Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence

$\mathcal{HC}$-Search for Multi-Label Prediction: An Empirical Study

Janardhan Rao Doppa, Jun Yu, Chao Ma, Alan Fern and **Prasad Tadepalli**
{doppa, yuju, machao, afern, tadepall}@eecs.oregonstate.edu
School of EECS, Oregon State University, Corvallis, OR, USA, 97331

Abstract

Multi-label learning concerns learning multiple, overlapping, and correlated classes. In this paper, we adapt a recent structured prediction framework called $\mathcal{HC}$-Search for multi-label prediction problems. One of the main advantages of this framework is that its training is sensitive to the loss function, unlike the other multi-label approaches that either assume a specific loss function or require a manual adaptation to each loss function. We empirically evaluate our instantiation of the $\mathcal{HC}$-Search framework along with many existing multi-label learning algorithms on a variety of benchmarks by employing diverse task loss functions. Our results demonstrate that the performance of existing algorithms tends to be very similar in most cases, and that the $\mathcal{HC}$-Search approach is comparable and often better than all the other algorithms across different loss functions.

1 Introduction

We consider the problem of multi-label prediction, where the learner needs to predict multiple labels for a given input example (Tsoumakas, Zhang, and Zhou 2012). Multi-label problems commonly arise in domains involving data such as text, images, audio, and bio-informatics where instances can fall into overlapping conceptual categories of interest. For example, in document classification an input document can belong to multiple topics and in image classification an input image can contain multiple scene properties and objects of interest.

An important aspect of most multi-label problems is that the individual output labels are not independent, but rather are correlated in various ways. One of the challenges in multi-label prediction is to exploit this label correlation in order to improve accuracy compared to predicting labels independently. Unfortunately, existing approaches for multi-label prediction that consider label correlation suffer from the intractable problem of making optimal predictions (inference) (Dembczynski, Cheng, and Hüllermeier 2010; Ghamrawi and McCallum 2005; Zhang and Zhang 2010; Guo and Gu 2011; Petterson and Caetano 2011). Another challenge is to automatically adapt the learning approach to the task loss function that is most appropriate for the real-world application at hand. However, most approaches are designed to minimize a single multi-label loss function (Elisseeff and Weston 2001; Read et al. 2011; Fürnkranz et al. 2008). There are existing frameworks for multi-label prediction that can handle varying loss functions, but unfortunately they are non-trivial to adapt for a new task loss based on the needs of the application. For example, Probabilistic Classifier Chains (PCC) require a Bayes optimal inference rule (Dembczynski, Cheng, and Hüllermeier 2010) and Structured Support Vector Machines (SSVMs) require a loss-augmented inference routine for the given task loss (Tsochantaridis et al. 2005).

In this paper, we treat multi-label learning as a special case of structured-output prediction (SP), where each input x is mapped to a binary vector y (i.e., a structured output) that indicates the set of labels predicted for x. The main contribution of this paper is to investigate a simple framework for multi-label prediction called Multi-Label Search (MLS) that makes joint predictions without suffering from intractability of the inference problem, and can be easily adapted to optimize arbitrary loss functions[1]. The framework is based on instantiating a recent approach to structured prediction called $\mathcal{HC}$-Search (Doppa, Fern, and Tadepalli 2013) to multi-label learning.

The MLS approach first defines a generic combinatorial search space over all possible multi-label outputs. Next, a search procedure (e.g. breadth-first or greedy search) is specified, which traverses the output space with the goal of uncovering high-quality outputs for a given input x. Importantly, for this search to be effective, it will often be necessary to guide it using a learned heuristic. Finally a learned cost function is used to score the set of outputs uncovered by the search procedure, and the least-cost one is returned. The effectiveness of the MLS approach depends on: 1) the ability of the search to uncover good outputs for given inputs, which for difficult problems will depend on the quality of the search heuristic $\mathcal{H}$, and 2) the ability of the cost function $\mathcal{C}$ to select the best of those outputs. We employ existing learning approaches proposed within the $\mathcal{HC}$-Search framework for learning effective heuristics and cost functions for

[1] In a concurrent work to ours, the Condensed Filter Tree (CFT) algorithm was proposed for training loss-sensitive multi-label classifiers (Li and Lin 2014).

these purposes as they are shown to be very effective in practice.

Our second contribution is to conduct a broad evaluation of several existing multi-label learning algorithms along with our MLS approach on a variety of benchmarks by employing diverse task loss functions. Our results demonstrate that the performance of existing algorithms tends to be very similar in most cases, and that our MLS approach is comparable and often better than all the other algorithms across different loss functions. Our results also identify particular ways where our approach can be improved, pointing to future work.

2 Related Work

Typical approaches to multi-label learning decompose the problem into a series of independent binary classification problems, and employ a thresholding or ranking scheme to make predictions (Elisseeff and Weston 2001; Read et al. 2011; Fürnkranz et al. 2008; Tsoumakas, Katakis, and Vlahavas 2010). The Binary Relevance (BR) method ignores correlations between output labels and learns one independent classifier for every label (Tsoumakas, Katakis, and Vlahavas 2010). The Classifier Chain (CC) (Read et al. 2011) approach learns one classifier for every label based on input x and the assignments to previous labels in a fixed ordering over labels. CC leverages the interdependencies between output labels to some extent, but it suffers from two major problems: 1) It is hard to determine a good ordering of the labels in the chain, 2) Errors can propagate from earlier predictions to later ones (Ross and Bagnell 2010; Hal Daumé III, Langford, and Marcu 2009; Ross, Gordon, and Bagnell 2011). To address some of these issues, researchers have employed ensembles of chains (ECC), beam search (Kumar et al. 2013) and monte carlo search (MCC) (Read, Martino, and Luengo 2013) techniques to find a good ordering and for predicting the labels. All these label decomposition approaches try to optimize the Hamming loss.

Output coding methods try to exploit the correlations between output labels by coding them using a different set of latent classes. There are several output coding techniques for multi-label learning, including coding based on compressed sensing (Hsu et al. 2009), Principal Component Analysis (PCA) (Tai and Lin 2012), and Canonical Correlation Analysis (CCA) (Zhang and Schneider 2011). These methods either try to find a discriminative set of codes ignoring predictability, or vice versa. The recent max-margin output coding method (Zhang and Schneider 2012) tries to overcome some of the drawbacks of previous coding approaches by trying to find a set of codes that are both discriminative and predictable via a max-margin formulation. However, their formulation poses the problem of finding these codes as an intractable optimization problem for which they propose an approximate solution. These output coding approaches optimize a fixed, but unknown loss function.

Graphical modeling approaches including Conditional Random Fields (CML) (Ghamrawi and McCallum 2005), Bayesian Networks (LEAD) (Zhang and Zhang 2010), and Conditional Dependency Networks (CDN) (Guo and Gu 2011) try to capture label dependencies, but unfortunately suffer from the intractability of the exact inference problem due to the high tree-width graphical structure. It is possible to employ approximate inference methods (e.g., Loopy Belief Propagation and MCMC), with the associated risk of converging to local optima. These methods try to optimize the structural log loss or some variant of it.

Probablistic Classifier Chains (PCC) (Dembczynski, Cheng, and Hüllermeier 2010) estimate the conditional probablity of every possible label set for an input instance, and employ a Bayes optimal inference rule to optimize the given task loss function. However, the PCC framework suffers from two problems: 1) It is hard to accurately estimate the conditional probabilities, and 2) It is non-trivial to come up with the inference rule for a new loss function. Exact inference rules exist for limited loss functions: Hamming loss, Rank loss, and F1 loss (Dembczynski, Cheng, and Hüllermeier 2010; Dembczynski et al. 2011; Dembczynski, Kotlowski, and Hüllermeier 2012; Dembczynski et al. 2013). Approximate inference methods can be employed to optimize the Exact-Match loss (Dembczynski, Waegeman, and Hüllermeier 2012; Kumar et al. 2013; Read, Martino, and Luengo 2013).

The Structured Support Vector Machines (SSVMs) (Tsochantaridis et al. 2005) framework allows varying loss functions, but requires a loss-augmented inference routine for the given task loss function, which is non-trivial if the loss function is non-decomposable. Existing multi-label prediction approaches based on this framework either resort to approximate inference or some form of convex relaxation for non-decomposable losses (Hariharan et al. 2010; Petterson and Caetano 2010; 2011).

Label powerset (LP) methods reduce the multi-label learning problem to a multi-class classification problem, and optimize the Exact-Match loss. These approaches are very inefficient for training and testing. RAndom K-labELsets (RAKEL) is a representative approach of LP methods (Tsoumakas and Vlahavas 2007). Some recent work has proposed a variant of RAKEL to optimize weighted Hamming loss (Lo et al. 2011). ML-kNN (Zhang and Zhou 2007) is an extension of the traditional k-Nearest Neighbor classification algorithm for multi-label prediction. It is very expensive to make predictions with ML-kNN for large-scale training data.

3 Multi-Label Search Framework

In this section, we first describe the formal problem setup. Next, we give an overview of the $\mathcal{HC}$-Search framework followed by our instantiation for multi-label prediction problems and then describe the learning algorithms.

3.1 Problem Setup

A multi-label prediction problem specifies a space of inputs $\mathcal{X}$, where each input $x \in \mathcal{X}$ can be represented by a d dimensional feature vector; a space of outputs $\mathcal{Y}$, where each output $y = (y_1, y_2, \cdots, y_T) \in \mathcal{Y}$ is a binary vector of length T; and a non-negative *loss function* $L : \mathcal{X} \times \mathcal{Y} \times \mathcal{Y} \mapsto \Re^+$ such that $L(x, y', y^*)$ is the loss associated with labeling a particular input x by output y' when the true output is y^*. We are

provided with a training set of input-output pairs $\{(x, y^*)\}$ drawn from an unknown target distribution $\mathcal{D}$. The goal is to return a function/predictor from inputs to outputs whose predicted outputs have low expected loss with respect to the distribution $\mathcal{D}$.

3.2 Overview of the $\mathcal{HC}$-Search Framework

The $\mathcal{HC}$-Search framework for structured prediction is based on search in the output space $\mathcal{Y}$ and is parameterized by the following elements: 1) a search space $\mathcal{S}_o$, where each state in the search space consists of an input-output pair (x, y) where y represents the potential output for the input x, 2) a time-bounded search strategy $\mathcal{A}$ (e.g., depth-limited greedy search), 3) a learned heuristic function $\mathcal{H} : \mathcal{X} \times \mathcal{Y} \mapsto \Re$ in cases where the search strategy requires one, and 4) a learned cost function $\mathcal{C} : \mathcal{X} \times \mathcal{Y} \mapsto \Re$.

Given all of these elements and an input x, a prediction is made by first running the search procedure $\mathcal{A}$ (guided by $\mathcal{H}$ when appropriate), for a specified time bound τ. During the search a set of states is traversed, where each state represents a potential output for x. The cost function is employed to score each such output and the least-cost output is returned as the predicted label for x. The effectiveness of this approach depends on the quality of the search space (i.e., expected depth at which the target outputs can be located), the ability of the search procedure and heuristic function to guide the search to uncover high-quality outputs, and the quality of the cost function in terms of correctly scoring those outputs.

3.3 Multi-Label Search (MLS)

To instantiate the $\mathcal{HC}$-Search framework for multi-label prediction, we need to specify effective search spaces and search strategies that are appropriate for different multi-label prediction problems.

Search Space. The states of our multi-label search space correspond to input-output pairs (x, y), representing the possibility of predicting y as the multi-label output for x. In general, such a search space is defined in terms of two functions: 1) An *initial state function* I that takes an input x and returns an initial search state (x, y), and 2) A *successor function* S that takes a state as input and returns a set of child states. Given an input x, the search always begins at state $I(x)$ and then traverses the space by following paths allowed by the successor function.

In this paper, we employ a simple search space for multi-label problems, which we call the *Flipbit-null* space. In particular, the initial state function I is defined as $I(x) = (x, \text{null})$, where null is the zero vector indicating that no labels are present. The successor function $S((x, y))$ returns all states of the form (x, y'), where y' differs from y in exactly one label position, i.e. the hamming distance between y and y' is 1. Thus, individual search steps in this space can be viewed as picking a particular output label and flipping its value. This space is effectively the search space underlying Gibbs sampling. Clearly the search space is complete, since for any input x it is possible to reach any possible output starting from the initial state.

Search Space Quality. The quality of a search space can be understood in terms of the expected amount of search needed to uncover the correct output y^*, which often increases monotonically with the expected depth of the target in the search space. In particular, for a given input-output pair (x, y^*), the target depth $d(x, y^*)$ is defined as the minimum depth at which we can find a state corresponding to the target output y^*. Clearly according to this definition, the expected target depth of the Flipbit-null space is equal to the expected number of non-zero labels. That is, for the Flipbit-null space we have,

$$\begin{aligned} \mathbf{d} &= \mathbb{E}_{(x,y^*)\sim\mathcal{D}}\ d\,(x, y^*) \\ &= \mathbb{E}_{(x,y^*)\sim\mathcal{D}}\ |y^*|_1 \end{aligned} \tag{1}$$

Thus, the expected target depth of the Flipbit-null space is related to the average sparseness of the label vectors. We observe that for several standard benchmarks the outputs are very sparse[2] (80 perecent of the benchmarks have sparsity less than 4), which makes the above search space very effective. To the best of our knowledge, we are not aware of any multi-label approach that *explicitly* exploits the sparsity property of multi-label problems.

Other Search Space Choices. One possible way to decrease the expected target depth, if necessary, would be to define more sophisticated search spaces that are tuned for particular types of multi-label problems. As a simple example, if the number of zero entries in the outputs is small, then it would be more effective to define the initial state of the Flipbit space to be the vector of all ones. The expected target depth would then be the expected number of zero output labels. Another way to reduce the expected target depth would be to use an existing multi-label approach P (e.g., Binary Relevance) to produce the initial state. The resulting Flipbit space can then be viewed as biasing the search toward this solution (e.g., see (Lam et al. 2013)). In this case, the expected target depth of the search space would be equal to the expected Hamming error of P on the multi-label problem. Finally, even more sophisticated spaces such as the Limited Discrepancy Search space (Doppa, Fern, and Tadepalli 2012) defined in terms of a greedy classifier chain or a variant of its sparse version (Doppa, Fern, and Tadepalli 2014b) could be employed.

Search Strategies. Recall that in our MLS approach, the role of the search procedure is to uncover high-quality outputs. We can consider depth-bounded breadth-first search (BFS), but unfortunately BFS will not be practical for a large depth k and/or a large number of output labels T. Even when BFS is practical, it generates a large number of outputs (T^k) that will make the cost function learning problem harder. Therefore, in this paper, we consider depth-limited greedy search guided by a (learned) heuristic function $\mathcal{H}$ as our search strategy. Given an input x, greedy search traverses a path of user specified length k through the search space, at each point selecting the successor state that looks best according to the heuristic. In particular, if s_i is the state at search step i, greedy search selects $s_{i+1} = \arg\min_{s\in S(s_i)} \mathcal{H}(s)$, where $s_0 = I(x)$. The time

[2] http://mulan.sourceforge.net/datasets.html

complexity of generating this sequence is $O(k \cdot T)$, which makes it much more practical than BFS for larger values of k. The effectiveness of greedy search is determined by how well $\mathcal{H}$ guides the search toward generating state sequences that contain high quality outputs. It is possible to consider other heuristic search strategies, such as best-first search and beam-search. However, in our experience so far, greedy search has proven sufficient.

3.4 Learning Algorithms

We first describe the loss decomposition of the $\mathcal{HC}$-Search approach along with its staged learning. Next, we briefly talk about the heuristic and cost function learning algorithms in the context of greedy search. In this work, we focus on learning linear $\mathcal{H}$ and $\mathcal{C}$ of the form $\mathcal{H}(x, y) = w_{\mathcal{H}} \cdot \Phi_{\mathcal{H}}(\text{x,y})$ and $\mathcal{C}(x, y) = w_{\mathcal{C}} \cdot \Phi_{\mathcal{C}}(\text{x,y})$, where $\Phi_{\mathcal{H}}$ and $\Phi_{\mathcal{C}}$ are feature functions that compute the feature vectors for $\mathcal{H}$ and $\mathcal{C}$ respectively, and $w_{\mathcal{H}}$ and $w_{\mathcal{C}}$ stand for their weights that will be learned from the training data.

Loss Decomposition and Staged Learning. For any heuristic function $\mathcal{H}$ and cost function $\mathcal{C}$, the overall loss of the $\mathcal{HC}$-Search approach $\mathcal{E}(\mathcal{H}, \mathcal{C})$ can be decomposed into the loss due to $\mathcal{H}$ not being able to generate the target output (generation loss $\epsilon_{\mathcal{H}}$), and the additional loss due to $\mathcal{C}$ not being able to score the best outputs generated by $\mathcal{H}$ correctly (selection loss $\epsilon_{\mathcal{C}|\mathcal{H}}$) . The loss decomposition can be mathematically expressed as follows:

$$\mathcal{E}(\mathcal{H}, \mathcal{C}) = \underbrace{\mathbb{E}_{(x,y^*)\sim\mathcal{D}}\, L(x, y^*_{\mathcal{H}}, y^*)}_{\epsilon_{\mathcal{H}}} + \underbrace{\mathbb{E}_{(x,y^*)\sim\mathcal{D}}\, L(x, \hat{y}, y^*) - L(x, y^*_{\mathcal{H}}, y^*)}_{\epsilon_{\mathcal{C}|\mathcal{H}}}$$

where $y^*_{\mathcal{H}}$ is the best output that is generated by the search guided by $\mathcal{H}$ and $\hat{y}$ is the predicted output. The $\mathcal{HC}$-Search approach performs a staged-learning by first learning a heuristic to $\mathcal{H}$ to minimize the generation loss $\epsilon_{\mathcal{H}}$, and then the cost function $\mathcal{C}$ is learned by minimizing the selection loss $\epsilon_{\mathcal{C}|\mathcal{H}}$, given the learned $\mathcal{H}$.

Heuristic Learning. The heuristic function $\mathcal{H}$ is trained via imitation learning. For a given training time bound τ_{max} and task loss function L, we perform greedy search with the loss function used as an oracle heuristic on every training example (x, y^*) (ties are broken randomly) and generate training data for imitation (see Algorithm 1). The imitation example R_t at each search step t consists of one ranking example for every candidate state $s \in S(s_{t-1}) \setminus s_t$ such that $\mathcal{H}(x, y_t) < \mathcal{H}(x, y)$, where (x, y_t) and (x, y) correspond to the input-output pairs associated with states s_t and s respectively. The aggregate set of imitation examples collected over all the training data is then given to a rank learner (e.g., Perceptron or SVM-Rank) to learn the parameters of $\mathcal{H}$.

Cost Function Learning. The cost function $\mathcal{C}$ is trained via cross-validation to avoid over-fitting (see (Doppa, Fern, and Tadepalli 2014a) for full details). We divide the training data into k folds and learn k different heuristics, where each heuristic function $\mathcal{H}_i$ is learned using the data from all the folds excluding the i^{th} fold. We generate ranking examples for cost function learning using each heuristic function $\mathcal{H}_i$ on the data it was not trained on. Specifically, we perform greedy search guided by $\mathcal{H}_i$ to generate a set of outputs $\mathcal{Y}_{\mathcal{H}_i}(x)$, and generate ranking examples for any pair of outputs $(y_{best}, y) \in \mathcal{Y}_{best} \times \mathcal{Y}_{\mathcal{H}_i}(x) \setminus \mathcal{Y}_{best}$ such that $\mathcal{C}(x, y_{best}) < \mathcal{C}(x, y)$, where $\mathcal{Y}_{best}$ is the set of all best loss outputs. We give the aggregate set of ranking examples to a rank learner to learn the cost function $\mathcal{C}$.

Algorithm 1 Heuristic Function Learning for Greedy Search

Input: $\mathcal{D}$ = Training data, (I, S) = Search space, L = Loss function, τ_{max} = no. of training steps

```
Initialize the set of ranking examples R = ∅
for each training example (x, y*) ∈ D do
    s_0 ← I(x) // initial state
    for each search step t = 1 to τ_max do
        Generate example R_t to imitate this search step
        Aggregate training data: R = R ∪ R_t
        s_t ← arg min_{s∈S(s_{t-1})} L(s) // oracle search
    end for
end for
H = Rank-Learner(R)
return heuristic function H
```

4 Empirical Results

In this section, we evaluate our MLS approach along with several existing multi-label algorithms on a variety of benchmarks and evaluation measures.

4.1 Datasets

We employ nine benchmark[3] datasets for our evaluation. We selected these datasets based on the diversity of domains (text, images, audio and bio-informatics) and their popularity within the multi-label learning community. Table 2 presents the properties of different datasets. Ten percent of the training data were used to tune the hyper-parameters.

4.2 Experimental Setup

Evaluation Measures. We consider four diverse loss functions: Hamming loss, Exact-Match (0/1) loss, F1 loss, and Accuracy loss. F1 and Accuracy losses do not decompose over individual labels. They are defined as follows: F1 loss = 1 - $\frac{2\|\hat{y}\cap y^*\|_1}{\|\hat{y}\|_1+\|y^*\|_1}$; Accuracy loss = 1 - $\frac{\|\hat{y}\cap y^*\|_1}{\|\hat{y}\cup y^*\|_1}$, where $\hat{y}$ is the predicted output and y^* is the correct output. When both the predicted labels and ground-truth labels are zero vectors, then we consider the loss to be *zero* for both F1 and Accuracy unlike existing software packages including Mulan and Meka[4] that consider the loss to be *one* in this case.

MLS Approach. We employ the simple Flipbit-null space and greedy search as described in Section 3. We use unary and pair-wise potential features for our heuristic and cost function representation, i.e., these functions are represented as $w \cdot \Phi(x, y)$, where w is the parameter vector to be learned, and $\Phi(x, y)$ is a feature vector that contains indicator features for the activation levels of all label pairs and features that are the cross product of the label space and input features composing x. We estimate the expected target

[3] http://mulan.sourceforge.net/datasets.html

[4] http://meka.sourceforge.net/

ALGORITHMS	Scene	Emotions	Medical	Genbase	Yeast	Enron	LLog	Slashdot	Tmc2007
a. Hamming Accuracy Results									
BR	86.90	77.10	98.50	99.80	78.30	94.00	97.70	94.50	**94.70**
CC	88.70	76.80	98.60	99.90	78.50	95.10	98.40	94.50	94.60
ECC	89.00	78.40	98.30	99.90	78.50	94.30	98.40	94.70	**94.70**
M2CC	89.80	78.50	98.30	99.90	78.10	94.50	98.50	94.90	94.60
CLR	89.10	78.40	97.90	96.60	77.00	94.00	97.10	92.70	94.50
CDN	89.40	80.30	98.40	99.60	78.10	94.70	97.70	94.60	94.60
CCA	88.59	79.05	97.79	99.19	79.05	93.66	95.10	94.60	94.22
PIR	87.81	67.99	98.79	**99.93**	77.18	94.66	97.05	94.11	94.34
SML	86.32	78.64	**98.83**	99.04	78.64	94.80	**99.60**	95.28	94.01
RML	88.11	79.04	**98.83**	99.89	79.71	95.25	98.46	95.48	94.68
DecL	90.12	81.89	98.79	99.80	79.67	95.40	98.40	95.22	93.91
MLS	**90.41**	**82.75**	**98.83**	99.80	**80.72**	**95.60**	98.50	**95.63**	94.10
b. Instance-based F1 Results									
BR	52.60	60.20	63.90	98.70	63.20	53.90	36.00	46.20	71.80
CC	59.10	57.50	64.00	99.40	63.20	53.30	26.50	44.90	70.30
ECC	68.00	62.60	65.30	99.40	64.60	59.10	32.20	50.20	72.70
M2CC	68.20	63.20	65.40	99.40	64.90	59.10	32.30	50.30	72.80
CLR	62.20	**66.30**	66.20	70.70	63.80	56.50	22.70	46.60	70.80
CDN	63.20	61.40	68.90	97.80	64.00	58.50	36.60	53.10	71.30
CCA	66.43	63.27	49.60	98.60	61.64	53.83	25.80	48.00	69.53
PIR	74.45	60.92	80.17	99.41	**65.47**	61.14	38.95	57.55	**73.73**
SML	68.50	64.32	68.34	**99.62**	64.32	57.46	34.95	55.73	71.63
RML	74.17	64.83	**80.73**	98.80	63.18	57.79	35.97	51.30	71.34
DecL	73.76	65.29	78.02	97.89	63.46	61.19	37.52	54.67	69.08
MLS	**75.89**	66.17	78.19	98.12	63.78	**62.34**	**39.76**	**57.98**	69.17
c. Instance-based Accuracy Results									
BR	48.50	52.30	61.50	98.00	52.30	44.10	27.80	41.90	62.30
CC	55.90	49.70	61.00	99.10	51.80	43.00	25.30	42.00	61.70
ECC	63.40	54.80	62.20	99.10	53.70	47.00	29.40	45.70	63.50
M2CC	63.70	55.00	62.90	99.10	53.40	47.10	29.50	46.00	63.70
CLR	62.50	56.80	58.10	56.10	51.30	42.70	17.20	38.10	60.00
CDN	61.50	56.80	64.70	96.60	52.80	47.00	32.30	48.40	62.10
CCA	62.12	55.40	60.10	98.20	50.82	42.90	19.60	43.30	62.38
PIR	67.87	49.75	**76.33**	**99.16**	**53.92**	49.16	34.42	**52.87**	63.76
SML	63.65	52.38	64.03	98.42	52.38	48.08	33.49	43.92	**63.77**
RML	67.23	53.91	75.90	98.17	52.41	47.98	33.16	47.27	63.05
DecL	66.19	54.17	74.23	97.91	50.45	49.87	35.78	48.77	58.56
MLS	**69.12**	**57.89**	74.98	98.34	51.23	**51.21**	**36.67**	**52.85**	59.76
d. Exact-Match Results									
BR	45.90	24.80	46.20	95.50	15.60	10.90	21.90	31.50	32.20
CC	47.50	25.20	47.80	98.00	19.20	12.50	22.60	32.00	**34.00**
ECC	52.00	28.20	43.60	98.00	19.60	11.90	22.40	32.50	33.50
M2CC	**59.63**	**32.20**	43.90	98.00	**21.50**	13.50	**24.70**	33.20	**34.00**
CLR	58.30	28.70	33.60	11.10	5.80	2.80	2.70	13.90	25.10
CDN	57.60	**32.20**	52.20	94.00	17.00	12.60	22.40	34.10	32.40
CCA	**59.63**	30.20	22.48	97.99	20.39	**15.70**	15.07	32.00	31.04
PIR	50.08	19.80	**64.65**	**98.49**	14.29	13.64	23.63	**38.80**	30.73
SML	52.84	30.06	62.17	91.51	15.05	12.15	24.23	35.03	31.75
RML	47.32	25.74	62.94	91.45	13.63	12.43	24.48	35.09	30.44
DecL	59.00	31.89	63.76	96.81	15.12	12.11	20.81	37.89	29.98
MLS	58.10	31.18	63.46	96.75	14.30	12.71	19.12	38.13	28.29

Table 1: Performance of different multi-label prediction algorithms.

Dataset	Domain	#TR	#TS	#F	#L	$\mathbb{E}[d]$
Scene	image	1211	1196	294	6	1.07
Emotions	music	391	202	72	6	1.86
Medical	text	333	645	1449	45	1.24
Genbase	biology	463	199	1185	27	1.25
Yeast	biology	1500	917	103	14	4.23
Enron	text	1123	579	1001	53	3.37
LLog	text	876	584	1004	75	1.18
Slashdot	text	2269	1513	1079	22	1.18
Tmc2007	text	21519	7077	500	22	2.15

Table 2: Characteristics of the datasets: the number of training (#TR) and testing (#TS) examples; number of features (#F); number of labels (#L); and the expected target depth of our Flipbit-null space ($\mathbb{E}[d]$).

depth $\mathbb{E}[d]$ of the Flipbit-null space for each dataset and use $2 * \lceil \mathbb{E}[d] \rceil$ steps for training and testing the heuristic function noting that we didn't see any improvements with larger time-bounds. For cost function learning, we experimented with 3 folds and 5 folds, but larger folds didn't help much. We employ SVM-Rank (Joachims 2006) as our base rank learner for both heuristic and cost function learning. The C parameter was tuned using the validation set. The MLS approach cannot work for Exact-Match loss[5], so we present the Exact-Match results by training with Hamming loss. In all other cases, we train for the given task loss function. Our base rank learner did not scale to the Tmc2007 dataset, so we performed our training on a subset of 5000 training examples.

Baseline Methods. Our baselines include Binary Relevance `BR` (Tsoumakas, Katakis, and Vlahavas 2010); Classifier Chain with greedy inference `CC` (Read et al. 2011); Ensemble of Classifier Chains `ECC` (Read et al. 2011); Monte Carlo optimization of Classifier Chains `M2CC` (Read, Martino, and Luengo 2013); Calibrated Label Ranking `CLR` (Fürnkranz et al. 2008); Conditional Dependency Networks `CDN` (Guo and Gu 2011); Canonical Correlation Analysis `CCA` (Zhang and Schneider 2011); Plug-in-Rule approach `PIR` (Dembczynski et al. 2013); Submodular Multi-Label prediction `SML` (Petterson and Caetano 2011); Reverse Multi-Label prediction `RML` (Petterson and Caetano 2010); and Decomposed Learning `DecL` (Samdani and Roth 2012). The last method, DecL, is a variant of our MLS approach that employs a different cost function learning algorithm by trying to rank the correct output y^* higher than all the outputs with a hamming distance of at most k from y^*. We employed the Meka package to run BR, CC, ECC, M2CC, CLR, and CDN. We ran the code provided by the authors for CCA[6], PIR[7], SML and RML[8].

For the methods that require a base classifier, we employed logistic regression with L2 regularization. The regularization parameter was tuned via 5-fold cross validation. We employed the natural ordering of labels for CC and 20 random orderings for ECC. We used 10 iterations for learning the ordering, and 100 iterations for inference with M2CC. The parameters of CCA were tuned as described in the original paper. For PIR, we tuned the hyper-parameter λ via 5-fold cross-validation and report the results with their exact algorithm (EFP). The hyper-parameters for RML (λ), and SML (λ, C) were tuned based on the validation set. For DecL, we employed the largest value of k that was practical for training the cost function.

[5]We cannot differentiate between the outputs with loss *one*.

[6]http://www.cs.cmu.edu/∼yizhang1/files/AISTAT2011_Code.zip

[7]https://github.com/multi-label-classification/PCC

[8]http://users.cecs.anu.edu.au/∼jpetterson/

4.3 Results

Table 1 shows the accuracy results (higher is better) of different multi-label approaches with different evaluation measures. We can make several interesting observations from these results. First, the performance of several algorithms tend to be very similar in most cases. Second, our MLS approach performs comparably and often better than all other algorithms for all evaluation measures other than the Exact-Match accuracy, which MLS cannot optimize. The results of MLS for Tmc2007 are competetive with other methods even though MLS was trained only on one-fourth of the training data. Third, ECC performs better than CC as one would expect, and M2CC significantly improves over both CC and ECC showing the benefit of learning the ordering and performing a more elaborate search instead of greedy search. Fourth, PIR performs comparably or better than all other methods on F1 accuracy across all the datasets excluding Emotions. This behavior is expected because PIR is designed to optimize F1 loss.

Due to the lack of space, we do not provide the error decomposition results for our MLS approach, but we would like to mention that the generation error for most datasets is close to zero, which means that most of our error is coming from the cost function, i.e., even though the heuristic is able to generate good outputs, cost function is not able to score them properly. Therefore, it would be productive to consider more powerful rank learners[9] (e.g., Regression trees (Mohan, Chen, and Weinberger 2011)) for cost function learning to improve the results.

5 Summary and Future Work

We introduced the Multi-Label Search (MLS) approach by adapting the $\mathcal{HC}$-Search framework for multi-label prediction problems. MLS can automatically adapt its training for a given task loss function, and can jointly predict all the labels without suffering from the intractability of inference. We show that the MLS approach gives comparable or better results than existing multi-label approaches across several benchmarks and diverse loss functions. Future work should tackle the problem of designing an appropriate search space for the problem at hand, and the problem of learning in the context of more sophisticated search strategies when the expected depth of the search space is high and the greedy search is not effective.

Acknowledgements. This work was supported in part by NSF grants IIS 1219258, IIS 1018490 and in part by the DARPA and AFRL under contract No. FA8750-13-2-0033.

[9]http://sourceforge.net/p/lemur/wiki/RankLib/

References

Dembczynski, K.; Waegeman, W.; Cheng, W.; and Hüllermeier, E. 2011. An exact algorithm for F-measure maximization. In *NIPS*, 1404–1412.

Dembczynski, K.; Jachnik, A.; Kotlowski, W.; Waegeman, W.; and Hüllermeier, E. 2013. Optimizing the F-measure in multi-label classification: Plug-in rule approach versus structured loss minimization. In *ICML (3)*, 1130–1138.

Dembczynski, K.; Cheng, W.; and Hüllermeier, E. 2010. Bayes optimal multilabel classification via probabilistic classifier chains. In *ICML*, 279–286.

Dembczynski, K.; Kotlowski, W.; and Hüllermeier, E. 2012. Consistent multilabel ranking through univariate losses. In *ICML*.

Dembczynski, K.; Waegeman, W.; and Hüllermeier, E. 2012. An analysis of chaining in multi-label classification. In *ECAI*, 294–299.

Doppa, J. R.; Fern, A.; and Tadepalli, P. 2012. Output space search for structured prediction. In *ICML*.

Doppa, J. R.; Fern, A.; and Tadepalli, P. 2013. HC-Search: Learning heuristics and cost functions for structured prediction. In *AAAI*.

Doppa, J. R.; Fern, A.; and Tadepalli, P. 2014a. HC-Search: A learning framework for search-based structured prediction. *To appear in Journal of Artificial Intelligence Research (JAIR)*.

Doppa, J. R.; Fern, A.; and Tadepalli, P. 2014b. Structured prediction via output space search. *Journal of Machine Learning Research (JMLR)* 15.

Elisseeff, A., and Weston, J. 2001. A kernel method for multi-labelled classification. In *NIPS*, 681–687.

Fürnkranz, J.; Hüllermeier, E.; Mencía, E. L.; and Brinker, K. 2008. Multilabel classification via calibrated label ranking. *Machine Learning* 73(2):133–153.

Ghamrawi, N., and McCallum, A. 2005. Collective multi-label classification. In *CIKM*, 195–200.

Guo, Y., and Gu, S. 2011. Multi-label classification using conditional dependency networks. In *IJCAI*, 1300–1305.

Hal Daumé III; Langford, J.; and Marcu, D. 2009. Search-based structured prediction. *Machine Learning* 75(3):297–325.

Hariharan, B.; Zelnik-Manor, L.; Vishwanathan, S. V. N.; and Varma, M. 2010. Large scale max-margin multi-label classification with priors. In *ICML*, 423–430.

Hsu, D.; Kakade, S.; Langford, J.; and Zhang, T. 2009. Multi-label prediction via compressed sensing. In *NIPS*, 772–780.

Joachims, T. 2006. Training linear SVMs in linear time. In *KDD*, 217–226.

Kumar, A.; Vembu, S.; Menon, A. K.; and Elkan, C. 2013. Beam search algorithms for multilabel learning. *Machine Learning* 92(1):65–89.

Lam, M.; Doppa, J. R.; Hu, X.; Todorovic, S.; Dietterich, T.; Reft, A.; and Daly, M. 2013. Learning to Detect Basal Tubules of Nematocysts in SEM Images. In *ICCV Workshop on Computer Vision for Accelerated Biosciences*.

Li, C.-L., and Lin, H.-T. 2014. Condensed filter tree for cost-sensitive multi-label classification. In *ICML (To appear)*.

Lo, H.-Y.; Wang, J.-C.; Wang, H.-M.; and Lin, S.-D. 2011. Cost-sensitive multi-label learning for audio tag annotation and retrieval. *IEEE Transactions on Multimedia* 13(3):518–529.

Mohan, A.; Chen, Z.; and Weinberger, K. Q. 2011. Web-search ranking with initialized gradient boosted regression trees. *Journal of Machine Learning Research - Proceedings Track* 14:77–89.

Petterson, J., and Caetano, T. S. 2010. Reverse multi-label learning. In *NIPS*, 1912–1920.

Petterson, J., and Caetano, T. S. 2011. Submodular multi-label learning. In *NIPS*, 1512–1520.

Read, J.; Pfahringer, B.; Holmes, G.; and Frank, E. 2011. Classifier chains for multi-label classification. *Machine Learning* 85(3):333–359.

Read, J.; Martino, L.; and Luengo, D. 2013. Efficient Monte Carlo optimization for multi-label classifier chains. In *ICASSP*, 3457–3461.

Ross, S., and Bagnell, D. 2010. Efficient reductions for imitation learning. In *AISTATS*.

Ross, S.; Gordon, G.; and Bagnell, D. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*.

Samdani, R., and Roth, D. 2012. Efficient decomposed learning for structured prediction. In *ICML*.

Tai, F., and Lin, H.-T. 2012. Multilabel classification with principal label space transformation. *Neural Computation* 24(9):2508–2542.

Tsochantaridis, I.; Joachims, T.; Hofmann, T.; and Altun, Y. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research* 6:1453–1484.

Tsoumakas, G., and Vlahavas, I. P. 2007. Random k-labelsets: An ensemble method for multilabel classification. In *ECML*, 406–417.

Tsoumakas, G.; Katakis, I.; and Vlahavas, I. P. 2010. Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook*. 667–685.

Tsoumakas, G.; Zhang, M.-L.; and Zhou, Z.-H. 2012. Introduction to the special issue on learning from multi-label data. *Machine Learning* 88(1-2):1–4.

Zhang, Y., and Schneider, J. G. 2011. Multi-label output codes using canonical correlation analysis. *Journal of Machine Learning Research - Proceedings Track* 15:873–882.

Zhang, Y., and Schneider, J. G. 2012. Maximum margin output coding. In *ICML*, 1223–1230.

Zhang, M.-L., and Zhang, K. 2010. Multi-label learning by exploiting label dependency. In *KDD*, 999–1008.

Zhang, M.-L., and Zhou, Z.-H. 2007. ML-kNN: A lazy learning approach to multi-label learning. *Pattern Recognition* 40(7):2038–2048.