a word, its letters are instances, and the bag label set is the union of instance labels. The instance features are sampled randomly from the UCI Letter Recognition dataset [65].

6.5.2 Prior & Baseline Methods

We compare MIML-ECC with a number of prior methods that can be applied to MIML instance annotation.

6.5.2.1 M^3MIML

Originally intended for label-set prediction, M^3MIML is a MIML support-vector machine algorithm, which builds one linear instance-level model per class by minimizing a heuristic relaxation

of bag-level hinge loss, and connecting instance labels with bag label sets by the max model. Although not intended for this purpose, the learned instance-level models can be used for instance annotation.

6.5.2.2 Rank-loss SIM

Rank-loss SIM was introduced by [16], and refers to a class of instance annotation algorithms which learn one linear instance-level model per class by minimizing a bag-level rank-loss objective. Different variants of rank-loss SIM consider different models for connecting bag-level output with instance-level outputs, and apply different procedures for optimizing the rank-loss objective. We consider SIM-Heuristic using a softmax model and SIM-CCCP with the max model, with random Fourier kernel features [119] to achieve nonlinear classification by approximating an RBF kernel. These models are chosen for comparison because they achieved the best accuracy in [17].

6.5.2.3 CLPL

Like the other SVM-style algorithms, Convex Learning from Partial Labels (CLPL) [40] learns one linear instance-level model per class, but uses an ALC formulation instead of MIML. CLPL minimizes a loss function which can be seen as an upper bound to the 0/1 loss on the true-unknown label, which is part of the candidate label set.

6.5.2.4 LSB-CMM

Logistic Stick-Breaking Conditional Multinomial Model (LSB-CMM) [99] is a recent hybrid generative / discriminative graphical model for SLL that have been used (by reduction) to solve the instance annotation problem. In particular, the same Birdsong and MSRCv2 datasets were used in [99] to evaluate its instance annotation accuracy. We compare to the results reported in [99] on these two datasets.

6.5.2.5 SISL Random Forest and SVM

We also consider SISL algorithms, which have an unfair advantage of learning directly from instance labels. Results with these SISL algorithms are presented for the inductive mode as an empirical upper bound on the accuracy that can be achieved on these datasets. For this comparison, we use a SISL RF (with 1000 trees), and refer to prior results from [17] with a SISL multi-class linear SVM.

6.5.3 Transductive and Inductive

In the transductive mode, there is no cross-validation (the whole dataset is used for training and testing). However, because MIML-ECC is a randomized algorithm, we run 10 repetitions and report the average accuracy $\pm$ the standard deviation over repetitions. Most of the other algorithms we compare to are not randomized, so in the transductive mode there is no uncertainty associated with the accuracy result.

In the inductive mode, we use 10-fold cross validation, except for the VOC dataset, for which there is a pre-specified partition into "train" and "val" sets. Results with 10-fold cross-validation are reported as average accuracy over all folds $\pm$ standard deviation in accuracy. A different random instantiation of MIML-ECC is used in each fold, so we do not run multiple repetitions on top of cross-validation. However, because there is only one fold for the VOC dataset, we report results $\pm$ standard deviation over 10 repetitions for MIML-ECC (and the randomized baseline method SISL Random Forest) on VOC.

M^{3}MIML, CLPL, and rank-loss SIM-Heuristic/CCCP all build one instance-level model per class $f_j(\mathbf{x})$. In the inductive mode, these models are used to predict an instance label by the rule $f(\mathbf{x}) = \arg\max_{j=1,\ldots,c} f_j(\mathbf{x})$. In the transductive mode, the rule is $f(\mathbf{x}, Y) = \arg\max_{j \in Y} f_j(\mathbf{x})$ (hence when the bag label set Y is known, it is used to constrain the instance-label predictions). This constraint provides some context for instance-label prediction, so there is not as much benefit to be had from looking at other instances in the transductive mode.

6.5.4 Parameter Selection

All of the rank-loss SIM algorithms, CLPL, M^{3}MIML, and SISL SVM have a regularization parameter (either λ or C). When random kernel features are used to approximate the RBF kernel, there is also a kernel parameter γ, and a parameter D which controls the approximation

accuracy. In prior work, these parameters are optimized post-hoc by a grid search as described in [17]. This means the experiment is run once for each parameter setting in a grid, and the best test accuracy over all parameters is reported. Post-hoc selection is not feasible without using instance labels to compute which parameter setting has the best accuracy, but it has been accepted in prior work on MIML instance annotation because it is an unsolved problem. Results using post-hoc selection can be interpreted as the highest accuracy that can be achieved using an oracle to select meta-parameters.

An important practical advantage of MIML-ECC compared to the above prior methods is that it does not have regularization parameters that must be tuned. Note that MIML-ECC has parameters L,K, and T. The accuracy of the algorithm tends to increase as these parameters increases up to a limit. So the parameter choices primarily depend on the time budget for training and testing. Our experiments set $L = 20, K = 20, T = 100$, which provides a good tradeoff between runtime and accuracy.

LSB-CMM [99] has some parameters which can affect accuracy, but in their experiments these parameters are set to standard values for all datasets.

6.5.5 Results

MIML instance annotation algorithms are evaluated based on accuracy, which is the fraction of correctly classified instances. These experiments compare multiple classifiers on multiple datasets, so following the recommendations of [48], we summarize results using wins, ties, and losses, and average ranks. Table 6.4 lists the accuracy and average rank results in transductive and inductive modes. Average ranks are computed by sorting the accuracy of MIML-ECC, and the prior methods M^{3}MIML, CLPL, SIM-Heuristic, and SIM-CCCP on each dataset, then averaging the position in the sorted list over all datasets. We do not include LSB-CMM in the ranking because there are only 2 datasets with comparable results.

In the inductive mode, MIML-ECC ties with SIM-CCCP max with RBF kernel on the Carroll dataset, and wins in all other comparisons. Results are not as decisive in the transductive mode, but MIML-ECC still achieves the best average rank over all datasets. This is consistent with our expectation because the known label sets provide a surrogate for context to the other algorithms.

It should be noted that due to the use of post-hoc selection in experiments for CLPL, M^{3}MIML and SIM, they are actually given an unfair advantage compared to MIML-ECC,

Table 6.4: Instance annotation accuracy results († – results from [17], ‡ – results from [99])

(a) Transductive accuracy ± standard deviation over 10 repetitions for MIML-ECC and SIM-RF

Algorithm	Carroll	Frost	Birdsong	MSRCv2	VOC	Avg Rank
Proposed Methods						
MIML-ECC ($L = 20, K = 20, T = 100$)	.803 ± .006	.831 ± .004	.779 ± .003	.805 ± .007	.624 ± .004	1.8
Prior Methods						
† CLPL	.672	.688	.742	.678	.598	4.0
† M^{3}MIML	.454	.532	.651	.547	.533	5.0
† SIM-CCCP `max` + kernel	.807	.780	.829	.798	.623	2.2
† SIM-Heuristic `softmax` + kernel	.794	.819	.833	.766	.634	2.0
Baseline Methods						
SIM-RF ($K = 20, T = 100$)	.763 ± .014	.787 ± .015	.791 ± .010	.799 ± .007	.618 ± .003	

(b) Inductive accuracy ± standard deviation over 10-fold cross validation or 10 repetitions for VOC

Algorithm	Carroll	Frost	Birdsong	MSRCv2	VOC	
Proposed Methods						
MIML-ECC ($L = 20, K = 20, T = 100$)	.618 ± .059	.646 ± .048	.666 ± .052	.611 ± .038	.43 ± .004	1
Prior Methods						
† CLPL	.464 ± .058	.506 ± .063	.620 ± .038	.431 ± .036	.345	3.6
† M^{3}MIML	.288 ± .041	.313 ± .041	.433 ± .073	.317 ± .055	.396	4.2
† SIM-CCCP `max` + kernel	.618 ± .042	.576 ± .065	.630 ± .040	.519 ± .044	.343	2.6
† SIM-Heuristic `softmax` + kernel	.596 ± .041	.587 ± .066	.642 ± .039	.506 ± .038	.337	2.8
‡ LSB-CMM	–	–	.715	.459	–	
Baseline Methods						
SIM-RF ($K = 20, T = 100$)	.522 ± .079	.589 ± .040	.645 ± .055	.575 ± .045	.444 ± .002	
MIML-ECC ($L = 1, K = 20, T = 2000$)	.530 ± .047	.598 ± .040	.644 ± .044	.580 ± .047	.425 ± .003	
SISL Methods (uses instance labels)						
† SISL SVM (multi-class,linear)	.772 ± .049	.753 ± .038	.772 ± .032	.638 ± .045	.440	
SISL Random Forest ($T = 1000$)	.809 ± .049	.807 ± .076	.805 ± .033	.729 ± .050	.511 ± .002	

which does not use the test data ground truth in training or parameter selection.

The comparison with LSB-CMM on two datasets is less conclusive. MIML-ECC outperforms LSB-CMM by a margin of 15.2% on the MSRCv2 dataset, but LSB-CMM is slightly better (by a margin of 4%) on the Birdsong dataset.

6.5.6 Ensemble of Chains vs. Binary Relevance (SIM-RF)

MIML-ECC is motivated by the idea that bag-level label correlations captured through the chain structure are useful for predicting instance labels. However, it is possible that the improved performance we observe compared to prior linear/kernel algorithms is not due to exploiting

label correlations, but instead to using a RF as the base-classifier. To address this hypothesis, we consider an additional comparison against a baseline that we call SIM-RF, which is the same as MIML-ECC in all details except it does not use a chain or model correlations. SIM-RF is equivalent to running MIML-ECC with one chain ($L = 1$) but omitting all of the concatenation of label set bits, i.e. $\oplus Y^{\pi 1:\pi(j-1)}$. SIM-RF is also equivalent to binary relevance with each class modeled by a MIL classifier which alternates between computing support instances and training an RF on them.

MIML-ECC achieves better accuracy than SIM-RF most of the time. The win-loss count is 4-1 in favor of MIML-ECC for both transductive and inductive modes. The comparison to SIM-RF suggests that the chain structure is actually critical, and the improved performance of MIML-ECC compared to prior methods cannot be attributed only to switching from a linear or kernel SVM classifier to RF.

6.5.7 Single Chain vs. Ensemble of Chains

We want to know how much benefit the ensemble provides compared to a single chain. The results we reported so far are obtained with $L = 20, K = 20, T = 100$, i.e., 20 chains and 100 trees and 20 iterations of support instance updates. To understand the impact of using mulitple chains with a fair comparison, we run MIML-ECC with one chain order ($L = 1$), and $K = 20, T = 2000$, so the total number of decision trees that vote on an instance label is the same. Table 6.4 (b) lists results for 1-chain MIML-ECC in the inductive mode (see Baseline Methods). In this comparison, MIML-ECC with multiple chains achieves higher accuracy on all datasets than MIML-ECC with a single chain. These results suggest that given a fixed time budget, it is better to have multiple chains, each with less trees, than a single chain with more trees. Recall that when predicting instance scores for class j, each chain can only use the presence/absence of other classes which come before j in the chain. Using multiple chains with random orders increases the chance that relevant classes are available for use as context (at least in some of the chains).

6.5.8 Comparison to SISL

SISL methods achieve better accuracy in inductive experiments than MIML instance annotation, ALC and SLL (Table 6.4 (b)), which is expected because they are trained on unambigu-

Table 6.5: Runtime for training and classification with MIML-ECC, per fold of cross-validation or per repetition (seconds)

Mode	Carroll	Frost	Birdsong	MSRCv2	VOC
Transductive	104.9	84.4	251.8	304.5	798.0
Inductive	69.4	57.8	135.5	202.8	2895.8

ously labeled instances. This improved accuracy must be weighed against the greater human effort required to obtain instance labels compared to bag label sets.

6.5.9 Empirical Runtime

Table 6.5 lists empirical runtimes for training plus classification with MIML-ECC (with $L = 20, K = 20, T = 100$), on each dataset, averaged over the number of repetitions or folds of cross-validation. The runtime is on the order of seconds or minutes for all datasets. In our experiments, training is parallelized using threads, and classification is done sequentially.[2]

6.6 Related Work

Graphical models for MIML sometimes include instance labels as hidden variables. Inference over these hidden variables can be used for instance annotation. In addition to LSB-CMM, some recent examples of graphical models for MIML include Dirichlet-Bernoulli Alignment [160] and Exponential Multinomial Mixture model [161]. [165] proposed MLMIL, a conditional random field for MIML which uses Gibbs sampling to infer instance labels.

[148] developed a MIML SVM algorithm which uses a bag-level kernel. Their algorithm predicts instance labels by applying the bag-level classifier to a bag of one instance. [147] proposed a MIML instance annotation algorithm which alternates between sampling random instance labels and training a Semantic Texton Forest (a specialization of RF to images). [115] proposed a MIML algorithm which alternates between assigning instance labels and training a maximum margin classifier. [96] considers the problem of selecting a set of instances explaining each label, which is different from instance annotation, where the goal is to label all instances.

[2]Code is C++ compiled with GCC 4.0 (most speed optimizations enabled). Experiments ran on a Mac Pro with 2x 2.4 GHz Quad-Core Intel Xeon processor and 16 GB 1066 MHz DDR3 memory, with OS X 10.8.1.

Prior work on multi-instance (single-label) learning has considered the case where instance labels are not independent, and encoded instance-label relationships through a graph [95, 173]. These approaches can be viewed as a different way to model instance-label correlations.

Several formulations besides the max model have been used for MIL and MIML to relate instance and bag labels. Different formulations encode different assumptions about instance labels. One version of the Diverse Density algorithm for MIL [109] used a Noisy-OR model $P(y = 1|B,\theta) = 1 - \prod_{\mathbf{x}\in B}\Big(1 - P(y = 1|\mathbf{x},\theta)\Big)$. [110] points out that the max model makes fewer independence assumptions than the Noisy-OR model, although both generate similar probabilities in many cases. In later work the EM-DD [169] algorithm replaced Noisy-OR with max. [121] proposed Multiple-Instance Logistic Regression, which uses a smooth softmax approximation to max. [16, 17] used a multi-class softmax model. [156] propose a model where the bag-label probability is the average of the instance-label probabilities.

6.7 Conclusion & Future Work

We proposed MIML-ECC, an algorithm for context-aware MIML instance annotation. Experiments on image, audio, and artificial datasets show that MIML-ECC achieves better accuracy than other recent algorithms.

MIML-ECC exploits context through correlations, which can be summarized by statements like "if A is present, B is also likely to be present." However, MIML-ECC cannot exploit a different kind of context, which can be summarized as "if one A is present, there are likely to be more A's." For example, consider Fig 6.2. It might be easy to recognize some of the larger cows in the image, but harder to recognize the small ones. However, after recognizing one cow, it we might expect to find more cows. MIML-ECC will not exploit this kind of context because it can only use information about the presence or absence of other classes to inform its prediction.

Chapter 7: Conclusion & Future Work

This chapter concludes with a summary of the capabilities enabled by the work in this thesis, and discusses new directions for future work.

7.1 Conclusion

This thesis presented several new algorithms for multi-instance and/or multi-label supervised classification. The proposed methods make significant progress toward the goal of using machine learning for automatic unattended monitoring of bird populations. Several of the proposed algorithms are general, and can also be applied to other domains such as object recognition in images.

One of the basic challenges in classifying audio is that many features characterize frames (very short intervals), but it is often more convenient to use a feature vector that summarizes a recording as a whole. This situation necessitates a method for aggregating frame-level features. Many prior works simply average frame-level features, which is a poor representation of multi-model feature distributions. In chapter 3, we developed a probability model in which audio recordings are viewed as a collection of frame-level features, which are i.i.d. samples from a distribution specific to each recording (described by a histogram). We showed that in this model, the MAP classifier is well approximated by a nearest-neighbor classifier using KL-divergence between the histograms. This approach provides a principled way to perform supervised classification while treating a recording as a collection of frame-level features (multi-instance), and is applicable to recordings containing a single species of bird (single-label).

In a real-world acoustic monitoring scenario, recordings containing multiple simultaneously vocalizing bird species are common. In chapter 4, we developed one of the first systems that can handle this situation. In contrast with prior work, where the goal was either to associate a single species with a single frame, syllable, or whole recording, we pose the problem as predicting the *set* of species that are present in a recording. We identified MIML as a suitable framework for this problem, and developed a MIML representation of bird sound, which can separate calls that overlap in time.

Most prior work on MIML focusses on predicting the label set for a new bag. In chapters 5 and 6, we formalize MIML instance annotation, and propose two new algorithms for it, namely rank-loss SIM and MIML-ECC. In MIML instance annotation, the goal is to predict the labels of instances, while training only on bags paired with label sets. This formulation enables a prediction of the species label responsible for each individual call in a recording. It is possible to predict species labels for individual calls using SISL, but doing so requires labeled calls as training data, which are very labor intensive and error-prone to obtain. In contrast, the MIML instance annotation approach facilitates this kind of prediction, while making efficient use of human labeling effort, by training only on recordings paired with the set of species they contain.

7.2 Future Work

Acoustic monitoring of birds presents many challenges that can serve as inspiration for research in machine learning. Further investigation of this area will likely lead not only to improved methods for monitoring, but also new general-purpose machine learning algorithms. In this section, we discuss several directions that we believe to be promising for future research, with emphasis on what can be done for the HJA dataset and other similar data.

7.2.1 Efficient Use of Human Effort

Supervised classification of bird sound requires labeled examples for training data. Such examples are expensive to obtain because the labelers must have considerable expertise (often years of experience). Furthermore, in an automatic monitoring scenario, it is easy to collect a vast amount of data (e.g., we collected 2559 hours of audio with Songmeters in HJA in 2009 alone), so it is only feasible to label a small fraction of the available data. Therefore it is important to consider the "human-in-the-loop" nature of the problem, and maximize the value of human effort at every step. There are several frameworks in machine learning that can be applied to achieve these objectives. In many cases, the SISL version of these problems have been studied extensively, but the MIML counterpart is relatively open.

At the earliest stage, there is no available training data. In this situation, it is desirable to apply fully unsupervised methods, and high-volume visualization, in order to extract as much useful information as possible while expending little or no human effort. For example, we divided all of the data collected in HJA in 2009 into ten-second intervals, then applied the

segmentation and feature extraction methods discussed in chapter 4[1] to 10% chosen randomly without replacement, resulting 92,095 intervals. This process extracted 218,321 segments, each characterized by a 38-dimensional feature vector. We then applied k-means clustering to the segments with $k = 250$, and generated a visualization of the segments in each cluster. The visualizations of each cluster can be viewed very quickly compared with labeling a standard supervised learning dataset. With only a few minutes of human effort, we identified one of the clusters as consisting mostly of calls from a Varied Thrush (Fig. 7.1). For comparison, Fig. 7.2 shows manually identified calls of a Varied Thrush. Having identified a set of segments as (mostly) belonging to a particular species, we can then count these segments as a function of when they occurred. Fig. 7.3 shows peaks in Varied Thrush activity around dawn and dusk. The significance of this result is that we are able to obtain ecologically meaningful information from a large volume of raw data, with a very low expenditure of human effort. Further investigation is needed to derive additional useful information from this unsupervised/visualization-based approach.

Although unsupervised methods can be useful, it is still necessary to apply supervised classification to get a complete characterization of acoustic events in the data. Two frameworks in machine learning are relevant to the situation where there is no available training data. One is initial set selection, where the goal is to pick a batch of data to be labeled in such a way that a classifier trained on the data achieves the best accuracy possible. The initial set selection problem has received little study. The most widely used method of initial set selection in active learning literature is to select k examples randomly from each class [7, 72, 76, 77, 82, 103, 141, 158]. A similar method is to select k examples randomly, and ensure there is at least one from each class [21, 27]. However, in a real world problem where all labels are unknown, it is not possible to select exactly k from each class, or even one from each class, without requiring more effort than simply inspecting k examples. Another method of initial set selection that has been previously employed is to divide the dataset into k clusters (using k-means or a similar algorithm), then select one representative example from each cluster that is closest to the cluster center [46, 86]. Initial set selection has been studied primarily under the SISL formulation, and has not been considered for MIML.

Similar to initial set selection is the problem of class discovery (or in the acoustic monitoring context, species discovery). The idea is to select a fixed number of examples (e.g., 100 ten-second recordings) from a larger dataset, to find as many classes/species as possible. This

[1] A slight modification to the segmentation algorithm was introduced to better handle segmentation in rain.

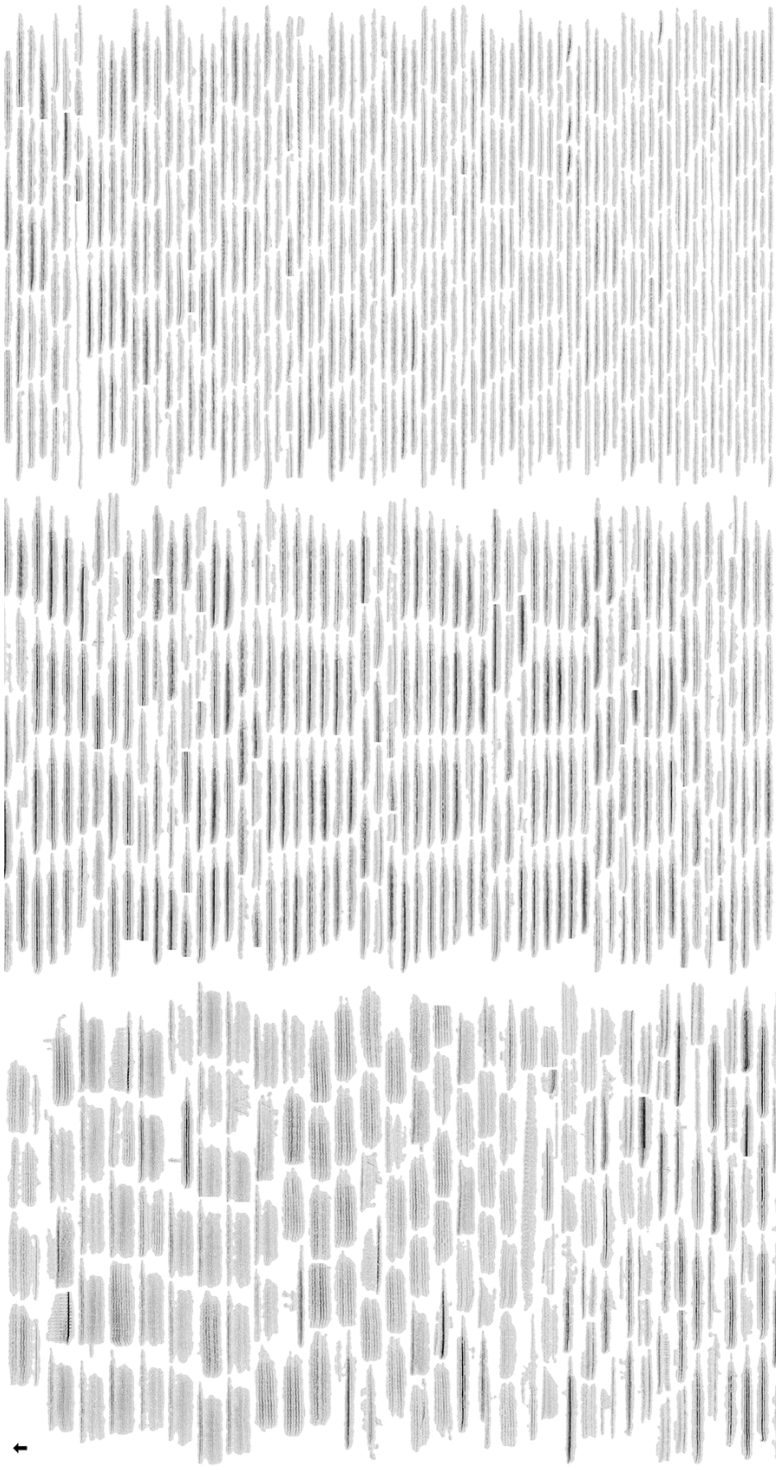


Figure 7.1: Visualization of all segments from cluster 231 in k-means++ with $k = 250$. This cluster corresponds to calls from a Varied Thrush. Look at the image with the ↑ pointing up.

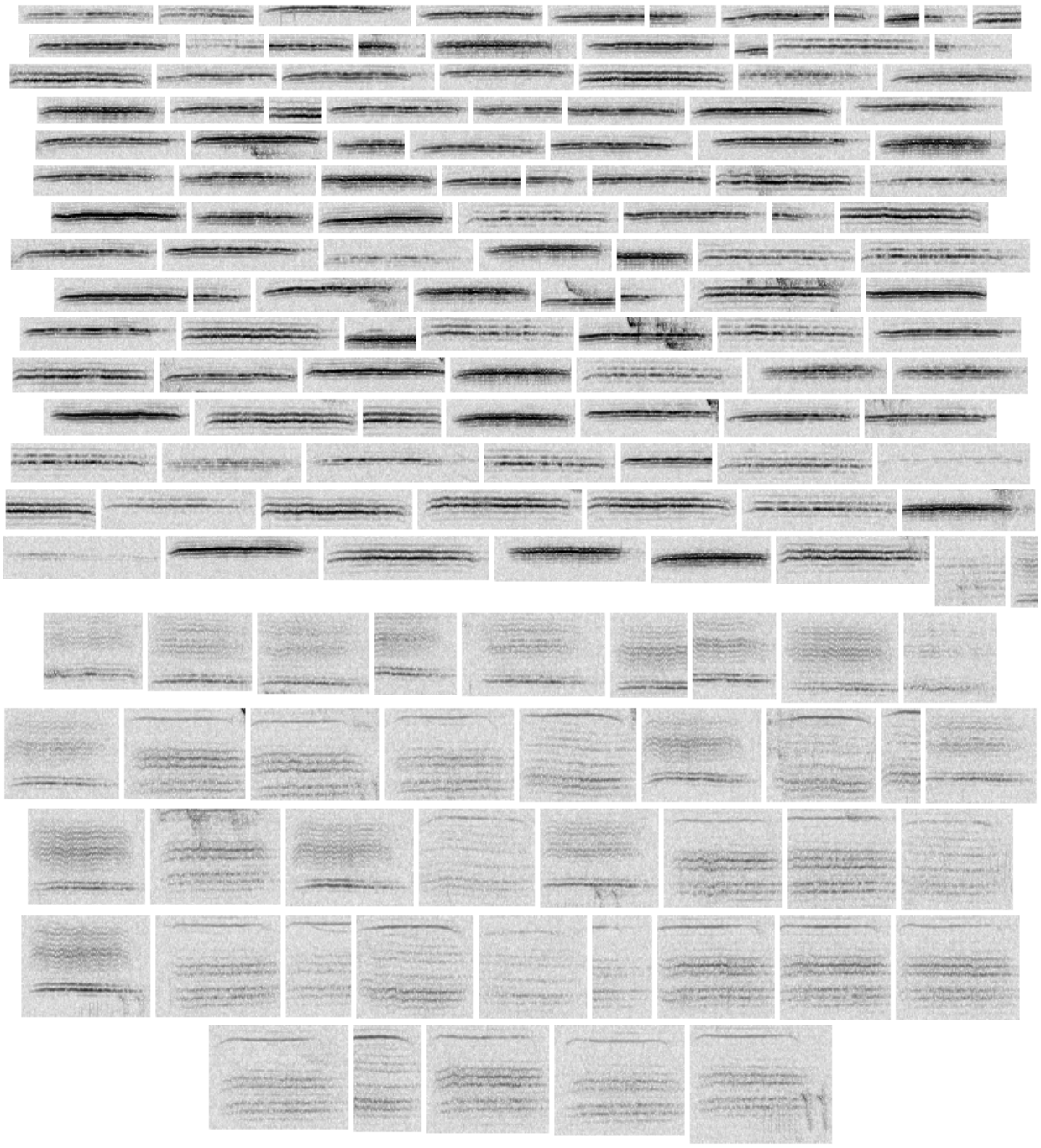


Figure 7.2: Manually identified calls from Varied Thrush.

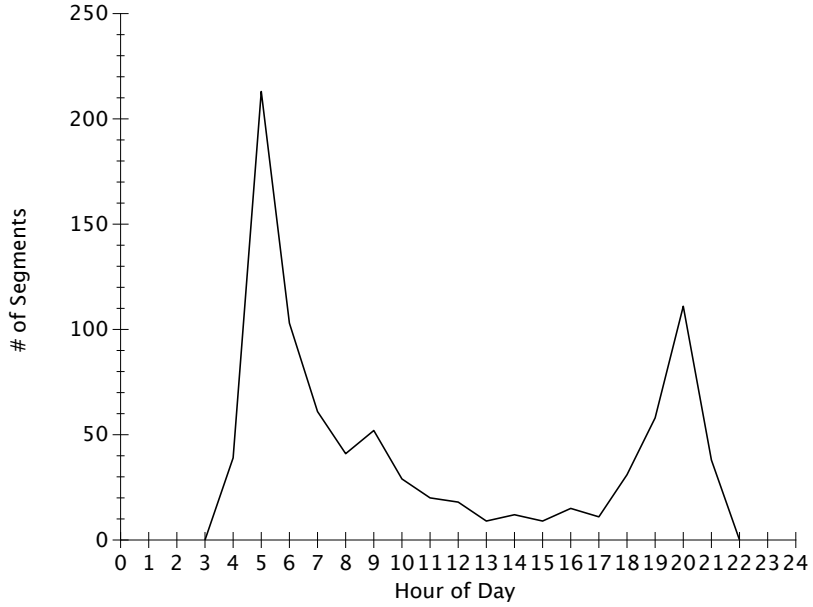


Figure 7.3: Segments from cluster 231 (identified as Varied Thrush), per hour of day.

problem is particularly important, because in a large collection of audio, the number of classes is not necessarily known in advance, but most supervised classification systems assume that there are some training examples available for all classes that will be encountered in the data. Furthermore, some classes/species may be much rarer than others, and the rarest are typically the most interesting from a conservation perspective. The species discovery problem has been studied extensively in ecology [107], and is closely related to the important problem of biodiversity estimation. Some work in machine learning addresses class discovery under the SISL framework, e.g. [29], which uses clustering and farthest-point-first traversals, but the problem has not been studied with MIML.

The need for labeled training examples continues after an initial set has been selected. A good human-in-the-loop system will continue to prompt experts for additional training data to improve accuracy on examples that the classifier is unable to label with high certainty. This process is known as active learning, and has been studied extensively under the SISL framework. However, there has been relatively little work on active learning for MIML, with the exception of [148].

Despite efforts in initial selection, class discovery, and active learning, with a large enough

dataset, it is inevitable that some recordings will contain classes which are not present in the training data. It is preferable to be able to identify these cases, rather than misclassifying them as one of the known classes. This situation motivates the problem of discovering when a bag contains instances of a class not present in the training data [102].

7.2.2 Exploiting Context

Current state-of-the-art methods for supervised classification often suffer in accuracy when applied to problems with a large number of classes. This is to be expected because the problem becomes more difficult as more classes are added. It is estimated that there are roughly 60 species of bird in HJA, although our experiments focussed on the 13 most common species at dawn. In order to achieve reliable classification with a larger number of species, we suggest that it will be useful to develop classifiers which exploit context. In chapter 6, we proposed MIML-ECC, which exploits one kind of context, namely the correlations between label sets. Another kind of context that MIML-ECC does not exploit is repetitions of instances from the same class within a bag (e.g., several repeated calls from the same bird). Some work has been done on MIML algorithms to exploit this latter kind of context [95, 173], based on the observation that instances within a bag are not independent.

In this thesis, we argued that reducing MIML problems to MLC is undesirable because doing so discards useful structure which can improve classification accuracy. However, some contextual information does not naturally fit in instance feature vectors. For example, given an audio recording, we may know the time and location where it was recorded. This information characterizes the bag as a whole, rather than a particular instance. Recently, participants in the MLSP 2013 bird classification challenge (where the goal was to predict the set of bird species present in recordings from HJA) widely reported that using time and location in their feature vectors significantly improved classification accuracy [20]. We suggest a generalization of MIML which adds a bag-level context feature vector in addition to the instance features. Specifically, the for this version of MIML the dataset is

$$(B_1, Y_1, \mathbf{z}_1), \ldots, (B_n, Y_n, \mathbf{z}_n) \tag{7.1}$$

In this definition, B_i and Y_i are the standard bags of instances and label sets, and $\mathbf{z}_i$ is a corresponding bag-level feature vector of contextual information that does not naturally fit into

instance features. We pose a challenge for future work: learn from such a dataset without resorting to a reduction to MLC, or to appending $\mathbf{z}_i$ to each instance feature vector (the former approach was used in [20], and the latter was used in chapter 6).

7.2.3 Representation of Bird Sound

Improvements in species classification accuracy can be obtained by developing better representations and features at both the instance and bag level. For example, we proposed a "histogram of segments" feature vector [19], which reduces a MIML bird sound dataset to an MLC dataset. Subsequently, many participants in the MLSP competition [20] extended this feature vector with additional bag-level features, and obtained improved classification accuracy over a baseline method using the histogram of segments features alone. Some useful bag/recording-level features are intended to help a classifier differentiate between bird sound and noise in cases where segmentation is inaccurate, or to capture the context provided by the time and location of a recording.

Fagerlund [59] noted that bird classification systems which begin with segmentation are highly sensitive to segmentation errors, because such errors propagate to all subsequent stages of processing. In preliminary experiments on a larger portion of the HJA dataset, we have seen that one particularly troublesome case for the segmentation algorithm proposed in chapter 4 is rain. Hence, we suggest that further work should focus on improving segmentation for recordings where rain is present. In [20], we introduced a modification to the segmentation algorithm which is designed to better handle rain, which is somewhat effective, but still requires refinement.

Although our proposed segmentation method can handle bird calls that overlap in time but not frequency, calls that overlap in both time and frequency are grouped together into a single segment. It would be very useful (and probably difficult) to separate such calls. It may be advantageous to use stereo rather than mono audio for this problem.

The majority of work on audio classification of bird species focusses on feature and representation engineering, while using relatively well-established algorithms for supervised classification. Hence, one could argue that most of the intelligence in the systems comes from the human designer of the features. One of the highest-accuracy methods in the MLSP competition [20] used convolutional deep belief nets. This result is striking because more intelligence comes from the learning algorithm itself (i.e. a human did not have to extensively engineer

features). Hence, further investigation of representation learning and deep learning methods for bird sound is strongly motivated.

Bibliography

[1] M.A. Acevedo, C.J. Corrada-Bravo, H. Corrada-Bravo, L.J. Villanueva-Rivera, and T.M. Aide. Automated classification of bird and amphibian calls using machine learning: A comparison of methods. *Ecological Informatics*, 4(4):206–214, 2009.

[2] T Mitchell Aide, Carlos Corrada-Bravo, Marconi Campos-Cerqueira, Carlos Milan, Giovany Vega, and Rafael Alvarez. Real-time bioacoustics monitoring and automated species identification. *PeerJ*, 1:e103, 2013.

[3] S. E. Anderson, A. S. Dave, and D. Margoliash. Template-based automatic recognition of birdsong syllables from continuous recordings. *J. Acoust. Soc. Am.*, 100(2):1209–1219, 1996. ISSN 0001-4966.

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

[5] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics. ISBN 9780898716245.

[6] A. Balmford, R.E. Green, and M. Jenkins. Measuring the changing state of nature. *Trends in Ecology & Evolution*, 18(7):326–330, 2003. ISSN 0169-5347.

[7] Y. Baram, R. El-Yaniv, and K. Luz. Online Choice of Active Learning Algorithms. *The Journal of Machine Learning Research*, 5:255–291, 2004.

[8] R. Bardeli, D. Wolff, F. Kurth, M. Koch, K.H. Tauchert, and K.H. Frommolt. Detecting bird sounds in a complex acoustic environment and application to bioacoustic monitoring. *Pattern Recognition Letters*, 2009. ISSN 0167-8655.

[9] S. Bastas, M. Wadood Majid, G. Mirzaei, J. Ross, M.M. Jamali, P.V. Gorsevski, J. Frizado, and V.P. Bingman. A novel feature extraction algorithm for classification of bird flight calls. In *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*, pages 1676–1679. IEEE, 2012.

[10] M. G. Betts, D. Mitchell, A. W. Diamond, and J. Bêty. Uneven rates of landscape change as a source of bias in roadside wildlife surveys. *J. Wildlife Management*, 71(7):2266–2273, 2007. ISSN 1937-2817.

[11] Matthew G. Betts, A.W. Diamond, G.J. Forbes, M.-A. Villard, and J.S. Gunn. The importance of spatial autocorrelation, extent and resolution in predicting forest bird occurrence. *Ecological Modelling*, 191(2):197 – 224, 2006. ISSN 0304-3800.

[12] A.P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[13] T.S. Brandes. Feature vector selection and use with hidden Markov models to identify frequency-modulated bioacoustic signals amidst noise. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(6):1173–1180, 2008. ISSN 1558-7916.

[14] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[15] F. Briggs, R. Raich, and X.Z. Fern. Audio classification of bird species: a statistical manifold approach. In *9th IEEE Int. Conf. on Data Mining*, pages 51–60, 2009.

[16] F. Briggs, X.Z. Fern, and R. Raich. Rank-loss support instance machines for miml instance annotation. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 534–542. ACM, 2012.

[17] F. Briggs, X.Z. Fern, R. Raich, and Q. Lou. Instance annotation for multi-instance multi-label learning. *To appear in Transactions on Knowledge Discovery from Data (TKDD), 2012*, 2012.

[18] F. Briggs, B. Lakshminarayanan, L. Neal, X.Z. Fern, R. Raich, S.J.K. Hadley, A.S. Hadley, and M.G. Betts. Acoustic classification of multiple simultaneous bird species: A multi-instance multi-label approach. *The Journal of the Acoustical Society of America*, 131: 4640, 2012.

[19] F. Briggs, X. Z. Fern, and J. Irvine. Multi-Label Classifier Chains for Bird Sound. *ICML 2013 Workshop on Machine Learning for Bioacoustics*, 2013.

[20] F. Briggs et al. The 9th Annual MLSP Competition: New Methods for Acoustic Classification of Multiple Simultaneous Bird Species in a Noisy Environment. In *IEEE Workshop on Machine Learning for Signal Processing, MLSP 2013*, 2013.

[21] K. Brinker. Incorporating Diversity in Active Learning with Support Vector Machines. In *Machine Learning-International Workshop then Conference*, volume 20, pages 59–66, 2003.

[22] Andreas Buja, Werner Stuetzle, and Yi Shen. Loss functions for binary class probability estimation and classification: Structure and applications. Technical report, 2005.

[23] J. Cai, D. Ee, B. Pham, P. Roe, and J. Zhang. Sensor network for the monitoring of ecosystem: bird species recognition. In *3rd Int. Conf. Intelligent Sensors, Sensor Networks and Information, 2007*, pages 293–298, 2008.

[24] Lewis Carroll. *Through the looking-glass: and what Alice found there*. 1896.

[25] C.K. Catchpole, P.J.B. Slater, and N. Mann. *Bird song: biological themes and variations*. Cambridge Univ Press, 2003.

[26] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001.

[27] E. Chang and S. Tong. SVMActive-Support Vector Machine Active Learning for Image Retrieval. In *Proceedings of the 9th ACM International Conference on Multimedia*, pages 107–118, 2001.

[28] Russell A Charif and Michael Pitzrick. Automated detection of cerulean warbler songs using xbat data template detector software. *Preliminary Report. Bioacoustics Research Program Cornell Laboratory of Ornithology Technical Report*, pages 08–02, 2008.

[29] Yang Chen, Alex Groce, Chaoqiang Zhang, Weng-Keen Wong, Xiaoli Fern, Eric Eide, and John Regehr. Taming compiler fuzzers. In *PLDI*, pages 197–208, 2013.

[30] Z. Chen and R. C. Maher. Semi-automatic classification of bird vocalizations using spectral peak tracks. *J. Acoust. Soc. Am.*, 120(5 Pt 1):2974–2984, November 2006. ISSN 0001-4966.

[31] J. Cheng, Y. Sun, and L. Ji. A call-independent and automatic acoustic system for the individual recognition of animals: A novel model using four passerines. *Pattern Recognition*, 43(11):3846–3852, 2010.

[32] J. Cheng, B. Xie, C. Lin, and L. Ji. A comparative study in birds: call-type-independent species and individual recognition using four machine-learning methods and two acoustic features. *Bioacoustics*, 21(2):157–171, 2012.

[33] ED Chesmore and E. Ohya. Automated identification of field-recorded songs of four British grasshoppers using bioacoustic signal recognition. *Bulletin of Entomological Research*, 94(04):319–330, 2004. ISSN 0007-4853.

[34] P.M. Cheung and J.T. Kwok. A regularization framework for multiple-instance learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 193–200. ACM, 2006.

[35] C.H. Chou and P.H. Liu. Bird species recognition by wavelet transformation of a section of birdsong. In *Ubiquitous, Autonomic and Trusted Computing, 2009. UIC-ATC'09. Symposia and Workshops on*, pages 189–193. IEEE, 2009.

[36] C.H. Chou, P.H. Liu, and B. Cai. On the studies of syllable segmentation and improving mfccs for automatic birdsong recognition. In *Asia-Pacific Services Computing Conference, 2008. APSCC'08. IEEE*, pages 745–750. IEEE, 2008.

[37] W. Chu and D.T. Blumstein. Noise robust bird song detection using syllable pattern-based hidden Markov models. *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*, 2011.

[38] Corinna Cortes and Vladimir Vapnik. Support vector networks. In *Machine Learning*, pages 273–297, 1995.

[39] T. Cour, B. Sapp, C. Jordan, and B. Taskar. Learning from ambiguously labeled images. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 919–926. IEEE, 2009.

[40] T. Cour, B. Sapp, and B. Taskar. Learning from partial labels. *Journal of Machine Learning Research*, 12:1225–1261, 2011.

[41] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision, ECCV*, volume 1, page 22, 2004.

[42] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conf. Computer Vision and Pattern Recognition*, volume 1, pages 886–893, 2005. ISBN 0769523722.

[43] Theodoros Damoulas, Samuel Henry, Andrew Farnsworth, Michael Lanzone, and Carla Gomes. Bayesian classification of flight calls with a novel dynamic time warping kernel. In *Machine Learning and Applications (ICMLA), 2010 Ninth International Conference on*, pages 424–429. IEEE, 2010.

[44] Anirban DasGupta. Asymptotic theory of statistics and probability. *International Statistical Review*, 77(1):160–161, 04 2009.

[45] Steven B. Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *Readings in Speech Recognition*, pages 65–74, 1990.

[46] D. DeBarr and H. Wechsler. Spam Detection using Clustering, Random Forests, and Active Learning. In *Conference on Email and Anti-Spam*, 2009.

[47] Krzysztof Dembczynski, Weiwei Cheng, and Eyke Hüllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 279–286, 2010.

[48] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.

[49] T.G. Dietterich, R.H. Lathrop, and T. Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71, 1997.

[50] A. Dimou, G. Tsoumakas, V. Mezaris, I. Kompatsiaris, and I. Vlahavas. An empirical study of multi-label learning methods for video annotation. In *7th Int. Workshop on Content-Based Multimedia Indexing*, pages 19–24, 2009.

[51] L. Du, L. Ren, D. Dunson, and L. Carin. A bayesian model for simultaneous image clustering, annotation and object segmentation. *Advances in Neural Information Processing Systems*, 22:486–494, 2009.

[52] Ping Du and Todd W. Troyer. A segmentation algorithm for zebra finch song at the note level. In *Neurocomputing*, volume 69, pages 1375–1379, 2006.

[53] S. Duan, M. Towsey, J. Zhang, A. Truskinger, J. Wimmer, and P. Roe. Acoustic component detection for automatic species recognition in environmental monitoring. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2011 Seventh International Conference on*, pages 514–519. IEEE, 2011.

[54] Yasser EL-Manzalawy and Vasant Honavar. *WLSVM: Integrating LibSVM into Weka environment*, 2005.

[55] A. Elisseeff and J. Weston. A kernel method for multi-labelled classification. *Advances in Neural Information Processing Systems*, 14:681–687, 2001.

[56] C. Elkan. Clustering documents with an exponential-family approximation of the Dirichlet compound multinomial distribution. In *Proceedings of the 23rd international conference on Machine learning*, pages 289–296, 2006.

[57] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2): 303–338, June 2010.

[58] S. Fagerlund and A. Härmä. Parametrization of inharmonic bird sounds for automatic recognition. In *13th European Signal Processing Conference (EUSIPCO 2005)*, pages 4–8, 2005.

[59] Seppo Fagerlund. *Automatic recognition of bird species by their sounds*. PhD thesis, Helsinki University of Technology, 2004.

[60] Seppo Fagerlund. Bird species recognition using support vector machines. *EURASIP Journal on Advances in Signal Processing*, 2007, 2007.

[61] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[62] Andrew Farnsworth and Robert W. Russell. Monitoring flight calls of migrating birds from an oil platform in the northern gulf of mexico. *Journal of Field Ornithology*, 78(3): pp. 279–289, 2007. ISSN 02738570. URL `http://www.jstor.org/stable/40345964`.

[63] Andrew Farnsworth, Sidney A Gauthreaux Jr, and Donald van Blaricom. A comparison of nocturnal call counts of migrating birds and reflectivity measurements on doppler radar. *Journal of Avian Biology*, 35(4):365–369, 2004.

[64] J. Foulds and E. Frank. A review of multi-instance learning assumptions. *The Knowledge Engineering Review*, 25(01):1–25, 2010.

[65] Peter W Frey and David J Slate. Letter recognition using holland-style adaptive classifiers. *Machine Learning*, 6:161, 1991.

[66] R. Frost. *Mountain Interval*. 1916.

[67] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.

[68] Todor Ganchev, Nikos Fakotakis, and George Kokkinakis. Comparative evaluation of various mfcc implementations on the speaker verification task. In *in Proc. of the SPECOM-2005*, pages 191–194, 2005.

[69] Corrado Gini. Variability and mutability, contribution to the study of statistical distributions and relations. *J. Am. Statistical Assoc.*, 66:534–544, 1971.

[70] N. Giret, P. Roy, A. Albert, F. Pachet, M. Kreutzer, and D. Bovet. Finding good acoustic features for parrot vocalizations: The feature generation approach. *The Journal of the Acoustical Society of America*, 129:1089, 2011.

[71] L. Girod and M.A. Roch. An overview of the use of remote embedded sensors for audio acquisition and processing. In *Multimedia, 2006. ISM'06. Eighth IEEE International Symposium on*, pages 567–574. IEEE, 2006.

[72] Y. Guo and D. Schuurmans. Discriminative Batch Mode Active Learning. *Proceedings of Advances in Neural Information Processing Systems*, 6, 2007.

[73] A. Härmä. Automatic identification of bird species based on sinusoidal modeling of syllables. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, volume 5, pages V–545–8 vol.5, 2003.

[74] A. Härmä and P. Somervuo. Classification of the harmonic structure in bird vocalization. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, volume 5, pages V–701–4 vol.5, 2004.

[75] J.R. Heller and J.D. Pinezich. Automatic recognition of harmonic bird sounds using a frequency track extraction algorithm. *The Journal of the Acoustical Society of America*, 124:1830, 2008.

[76] S.C.H. Hoi, R. Jin, J. Zhu, and M.R. Lyu. Batch Mode Active Learning and its Application to Medical Image Classification. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 417–424. ACM New York, NY, USA, 2006.

[77] S.C.H. Hoi, R. Jin, and M.R. Lyu. Batch Mode Active Learning with Applications to Text Categorization and Image Retrieval. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1233, 2009.

[78] C.W. Hsu and C.J. Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

[79] C.W. Hsu, C.C. Chang, C.J. Lin, et al. A practical guide to support vector classification, 2003.

[80] E. Hüllermeier and J. Beringer. Learning from ambiguously labeled examples. *Intelligent Data Analysis*, 10(5):419–439, 2006.

[81] S. Images. Transductive Multi-Instance Multi-Label learning algorithm with application to automatic image annotation. *Expert Systems with Applications*, 2009.

[82] P. Jain and A. Kapoor. Active Learning for Large Multi-class Problems. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2009.

[83] P. Jancovic and M. Köküer. Automatic detection and recognition of tonal bird sounds in noisy environments. *J. Adv. Signal Processing*, 2011, 2011. ISSN 1687-6172.

[84] C.F. Juang and T.M. Chen. Birdsong recognition using prediction-based recurrent neural fuzzy networks. *Neurocomputing*, 71(1):121–130, 2007.

[85] F. Jurie and B. Triggs. Creating efficient codebooks for visual recognition. In *Tenth IEEE International Conference on Computer Vision, 2005. ICCV 2005*, volume 1, 2005.

[86] J. Kang, K.R. Ryu, and H.C. Kwon. Using Cluster-Based Sampling to Select Initial Training Set for Active Learning in Text Classification. *Lecture Notes in Computer Science*, pages 384–388, 2004.

[87] R.E. Kass and P.W. Vos. *Geometrical foundations of asymptotic inference*. Wiley-Interscience, 1997.

[88] Joseph A. Kogan and Daniel Margoliash. Automated recognition of bird song elements from continuous recordings using dynamic time warping and hidden markov models: A comparative study. *The Journal of the Acoustical Society of America*, 103(4):2185–2196, 1998. doi: http://dx.doi.org/10.1121/1.421364.

[89] C. Kwan, G. Mei, X. Zhao, Z. Ren, R. Xu, V. Stanford, C. Rochet, J. Aube, and K.C. Ho. Bird classification algorithms: Theory and experimental results. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages 289–292, 2004.

[90] B. Lakshminarayanan, R. Raich, and X. Fern. A Syllable-Level Probabilistic Framework for Bird Species Identification. In *2009 International Conference on Machine Learning and Applications*, pages 53–59. IEEE, 2009.

[91] C.H. Lee, C.C. Han, and C.C. Chuang. Automatic classification of bird species from their sounds using two-dimensional cepstral coefficients. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(8):1541–1550, 2008.

[92] Chang-Hsing Lee, Yeuan-Kuen Lee, and Ren-Zhuang Huang. Automatic recognition of bird songs using cepstral coefficients. *Journal of Information Technology and Applications*, 1(1):17 – 23, 2006.

[93] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Ng. Efficient sparse coding algorithms. In *Advances in neural information processing systems*, pages 801–808, 2006.

[94] D.D. Lewis. Evaluating text categorization. In *Proc. Speech and Natural Language Workshop*, pages 312–318, 1991.

[95] Bing Li, Weihua Xiong, and Weiming Hu. Web horror image recognition based on context-aware multi-instance learning. In *International Conference on Data Mining*, pages 1158–1163, 2011.

[96] Y.F. Li, J.H. Hu, Y. Jiang, and Z.H. Zhou. Towards discovering what patterns trigger what labels. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

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

[98] Thomas Lidy and Andreas Rauber. Evaluation of feature extractors and psycho-acoustic transformations for music genre classification. In *Proceedings of the Sixth International Conference on Music Information Retrieval (ISMIR 2005)*, pages 34–41, 2005.

[99] Liping Liu and Thomas Dietterich. A conditional multinomial mixture model for superset label learning. In *Advances in Neural Information Processing Systems 25*, pages 557–565, 2012.

[100] M.T. Lopes, L.L. Gioppo, T.T. Higushi, C.A.A. Kaestner, CN Silla, and A.L. Koerich. Automatic bird species identification for large number of species. In *Multimedia (ISM), 2011 IEEE International Symposium on*, pages 117–122. IEEE, 2011.

[101] M.T. Lopes, A. Lameiras Koerich, C. Nascimento Silla, and CA Alves Kaestner. Feature set comparison for automatic bird species identification. In *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*, pages 965–970. IEEE, 2011.

[102] Qi Lou, Forrest Briggs, Xiaoli Fern, and Raviv Raich. Novelty Detection Under Multi-Label Multi-Instance Framework. In *IEEE Workshop on Machine Learning for Signal Processing, MLSP 2013*, 2013.

[103] T. Luo, K. Kramer, D.B. Goldgof, L.O. Hall, S. Samson, A. Remsen, and T. Hopkins. Active Learning to Recognize Multiple Types of Plankton. *Journal of Machine Learning Research*, 6(1):589, 2006.

[104] D.A. Luther. Signaller: receiver coordination and the timing of communication in Amazonian birds. *Biology Letters*, 4(6):651, 2008. ISSN 1744-9561.

[105] D.A. Luther and R. Wiley. Production and perception of communicatory signals in a noisy environment. *Biology Letters*, 5(2):183, 2009. ISSN 1744-9561.

[106] D.I. MacKenzie, J.D. Nichols, G.B. Lachman, S. Droege, J. Andrew Royle, and C.A. Langtimm. Estimating site occupancy rates when detection probabilities are less than one. *Ecology*, 83(8):2248–2255, 2002. ISSN 0012-9658.

[107] Anne E Magurran. Measuring biological diversity. 2004.

[108] M. Mandel and D.P.W. Ellis. Multiple-instance learning for music information retrieval. In *Proc. Int. Symposium on Music Information Retrieval*, 2008.

[109] O. Maron and T. Lozano-Pérez. A framework for multiple-instance learning. *Advances in Neural Information Processing Systems*, pages 570–576, 1998.

[110] Oded Maron. *Learning from ambiguity*. PhD thesis, Massachusetts Institute of Technology, 1998.

[111] AL McIlraith and HC Card. Birdsong recognition using backpropagation and multivariate statistics. *IEEE Transactions on Signal Processing*, 45(11):2740–2748, 1997.

[112] D.K. Mellinger and J. W. Bradbury. Acoustic measurement of marine mammal sounds in noisy environments. In *Proc. Int. Conf. Underwater Acoust. Measurements: Technologies and Results*, pages 273–280, 2007.

[113] T. Minka. Estimating a Dirichlet distribution. *Unpublished paper available at http://research. microsoft. com/ minka*, 2003.

[114] L. Neal, F. Briggs, R. Raich, and X.Z. Fern. Time-frequency segmentation of bird song in noisy acoustic environments. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 2012–2015. IEEE, 2011.

[115] N. Nguyen. A new svm approach to multi-instance multi-label learning. In *Tenth IEEE International Conference on Data Mining. ICDM'10*, pages 384–392, 2010.

[116] C. Parmesan and G. Yohe. A globally coherent fingerprint of climate change impacts across natural systems. *Nature*, 421(6918):37–42, 2003. ISSN 0028-0836.

[117] Scott F. Pearson. Hermit warbler (setophaga occidentalis), the birds of north america online. 1997. doi: 10.2173/bna.303. Retrieved from Birds of North America Online: `http://bna.birds.cornell.edu/bna/species/303` (last viewed 1/26/12).

[118] L.R. Rabiner and M.R. Sambur. An algorithm for determining the endpoints of isolated utterances. *The Bell System Technical Journal*, 54(2):297–315, 1975.

[119] A. Rahimi and B. Recht. Random features for large-scale kernel machines. *jAdvances in Neural Information Processing Systems*, 20:1177–1184, 2007.

[120] L. Ranjard and H.A. Ross. Unsupervised bird song syllable classification using evolving neural networks. *The Journal of the Acoustical Society of America*, 123:4358, 2008.

[121] Soumya Ray and Mark Craven. Supervised versus multiple instance learning: An empirical comparison. In *Proceedings of the 22nd international conference on Machine learning*, pages 697–704. ACM, 2005.

[122] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine learning*, 85(3):333–359, 2011.

[123] C.S. Robbins, S. Droege, and J.R. Sauer. Monitoring bird populations with breeding bird survey and atlas data. In *Annales Zoologici Fennici*, volume 26, pages 297–304, 1989.

[124] T. Scott Brandes. Automated sound recording and analysis techniques for bird surveys and conservation. *Bird Conservation Int.*, 18(S1):163–173, 2008. ISSN 0959-2709.

[125] Ç.H. Şekercioğlu, G.C. Daily, and P.R. Ehrlich. Ecosystem consequences of bird declines. *Proc. Nat. Acad. Sciences of the United States of Am.*, 101(52):18042, 2004.

[126] A. Selin, J. Turunen, and J.T. Tanttu. Wavelets in recognition of bird sounds. *EURASIP Journal on Advances in Signal Processing*, 2007:1–9, 2007.

[127] S.A. Selouani, M. Kardouchi, E. Hervet, and D. Roy. Automatic birdsong recognition based on autoregressive time-delay neural networks. In *Computational Intelligence Methods and Applications, 2005 ICSC Congress on*, pages 6–pp. IEEE, 2005.

[128] Claus Seyerlehner, Gerhard Widmer, and Peter Knees. Frame Level Audio Similarity - A Codebook Approach. In *Proceedings of the 11th International Conference on Digital Audio Effects (DAFx-08)*, Espoo, Finland, September 1–4 2008.

[129] S. Shalev-Shwartz and Y. Singer. Logarithmic regret algorithms for strongly convex repeated games. *The Hebrew University, Technical Report*, 2007.

[130] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th international conference on Machine learning*, pages 807–814. ACM, 2007.

[131] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

[132] C. Shen, J. Jiao, B. Wang, and Y. Yang. Multi-Instance Multi-Label Learning For Automatic Tag Recommendation. In *Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2009)*, 2009.

[133] H. Slabbekoorn and T.B. Smith. Bird song, ecology and speciation. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 357(1420):493, 2002. ISSN 0962-8436.

[134] P. Somervuo and A. Härmä. Bird song recognition based on syllable pair histograms. In *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*, volume 5, pages 825–828, 2004.

[135] Panu Somervuo and Aki Härmä. Analyzing bird song syllables on the self-organizing map. In *Proceedings of the Workshop on Self-Organizing Maps (WSOM'03)*, Kitakyushu, Japan, September 2003.

[136] Panu Somervuo, Aki Härmä, and Seppo Fagerlund. Parametric representations of bird sounds for automatic species recognition. In *IEEE Transactions on Audio, Speed, and Language Processing*, volume 14, 2006.

[137] B. Sriperumbudur and G. Lanckriet. On the convergence of the concave-convex procedure. *Advances in neural information processing systems*, 22:1759–1767, 2009.

[138] D. Stowell and M.D. Plumbley. Framewise heterodyne chirp analysis of birdsong. 2012.

[139] K.A. Swiston and D.J. Mennill. Comparison of manual and automated methods for identifying target sounds in audio recordings of pileated, pale-billed, and putative ivory-billed woodpeckers. *Journal of Field Ornithology*, 80(1):42–50, 2009.

[140] L.N. Tan, K. Kaewtip, M.L. Cody, C.E. Taylor, and A. Alwan. Evaluation of a sparse representation-based classifier for bird phrase classification under limited data conditions. In *13th Annual Conference of the International Speech Communication Association*, 2012.

[141] S. Tong and D. Koller. Support Vector Machine Active Learning with Applications to Text Classification. *The Journal of Machine Learning Research*, 2:45–66, 2001.

[142] V.M. Trifa, A.N.G. Kirschel, C.E. Taylor, and E.E. Vallejo. Automated species recognition of antbirds in a Mexican rainforest using hidden Markov models. *J. Acoust. Soc. Am.*, 123:2424, 2008.

[143] H. Tyagi, R.M. Hegde, H.A. Murthy, and A. Prabhakar. Automatic identification of bird calls using spectral ensemble average voice prints. In *Proceedings of the 13th European Signal Processing Conference*, 2006.

[144] Vivek Tyagi and Christian Wellekens. On desensitizing the mel-cepstrum to spurious spectral components for robust speech recognition. In *Proc. ICASSP*, volume 5, pages 529–532, 2005.

[145] G. Vaca-Castaño and D. Rodriguez. Using syllabic mel cepstrum features and k-nearest neighbors to identify anurans and birds species. In *Signal Processing Systems (SIPS), 2010 IEEE Workshop on*, pages 466–471. IEEE, 2010.

[146] G. Vaca-Castano, D. Rodriguez, J. Castillo, K. Lu, A. Rios, and F. Bird. A framework for bioacoustical species classification in a versatile service-oriented wireless mesh network. In *Proc. 2010 European Signal Processing Conference (EUSIPCO-2010)*, 2010.

[147] A. Vezhnevets, J.M. Buhmann, and ETH Zurich. Towards Weakly Supervised Semantic Segmentation by Means of Multiple Instance and Multitask Learning. In *Conference on Computer Vision and Pattern Recognition*, 2010.

[148] S. Vijayanarasimhan and K. Grauman. What's it going to cost you?: Predicting effort vs. informativeness for multi-label image annotations. *IEEE Conf. Computer Vision and Pattern Recognition*, pages 2262–2269, 2009.

[149] E. Vilches, I.A. Escobar, E.E. Vallejo, and C.E. Taylor. Data mining applied to acoustic bird species recognition. *Pattern Recognition*, 3:400–403, 2006. ISSN 1051-4651.

[150] J. Volkmann, S. S. Stevens, and E. B. Newman. A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, 8(3): 208–208, 1937. doi: 10.1121/1.1901999. URL `http://link.aip.org/link/?JAS/8/208/2`.

[151] W. Wang and Z.H. Zhou. Learnability of multi-instance multi-label learning. *Chinese Science Bulletin*, pages 1–4, 2012.

[152] R. Wielgat, T.P. Zielinski, T. Potempa, A. Lisowska-Lis, and D. Król. Hfcc based recognition of bird species. In *Signal Processing Algorithms, Architectures, Arrangements and Applications, 2007*, pages 129–134. IEEE, 2007.

[153] J. Wimmer, M. Towsey, B. Planitz, I. Williamson, and P. Roe. Analysing environmental acoustic data through collaboration and automation. *Future Generation Computer Systems*, 2012.

[154] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *Tenth IEEE International Conference on Computer Vision, 2005. ICCV 2005*, volume 2, 2005.

[155] I.H. Witten and E. Frank. Data mining: practical machine learning tools and techniques with Java implementations. *ACM SIGMOD Record*, 31(1):76–77, 2002.

[156] X. Xu and E. Frank. Logistic regression and boosting for labeled bags of instances. *Advances in Knowledge Discovery and Data Mining*, pages 272–281, 2004.

[157] X.S. Xu, X. Xue, and Z.H. Zhou. Ensemble multi-instance multi-label learning approach for video annotation task. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 1153–1156. ACM, 2011.

[158] Z. Xu, K. Yu, V. Tresp, X. Xu, and J. Wang. Representative Sampling for Text Classification Using Support Vector Machines. *Lecture Notes in Computer Science*, pages 393–407, 2003.

[159] O. Yakhnenko. *Learning from Text and Images: Generative and Discriminative Models for Partially Labeled Data*. PhD thesis, Iowa State University, 2009.

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

[161] S.H. Yang, J. Bian, and H. Zha. Hybrid Generative/Discriminative Learning for Automatic Image Annotation. In *Conference on Uncertainty in Artificial Intelligence*, 2010.

[162] Sj Young. The hidden markov model toolkit. *Entropic Cambridge Research Laboratory, Ltd*, 2:2–44, 1995.

[163] K. Yu, S. Yu, and V. Tresp. Dirichlet enhanced latent semantic analysis. In *Conference in Artificial Intelligence and Statistics*, 2005.

[164] A.L. Yuille and A. Rangarajan. The concave-convex procedure (cccp). *Advances in Neural Information Processing Systems*, 2:1033–1040, 2002.

[165] Z.J. Zha, X.S. Hua, T. Mei, J. Wang, G.J. Qi, and Z. Wang. Joint multi-label multi-instance learning for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008*, pages 1–8, 2008.

[166] M.L. Zhang. A k-nearest neighbor based multi-instance multi-label learning algorithm. *22nd IEEE Int. Conf. Tools with Artificial Intelligence*, pages 207–212, 2010.

[167] M.L. Zhang and Z.J. Wang. MIMLRBF: RBF neural networks for multi-instance multi-label learning. *Neurocomputing*, 72(16-18):3951–3956, 2009.

[168] M.L. Zhang and Z.H. Zhou. M3MIML: A maximum margin method for multi-instance multi-label learning. In *Eighth IEEE International Conference on Data Mining. ICDM'08*, pages 688–697, 2008.

[169] Q. Zhang and S.A. Goldman. EM-DD: An improved multiple-instance learning technique. *Advances in Neural Information Processing Systems*, 2:1073–1080, 2002.

[170] Z.H. Zhou. Multi-instance learning: A survey. *AI Lab, Department of Computer Science and Technology, Nanjing University, Tech. Rep*, 2004.

[171] Z.H. Zhou and M.L. Zhang. Multi-instance multi-label learning with application to scene classification. *Advances in Neural Information Processing Systems*, 19:1609, 2007.

[172] Z.H. Zhou, M.L. Zhang, S.J. Huang, and Y.F. Li. Multi-instance multi-label learning. *Artificial Intelligence*, 176(1):2291–2320, 2012.

[173] Zhi-Hua Zhou, Yu-Yin Sun, and Yu-Feng Li. Multi-instance learning by treating instances as non-iid samples. In *International Conference on Machine Learning*, pages 1249–1256, 2009.

APPENDICES

Appendix A: Instance Annotation for Multi-Instance Multi-Label Learning – Proof of Bound on Sub-Gradient

We follow the approach of Shalev-Shwartz and Singer [129] to establish the convergence for sub-gradient descent. Because the number of iterations to reach an ϵ-suboptimal solution is $O(\frac{L}{\lambda\epsilon})$, where L is a bound on the sub-gradient $||\mathbf{V}||^2 \leqslant L$, we proceed with a derivation of L. The bound proved in this appendix applies to the sub-gradients (5.27) and (5.29). 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:

$$||\mathbf{v}_q|| \leqslant \lambda||\mathbf{w}_q|| + \sum_{ijk} \beta_i I[\cdot]\Big(||\hat{\mathbf{x}}||I[q=k] + ||\hat{\mathbf{x}}||I[q=j]\Big) \tag{A.1}$$

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

$$||\mathbf{v}_q|| \leqslant \lambda||\mathbf{w}_q|| + R\sum_{ijk} \beta_i \Big(I[q=k] + I[q=j]\Big) \tag{A.2}$$

$$||\mathbf{v}_q|| \leqslant \lambda||\mathbf{w}_q|| + R\Big(\sum_{ijk} \beta_i I[q=k] + \sum_{ijk} \beta_i I[q=j]\Big) \tag{A.3}$$

The first sum on the RHS of (A.3) can be computed as

$$\begin{aligned}
\sum_{ijk} \beta_i I[q=k] &= \sum_{i=1}^{n} \beta_i \sum_{j \in Y_i} \sum_{k \notin Y_i} I[q=k] && \text{(A.4)}\\
&= \sum_{i=1}^{n} \beta_i \sum_{j=1}^{c} \sum_{k=1}^{c} I[q=k] I[j \in Y_i] I[k \notin Y_i] && \text{(A.5)}\\
&= \sum_{i=1}^{n} \beta_i I[q \notin Y_i] \underbrace{\sum_{j=1}^{c} I[j \in Y_i]}_{|Y_i|} = \sum_{i=1}^{n} \beta_i I[q \notin Y_i]|Y_i| && \text{(A.6)}
\end{aligned}$$

Similarly, the second sum on the RHS of (A.3) can be simplified as

$$\sum_{ijk} \beta_i I[q = j] = \sum_{i=1}^{n} \beta_i I[q \in Y_i] |\bar{Y}_i| \tag{A.7}$$

Substituting these terms back into (A.3),

$$||\mathbf{v}_q|| \leqslant \quad \lambda ||\mathbf{w}_q|| + R\Big(\sum_{i=1}^{n} \beta_i I[q \notin Y_i] |Y_i| + \sum_{i=1}^{n} \beta_i I[q \in Y_i] |\bar{Y}_i| \Big) \tag{A.8}$$

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

$$a_q = \lambda ||\mathbf{w}_q|| \tag{A.9}$$

$$b_q = R \sum_{i=1}^{n} \beta_i \Big(I[q \notin Y_i] |Y_i| + I[q \in Y_i] |\bar{Y}_i| \Big) \tag{A.10}$$

Lemma A1.1

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

Proof.

$$||\mathbf{v}_q||^2 \leqslant (a_q + b_q)^2 = a_q^2 + b_q^2 + 2 a_q b_q \tag{A.11}$$

$$||\mathbf{V}||^2 = \sum_{q=1}^{c} ||\mathbf{v}_q||^2 \leqslant \sum_{q=1}^{c} a_q^2 + \sum_{q=1}^{c} b_q^2 + \sum_{q=1}^{c} 2 a_q b_q \tag{A.12}$$

$$\leqslant \sum_{q=1}^{c} a_q^2 + \sum_{q=1}^{c} b_q^2 + 2 \sqrt{\Big(\sum_{q=1}^{c} a_q^2 \Big) \Big(\sum_{q=1}^{c} b_q^2 \Big)} \text{ by Cauchy-Schwarz} \tag{A.13}$$

$$\leqslant \left(\sqrt{\sum_{q=1}^{c} a_q^2} + \sqrt{\sum_{q=1}^{c} b_q^2} \right)^2 \tag{A.14}$$

□

Lemma A1.2

$$\boxed{\sum_{q=1}^{c} a_q^2 \leqslant 2\lambda}$$

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

Lemma A1.3

$$\boxed{\sum_{q=1}^{c} b_q^2 \leqslant 4R^2}$$

Proof.

$$\sum_{q=1}^{c} b_q^2 = R^2 \sum_{q=1}^{c} \Big(\sum_{i=1}^{n} \beta_i (I[q \notin Y_i]|Y_i| + I[q \in Y_i]|\bar{Y}_i|) \Big)^2 \tag{A.15}$$

$$= R^2 \sum_{q=1}^{c} \sum_{i=1}^{n} \sum_{i'=1}^{n} \beta_i \beta_{i'} \Big(I[q \notin Y_i]|Y_i| + I[q \in Y_i]|\bar{Y}_i| \Big) \Big(I[q \notin Y_{i'}]|Y_{i'}| + I[q \in Y_{i'}]|\bar{Y}_{i'}| \Big) \tag{A.16}$$

Note the above 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.,

$$R^2 \sum_{q=1}^{c} \sum_{i=1}^{n} \sum_{i'=1}^{n} \beta_i \beta_{i'} I[q \notin Y_i] I[q \notin Y_{i'}] |Y_i||Y_{i'}| \tag{A.17}$$

$$= R^2 \sum_{i=1}^{n} \sum_{i'=1}^{n} \beta_i \beta_{i'} |Y_i||Y_{i'}| \sum_{q=1}^{c} I[q \notin Y_i] I[q \notin Y_{i'}] \tag{A.18}$$

$$\leqslant R^2 \sum_{i=1}^{n} \sum_{i'=1}^{n} \beta_i \beta_{i'} |Y_i||Y_{i'}| \sqrt{\Big(\sum_{q=1}^{c} I[q \notin Y_i]^2 \Big) \Big(\sum_{q=1}^{c} I[q \notin Y_{i'}]^2 \Big)} \text{ by Cauchy-Schwarz} \tag{A.19}$$

$$= R^2 \sum_{i=1}^{n} \sum_{i'=1}^{n} \beta_i \beta_{i'} |Y_i||Y_{i'}| \sqrt{|\bar{Y}_i||\bar{Y}_{i'}|} \tag{A.20}$$

$$= R^2 \sum_{i=1}^{n} \sum_{i'=1}^{n} \frac{|Y_i||Y_{i'}| \sqrt{|\bar{Y}_i||\bar{Y}_{i'}|}}{n^2 |Y_i||\bar{Y}_i||Y_{i'}||\bar{Y}_{i'}|} = \frac{R^2}{n^2} \sum_{i=1}^{n} \sum_{i'=1}^{n} \frac{1}{\sqrt{|\bar{Y}_i||\bar{Y}_{i'}|}} = \frac{R^2}{n^2} \Big(\sum_{i=1}^{n} \frac{1}{\sqrt{|\bar{Y}_i|}} \Big) \Big(\sum_{i=1}^{n} \frac{1}{\sqrt{|\bar{Y}_i|}} \Big) \tag{A.21}$$

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

$$\sum_{q=1}^{c} b_q^2 \leqslant \frac{R^2}{n^2} \left(\Big(\sum_{i=1}^{n} \frac{1}{\sqrt{|\bar{Y}_i|}} \Big)^2 + \Big(\sum_{i=1}^{n} \frac{1}{\sqrt{|Y_i|}} \Big)^2 + 2 \Big(\sum_{i=1}^{n} \frac{1}{\sqrt{|\bar{Y}_i|}} \Big) \Big(\sum_{i=1}^{n} \frac{1}{\sqrt{|Y_i|}} \Big) \right) \tag{A.22}$$

$$= \frac{R^2}{n^2} \left(\sum_{i=1}^{n} \Big(\frac{1}{\sqrt{|Y_i|}} + \frac{1}{\sqrt{|\bar{Y}_i|}} \Big) \right)^2 \leqslant R^2 (1 + \frac{1}{\sqrt{c-1}})^2 \tag{A.23}$$

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

Combining Lemmas A1.1, A1.2 and A1.3, we get

$$||\mathbf{V}||^2 \leqslant \left(\sqrt{\sum_{q=1}^{c} a_q^2} + \sqrt{\sum_{q=1}^{c} b_q^2} \right)^2 \tag{A.24}$$

$$\leqslant \left(\sqrt{2\lambda} + 2R \right)^2 \tag{A.25}$$

therefore $L' = \left(\sqrt{2\lambda} + 2R \right)^2$ is suitable bound for the Pegasos analysis. This result applies to

either SIM-CCCP or SIM-Heuristic with rank loss.

Appendix B: Instance Annotation for Multi-Instance Multi-Label Learning – Implementation Details

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

Average Support Instance Initialization The rank loss objective is non-convex 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 sub-gradient 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 sub-gradient 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 [137]. 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 sub-gradient 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 sub-gradient 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. $h^{(t)}_{RL,cccp}(\mathbf{W}^{(t,\tau)}) \leqslant h^{(t)}_{RL,cccp}(\mathbf{W}^{(t)})$, then τ is a sufficient number of iterations to ensure DC objective is monotonically decreasing, because

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

In our implementation of CCCP, we start by running K iterations of sub-gradient 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 sub-gradient 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 [40]), 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 above 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 the above processing steps effective.

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

$$\alpha^j_{iq} = \frac{e^{\mathbf{w}_j \cdot \mathbf{x}_{iq}}}{\sum_{\mathbf{x}' \in X_i} e^{\mathbf{w}_j \cdot \mathbf{x}'}}$$

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 which avoids numerical issues is

$$\alpha^j_{iq} = \frac{e^{\mathbf{w}_j \cdot \mathbf{x}_{iq} - b}}{\sum_{\mathbf{x}' \in X_i} e^{\mathbf{w}_j \cdot \mathbf{x}' - b}} \text{ where } b = \max_{\mathbf{x} \in X_i} \mathbf{w}_j \cdot \mathbf{x} \tag{B.2}$$