Mach Learn (2013) 92:65–89
DOI 10.1007/s10994-013-5371-6

Beam search algorithms for multilabel learning

Abhishek Kumar · Shankar Vembu · Aditya Krishna Menon · Charles Elkan

Received: 18 November 2012 / Accepted: 25 April 2013 / Published online: 22 May 2013

Abstract Multilabel learning is a machine learning task that is important for applications, but challenging. A recent method for multilabel learning called probabilistic classifier chains (PCCs) has several appealing properties. However, PCCs suffer from the computational issue that inference (i.e., predicting the label of an example) requires time exponential in the number of tags. Also, PCC accuracy is sensitive to the ordering of the tags while training. In this paper, we show how to use the classical technique of beam search to solve both these problems. Specifically, we show how to apply beam search to make inference tractable, and how to integrate beam search with training to determine a suitable tag ordering. Experimental results on a range of datasets show that the proposed improvements yield a state-of-the-art method for multilabel learning.

Keywords Multilabel classification · Probabilistic models · Beam search · Structured prediction

Editors: Tijl De Bie and Peter Flach.

Abhishek Kumar and Shankar Vembu contributed equally to this work. Shankar Vembu carried out parts of this research while working at UCSD.

A. Kumar · A.K. Menon · C. Elkan
Department of Computer Science and Engineering, University of California, San Diego, USA

A. Kumar
e-mail: abhishek@ucsd.edu

A.K. Menon
e-mail: akmenon@ucsd.edu

C. Elkan
e-mail: elkan@ucsd.edu

S. Vembu (✉)
Donnelly Centre for Cellular and Biomolecular Research, University of Toronto, Toronto, Canada
e-mail: shankar.vembu@utoronto.ca

1 Introduction

In the classical supervised learning task of binary classification, the goal is to learn a model that, given an input $x \in \mathcal{X}$, returns a single binary prediction $y \in \{0, 1\}$. This value y is considered to be a label of the example x, denoting whether it possesses some characteristic, or not. For example, x may represent an image by its pixel values, and y may denote whether or not the image contains a face. Multilabel learning is an extension of binary classification where the goal is to return multiple binary predictions, or equivalently a vector $y \in \{0, 1\}^K$. The label y now measures multiple characteristics of the example x, each of which we call a tag. For example, x may represent an image as before, and y could denote whether K specific people's faces appear in the image.

The recently proposed probabilistic classifier chain (Dembczyński et al. 2010; Read et al. 2011) (PCC) method is an attractive solution for multilabel classification for several reasons. First, it is based on a reduction of multilabel to binary classification, which allows us to leverage existing research on designing scalable and powerful binary classifiers. Second, it is a principled probabilistic model, and there is a theoretical understanding of how it may be used to produce Bayes optimal predictions for a variety of loss functions (Dembczyński et al. 2010). Third, it is computationally inexpensive to train unlike, for example, structured prediction methods (Finley and Joachims 2008), which involve inference during training. Fourth, it is trained on the original label space without any prior transformations (Breiman and Friedman 1997; Weston et al. 2002; Hsu et al. 2009; Bi and Kwok 2011), which is important in certain settings and applications.

Despite the above positive characteristics, the current formulation of PCCs suffers from at least a couple of drawbacks. First, on the computational side, they are only applicable to multilabel data sets with a few number of tags. This is because to use a PCC at test time for an example with K possible tags, we need to evaluate all 2^K possible labelings, and pick the highest scoring one. This becomes quickly infeasible as K increases. Second, on the performance side, their accuracy depends on a pre-specified ordering of the tags. Different orderings result in solutions of different accuracy, and so a natural question is whether one can determine the ordering that yields the best performance. As with the previous issue, the current understanding of PCCs requires either picking a random ordering, or trying all $K!$ possibilities.

In this paper, we propose to address these shortcomings with PCCs using beam search, a classical heuristic search algorithm. In particular, we propose to use beam search to perform inference with PCCs at test time, changing the runtime from $O(2^K)$ to $O(bK)$, where b is a tunable beam width. As we shall demonstrate, in practice a beam size $b \ll 2^K$ achieves good performance. We also present an algorithm that integrates the search for the best ordering of tags with the learning algorithm. To avoid the burden of training a classifier for each ordering, we use kernel target alignment (Cristianini et al. 2001) to score the viability of a given ordering. Finally, we propose a richer feature representation for learning individual tag models than what is used in existing PCC approaches (Dembczyński et al. 2010; Read et al. 2011). Experimental results on a range of multilabel data sets show that the new scheme is able to improve on PCC, and to extend its applicability to data sets with a large number of tags.

This paper is organized as follows. First, in Sect. 2, we review the problem of multilabel learning and the PCC model. In Sect. 3, we analyze PCC in detail, pointing out its relationship to existing models, and highlighting some of the challenges in using them. Next, in Sect. 4, we show how one may use beam search to speed up test time inference of the method. In Sect. 5, we show how beam search may be integrated during the learning phase

to determine the tag ordering. In Sect. 6, we discuss several issues related to the focus of this paper, and comment on possible extensions of our work. Finally, we present a range of experimental results in Sect. 7.

This paper extends our earlier conference publication (Kumar et al. 2012). We have included a new section analyzing PCCs in relation to other methods (Sect. 3) and a significantly expanded discussion (Sect. 6). We have extended our experimental results (Sect. 7) by (i) comparing our beam search method for inference with the approximate inference method of Dembczyński et al. (2012), (ii) comparing PCCs trained with our beam search method for learning the tag ordering with ensemble PCCs (Dembczyński et al. 2010), and (iii) providing a rigorous analysis of experimental results using statistical hypothesis tests.

2 Multilabel learning and probabilistic classifier chains

Let $\mathcal{X} \subseteq \mathbb{R}^d$ be the input space and $\mathcal{Y} = \{0, 1\}^K$ be the label output space defined over a fixed set of K tags. Given a set of m training samples $\mathcal{T} = \{(x^{(i)}, y^{(i)})\}_{i=1}^m$ where $(x^{(i)}, y^{(i)}) \in \mathcal{X} \times \mathcal{Y}$, the goal of a multilabel classification algorithm is to learn a mapping $f : \mathcal{X} \to \mathcal{Y}$. We will use the notation $y_k^{(i)}$ to denote the kth tag of the ith example.

The naïve solution to the multilabel learning problem is to decompose it into K independent binary classification problems, one for each tag y_k. This method is known as binary relevance. This method is optimal in theory for certain loss functions, such as the Hamming and the ranking loss (Dembczyński et al. 2010). However, in practical situations where training data is limited, and for loss functions that take the consistency of the entire tag sequence into account, it is intuitively necessary to exploit correlations between tags to make better predictions. This intuition has motivated numerous multilabel learning algorithms (see Tsoumakas and Katakis 2007; Tsoumakas et al. 2010; Sorower 2010 for surveys).

Recently, Read et al. (2011) proposed a simple decomposition method called the classifier chain (CC) that appears similar to binary relevance, but is able to exploit tag correlations. As with binary relevance, the idea is to train K separate models, one for each tag. The difference is that the model for tag k uses as input features not only the data point x but also the $(k-1)$ tags, $y_1, y_2, \ldots, y_{k-1}$, previously modeled. We thus attempt to use any relevant information in the previous tags to improve the model.

The focus in this paper is the PCC method (Dembczyński et al. 2010), which generalizes the CC approach through a probabilistic framework. A probabilistic classifier chain (Dembczyński et al. 2010) estimates the conditional distribution $p(y \mid x)$ using the chain rule of probabilities:

$$p(y \mid x) = p(y_{\pi(1)} \mid x) \prod_{k=2}^{K} p(y_{\pi(k)} \mid x, y_{\pi(1)}, \ldots, y_{\pi(k-1)})$$

where $\pi(\cdot)$ is some fixed permutation/ordering of tags. Thus, learning a multilabel classifier is reduced to learning K independent probabilistic binary classifiers. These independent base classifiers may be, for example, logistic regression models with a specialized feature representation:

$$p(y_{\pi(k)} \mid x, y_{\pi(1)}, \ldots, y_{\pi(k-1)}; \theta) \propto \exp\left(\left\langle \theta_{\pi(k)}, \phi_{\pi(k)}(x, y) \right\rangle\right)$$

for $k = 2$ to $k = K$. In Dembczyński et al. (2010), the choice $\phi_k(x, y) = x \oplus (y_1, \ldots, y_{k-1})$ was used, where $a \oplus b$ is the concatenation of the vectors a and b, so that $\theta_k \in \mathbb{R}^{d+k-1}$. This

means we need to learn $\mathbb{R}^{dK+K(K-1)/2}$ parameters in total, as opposed to $\mathbb{R}^{dK}$ parameters with binary relevance. Suppose we write $\theta_k = [w_k; v_k]$, where $w_k \in \mathbb{R}^d$ and $v_k \in \mathbb{R}^{k-1}$. Then, it may be verified that the joint probability model with a logistic regression base learner is

$$p(y \mid x; \theta) = \frac{\exp(y^T W x + y^T V y)}{\prod_{k=1}^{K}(1 + \exp(Wx + Vy)_k)} \tag{1}$$

where $W = [w_1 \ \ldots \ w_K]$ and $V = [v_1 \ \ldots \ v_K]$. The matrix V is lower-triangular, since we first model $y_{\pi(1)}$, then $y_{\pi(2)}$, and so forth.

3 Analysis of probabilistic classifier chains

Having reviewed the basic idea of PCCs in the previous section, we now study them in more detail. We begin by comparing the model to other probabilistic approaches to multilabel learning. We then analyze some of the advantages of the framework, as well as some of the challenges it poses. Overcoming the latter will form the basis for the rest of the paper.

When analyzing any multilabel learning model, perhaps the first question is how it differs from the binary relevance (BR) baseline. We contrast this model to the one used by PCCs, and then discuss maximum entropy Markov models, conditional random fields and nonlinear base classifiers.

3.1 Connection to binary relevance

In BR, for each tag k, we learn an independent model for $p(y_k \mid x)$. If we use a logistic regression model for each of these probabilities, we have

$$p(y_k \mid x; \theta) \propto \exp\big(\langle w_k, x\rangle\big).$$

We can put these individual tag models together to form the label model

$$p(y \mid x; \theta) = \frac{\exp(y^T W x)}{\prod_{k=1}^{K}(1 + \exp(Wx)_k)}$$

where, as before, $W = [w_1 \ \ldots \ w_K]$. Contrasting this to the PCC model of Eq. (1), the difference is the tag interaction term $y^T V y$. Intuitively, the presence of this term in the PCC model allows it to capture relationships between the various tags.

Another perspective on the difference between the two models can be gained by looking at how PCCs model each $p(y_k \mid x)$. Consider the case of two tags, $K = 2$, with the use of logistic regression for each per-tag model. Suppose we pick the identity permutation for the ordering $\pi(\cdot)$. Let $\sigma(t) = 1/(1 + \exp(-t))$ be the sigmoid function. Then $p(y_1 = 1 \mid x) = \sigma(\langle w_1, x\rangle)$ and

$$\begin{aligned} p(y_2 = 1 \mid x) &= p(y_2 \mid y_1 = 0, x)p(y_1 = 0 \mid x) + p(y_2 \mid y_1 = 1, x)p(y_1 = 1 \mid x) \\ &= \sigma\big(\langle w_2, x\rangle\big)\big(1 - \sigma\big(\langle w_1, x\rangle\big)\big) + \sigma\big(\langle w_2, x\rangle + v_2\big)\sigma\big(\langle w_1, x\rangle\big) \end{aligned}$$

where w_1, w_2 are weight vectors and v_2 is the bias term arising from taking into account the value of y_1 when modeling $p(y_2 \mid y_1, x)$. The model for the first tag is thus identical to BR. However, the model for $p(y_2 \mid x)$ is distinct. First, it is no longer a simple logistic

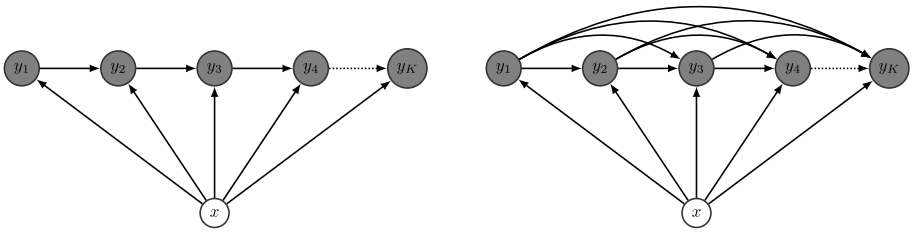

Fig. 1 Graphical models for MEMM (*left*) and PCC (*right*). Variables in white nodes are not generated by the models

regression model. Second, it shares parameters with the model for the first tag. Intuitively, PCCs allow for sharing of parameters between tags, which in turn allows us to leverage any shared structure between the tags.

The above analysis generalizes to more tags, in which case the conditional $p(y_k \mid x)$ involves the product of k sigmoids. This is reminiscent of a neural network model for each tag in BR. However, the key difference is that in PCCs, there is sharing of parameters amongst all tag models. Further, the complexity of the tag models increases as we go further along the chain, as opposed to using identical models for each tag in BR.

3.2 Connections to MEMMs and CRFs

The PCC model is a higher-order extension of the maximum entropy Markov Model (MEMM) (McCallum et al. 2000), used in structured prediction. An MEMM is a Bayesian network that uses log-linear models to model the conditional probability of a reaching a state given an observation and the previous state. However, PCCs are more powerful and expressive than MEMMs because the probability distributions in the chain are conditioned on all the previous tags and not only the preceding tag. The graphical models for PCC and MEMM are shown in Fig. 1. The inference problem in MEMM is tractable using dynamic programming, but in PCC it is not tractable due to the presence of higher-order dependencies among tags. This is precisely the reason for resorting to heuristic inference methods such as beam search.

Conditional random fields (CRFs) (Lafferty et al. 2001) are graphical models that have been successful in many applications. Linear-chain CRFs were first explored for multilabel learning by Ghamrawi and McCallum (2005). Their relationship to PCCs was explored by Dembczyński et al. (2011b). Although CRFs and the base classifiers in PCCs are log-linear models, they are conceptually quite different. It is more computationally expensive to train CRFs, because they have a global partition function $Z(\theta, x) = \sum_{z \in \mathcal{Y}} \exp(\langle \theta, \phi(x, z) \rangle)$, whereas for PCCs we can train a sequence of separate models. Another difference is that, unlike PCCs, linear-chain CRFs cannot capture dependencies among tags of order three or higher.

It is also interesting to consider the consequences of using nonlinear base classifiers in a PCC. Employing a nonlinear classifier such as kernel logistic regression does not change the structure of the graphical model of a PCC, but rather the nature of the individual conditional probability tables. As an example, a joint polynomial kernel defined on input features and tags expands the feature set, but does not modify the dependency structure of the tags. Using a nonparametric model as a base classifier can result in improved conditional probability estimates (Menon et al. 2012), but at the cost of increased time complexity for training the base classifier.

3.3 Advantages and challenges of PCCs

Having contrasted PCCs to existing multilabel learning models at a mathematical level, we now step back and consider whether the model has any particular strengths and/or weaknesses. Indeed, the PCC approach has a number of attractive properties as a multilabel classification method:

(i) It is based on a reduction from multilabel to binary classification, which allows us to leverage existing research on designing scalable and powerful binary classifiers. Compared to the original classifier chain (CC) method (Read et al. 2011), which also uses decomposition, the key difference in this regard is that the decomposition is probabilistically motivated. Also, unlike CCs, PCCs do not use the model's predictions of the past tags during test time inference.

(ii) It is a principled probabilistic model with a theoretical understanding of how it may be used to produce Bayes optimal predictions for a variety of loss functions (Dembczyński et al. 2010). This is in contrast to several multilabel learning methods, where the statistical consistency of the algorithm is unclear. Further, the probabilistic underpinning gives a clear idea on how to modify the algorithm. For example, as we shall see later, the probabilistic setup allows one to design inference methods that are more accurate than the greedy inference described in Read et al. (2011).

(iii) Being a decomposition method, it is computationally inexpensive to train, requiring only marginally more effort than the binary relevance baseline. This is unlike, for example, structured prediction methods (Finley and Joachims 2008) which involve inference during training.

(iv) It is trained on the original label space without any prior transformations (Breiman and Friedman 1997; Weston et al. 2002; Hsu et al. 2009; Bi and Kwok 2011). This is conceptually appealing and makes modifications much simpler. As an example, suppose we want to address the issue of class imbalance at the tag level. One way to do this is to appropriately modify the inputs to the models for each $p(y_k \mid x, y_1, \ldots, y_{k-1})$ by applying cost-sensitive weighting (Elkan 2001). By contrast, in transformation methods, since we lose the relationship to the original label space, it is not clear how modifications in the transformed space affect those in the original space.

Despite these attractive properties, it turns out that there are at least two challenges with using PCCs in practice. These are related to two crucial issues whose discussion we have deferred thus far: how to train them, and how to apply them at test time. At training time, one may maximize the log-likelihood of the given training set, which decomposes into K distinct optimizations for each tag:

$$\begin{aligned}\mathcal{L}\big(\theta; \pi(\cdot)\big) &= \sum_{i=1}^{m} \log p\big(y^{(i)} \mid x^{(i)}; \theta\big) \\ &= \sum_{i=1}^{m} \left[\log p\big(y^{(i)}_{\pi(1)} \mid x^{(i)}; \theta\big) + \sum_{k=2}^{K} \log p\big(y^{(i)}_{\pi(k)} \mid x^{(i)}, y^{(i)}_{\pi(1)}, \ldots, y^{(i)}_{\pi(k-1)}; \theta\big) \right]. \quad (2)\end{aligned}$$

The above equation hides a subtle issue: in theory the chain rule applies regardless of the ordering of the tags, but in practice the ordering can make a big difference. The reason is that the model for each individual tag $p(y_{\pi(k)} \mid x, y_{\pi(1)}, \ldots, y_{\pi(k-1)})$ may be misspecified, in which case some orderings will be better modeled than others. Therefore, we can expect different solutions based on the choice of $\pi(\cdot)$. This prompts the natural question of what the

best ordering $\pi(\cdot)$ is, in the sense of resulting in the highest possible value of $\mathcal{L}(\theta)$. It may seem that one should order the tags in order of some notion of difficulty, but this may not be optimal: for example, a tag that is difficult to model may make subsequent tags considerably easier to model. Thus, a principled algorithmic solution is necessary.

At test time, the problem becomes one of estimating, for a given feature vector x, the most likely set of tags under the learned parameters $\hat{\theta}$:

$$\hat{y} = \operatorname*{argmax}_{y \in \{0,1\}^K} p(y \mid x; \hat{\theta}).$$

This inference is unfortunately computationally intractable. Dembczyński et al. (2010) propose to simply perform brute-force enumeration of all possible labels.

To summarize, we see that there are two main issues with using PCCs in practice. On the *computational* side, the method proposed for test time inference by Dembczyński et al. (2010) requires that we enumerate all 2^K possible candidate labelings, and evaluate them. Indeed, existing applications of PCCs have been restricted to data sets with relatively few number of tags. A general purpose multilabel method should of course handle a large number of tags. On the *accuracy* side, the choice of ordering the tags while training can make a difference in generalization performance. One might hope to do significantly better than just a random ordering. While Dembczyński et al. (2010) proposed taking several random orderings to create an ensemble of PCCs, we would like a more principled procedure, one that searches more intelligently.

There are previous schemes that address the above problems for PCCs. An inference algorithm was proposed by Dembczyński et al. (2011b, 2012) which makes assumptions on the joint probability distribution of labels to guarantee polynomial-time convergence of the algorithm. However, it does not address the problem of learning tag orderings. Our algorithm based on beam search is generic in the sense that it is possible to accommodate a variety of scoring functions into the search algorithm, thereby allowing us to solve both the inference problem and the problem of learning tag orderings. Zaragoza et al. (2011) proposed an algorithm to learn an undirected network of dependencies between the tags. Since this task is intractable in general, they approximate the structure learning problem using the Chow-Liu algorithm to learn a tree dependency structure between tags. However, this tree structure is unlikely to represent many real-world scenarios and it is an empirical question whether such an approximation is good or not.

4 Label inference using beam search

Recall that the inference problem in PCC is $\hat{y} = \operatorname{argmax}_{y \in \mathcal{Y}} p(y \mid x; \hat{\theta})$, i.e., we wish to find the highest scoring label vector. Assuming the probability model is correctly specified, the resulting solution will give the Bayes optimal prediction for subset 0/1 loss (Dembczyński et al. 2010). This inference task is equivalent to finding the optimal path in a rooted, complete binary tree of height K, where each internal vertex at level k denotes a possible partial label vector of length k, so that the leaf vertices represent all possible 2^K label vectors (see Fig. 2). Thus, the inference problem is one of finding the optimal path from the root to one of the leaves in this binary tree, where the score of a vertex v at level k with a corresponding partial label $y^{(v)}$ is equal to the partial probability

$$s_k(v; \hat{\theta}) = p\big(y_1^{(v)} \mid x; \hat{\theta}\big) \prod_{j=2}^{k} p\big(y_j^{(v)} \mid x, y_1^{(v)}, \ldots, y_{j-1}^{(v)}; \hat{\theta}\big), \tag{3}$$

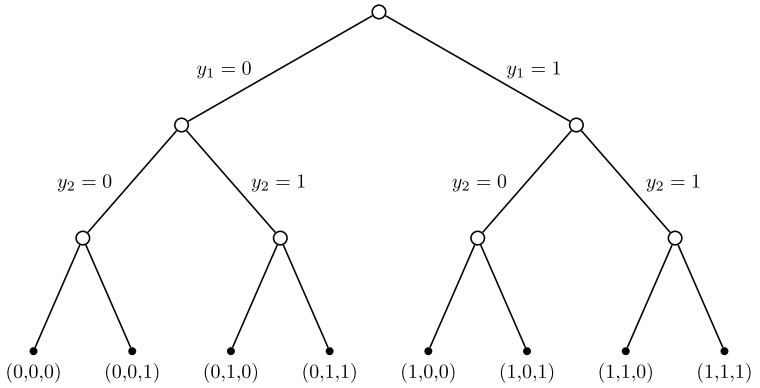

Fig. 2 Binary tree used in beam search for inference with $K = 3$ tags

Algorithm 1 Inference in PCC using beam search

Input: Query point x, learned model parameters $\hat{\theta}$, beam width b
Output: Estimate $\hat{y} = \text{argmax}_y\, p(y \mid x; \hat{\theta})$

$B^{(0)} = \{(1, 0)\}$ {initialize beam}
for $j = 1 \ldots K$ **do**
  $B^{(j)} = \{\}$
  for (parentTags, parentScores) $\in B^{(j-1)}$ **do**
    for $z \in \{0, 1\}$ **do**
      if $p(y_j = z \mid x, \text{parentTags}; \hat{\theta}) > \min\{v : (\cdot, v) \in B^{(j)}\}$ **then**
        $B^{(j)} \leftarrow B^{(j)} \cup (\text{parentTags} \cup \{z\}, p(y_j = z \mid x, \text{parentTags}; \hat{\theta}))$
        $B^{(j)} \leftarrow \text{Top-}b(B^{(j)})$
      end if
    end for
  end for
end for
$\hat{v} = \text{argmax}_v\{v : (\cdot, v) \in B^{(K)}\}$ {highest score}
return $\hat{y} : (\hat{y}, \hat{v}) \in B^{(K)}$

which can be computed recursively. The inference algorithm in the original classifier chain (Read et al. 2011) greedily searches for the optimal path by deciding at each level of the tree whether to traverse in the left or the right direction. However, this may not result in finding the optimal label vector (Dembczyński et al. 2010).

We propose an inference algorithm using beam search (Russell and Norvig 2003), which is a heuristic search technique. A* (Hart et al. 1968) and similar search algorithms could also be used as more sophisticated alternatives to beam search. The basic idea is to keep b candidate solutions at each level of the tree, where b is a user-defined parameter known as the beam width, which represent the best partial solutions seen thus far. We then explore the tree in a breadth-first fashion using these solutions.

The inference procedure for PCCs is described in Algorithm 1. At each level of the tree, we maintain a list of best-scoring candidate vertices of size at most b, where b is the beam

width. We traverse down the tree by considering the children of only those vertices that are in this list, sort them in increasing order of their partial probabilities (3), and prune all the vertices that are not in the top-b list.

The greedy inference algorithm used in classifier chain (Read et al. 2011) is recovered as a special case with beam width $b = 1$. Performing inference by exhaustively enumerating all possible labels is equivalent to doing beam search with $b = \infty$. Thus, by tuning b, we can control the trade-off between computation time and accuracy of the method. The hope is that for real-world multilabel data sets, we can use a relatively small value of b and get performance that is significantly better than the greedy approach, and commensurate with exhaustive enumeration. This is a question that we will answer empirically in Sect. 7.

5 Learning to order tags

Training a PCC involves picking a particular ordering $\pi(\cdot)$ of the tags, based on which we apply the chain rule decomposition. The tag ordering problem is to find the best $\pi(\cdot)$ in the sense of yielding the maximum log-likelihood $\mathcal{L}(\theta; \pi(\cdot))$ in Eq. (2). If each individual tag model is misspecified, this quantity varies based on the choice of the permutation $\pi(\cdot)$. Even if the model is correctly specified, the optimal solution may vary due to finite sample effects. For example, suppose that a tag y_k is rare; then, on any finite sample, we may misestimate $p(y_k = 1 \mid x)$, even if in the infinite sample case we will discover the correct probability.

Intuitively, one may expect the optimal ordering to progressively involve picking the easiest tag to model given the previously picked tags. But it may alternatively be the case that given a difficult tag, subsequent tags are easy to model. A basic question then is to how to determine a suitable tag ordering without resorting to heuristics, or performing exhaustive enumeration over all $K!$ possible orderings.

We propose to use beam search to solve the problem of determining a suitable tag ordering. We do so by casting it as a search problem over a tree. Instead of a complete binary tree as used in the inference algorithm, for the ordering problem we have a tree of height K, where every vertex at level t has $(K - t)$ children, as shown in Fig. 3. Given such a tree, our goal is again to find the optimal path from the root to one of the leaf vertices.

Our procedure to learn tag orderings for PCCs is described in Algorithm 2. Similar to the beam search algorithm used for inference, we use a beam of fixed width b, maintain a list of best-scoring candidate vertices of size at most b and prune all the vertices that are not in the

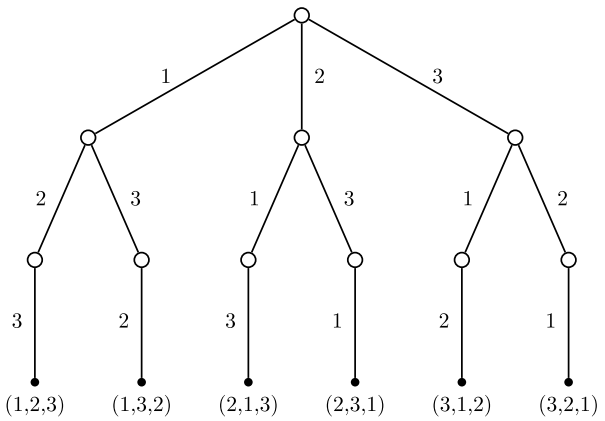

Fig. 3 Example of ordering tree for $K = 3$ tags

Algorithm 2 Learning to order tags in PCC using beam search

Input: Training set $\mathcal{T} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$, beam width b
Output: Model parameters $\hat{\theta}$

$B^{(0)} = \{(1, 0)\}$ {initialize beam}
for $j = 1 \ldots K$ **do**
  $B^{(j)} = \{\}$
  for (parentTags, parentScores) $\in B^{(j-1)}$ **do**
    for $i \in \{1, \ldots, K\} \backslash$parentTags **do**
      $\mathcal{K} \leftarrow k(\phi(x, y_{[\text{parentTags}]}), \phi(x', y'_{[\text{parentTags}]}))$
      $\ell = [y_i^{(1)}; y_i^{(2)}; \ldots; y_i^{(m)}]$
      if $\text{KTA}(\mathcal{K}, \ell) > \min\{v : (\cdot, v) \in B^{(j)}\}$ **then**
        $B^{(j)} \leftarrow B^{(j)} \cup (\text{parentTags} \cup \{i\}, \text{KTA}(\mathcal{K}, \ell))$
        $B^{(j)} \leftarrow \text{Top-}b(B^{(j)})$
      end if
    end for
  end for
end for
$\hat{v} = \text{argmax}_v\{v : (\cdot, v) \in B^{(K)}\}$ {highest score}
Return $\hat{\theta}$ learned by training a PCC using the ordering specified by $\hat{\pi} : (\hat{\pi}, \hat{v}) \in B^{(K)}$

top-b list. We now need to determine the scoring function used to prune the vertices. One possible scoring function is the validation error of classifier, i.e., for every candidate vertex in the tree, we train a (partial) PCC. More specifically, the score of a vertex v at level t is

$$s_t(v; \hat{\theta}) = \sum_{(x,y)\in\mathcal{V}} \log p(y_{\pi_v(t)} \mid x, y_{\pi_v(1)}, \ldots, y_{\pi_v(t-1)}; \hat{\theta})$$

where $\hat{\theta}$ are the parameters of the (partial) PCC that is being evaluated on a validation set of examples $\mathcal{V}$, and the partial tag ordering specified by $\pi_v(\cdot)$ is the directed path from the root to the vertex v. However, this results in training a (partial) PCC at every vertex of the tree which can be prohibitively expensive.

As a computationally cheaper alternative, we propose to instead use kernel target alignment (KTA) (Cristianini et al. 2001) as a measure to score vertices. We want to measure to what extent similar training examples agree on a single given binary tag. Let $y \in \{0, 1\}^m$ be a vector containing the value of this tag for each of the m training examples. The matrix $yy^\top$ is a kernel matrix based on this tag. Let $\mathcal{K}$ be the kernel matrix based on the feature vector representing each of the m examples. Let $\langle A, B\rangle_F = \sum_{ij} A_{ij} B_{ij}$ denote the Frobenius inner product between two matrices A and B. The kernel target alignment between the matrices $\mathcal{K}$ and $yy^\top$ is

$$\text{KTA}(\mathcal{K}, y) = \frac{\langle \mathcal{K}, yy^\top \rangle_F}{\sqrt{\langle \mathcal{K}, \mathcal{K} \rangle_F \langle yy^\top, yy^\top \rangle_F}}.$$

The KTA score is often more efficient to compute than training a (partial) PCC. (Indeed, this is true in the empirical study reported in Sect. 7.) Note that there are also hidden costs with training a classifier, such as performing cross-validation to determine regularization and other hyperparameters. Intuitively, the KTA score can be considered as a proxy for

the accuracy of a classifier trained on the same input features and outputs and therefore it is reasonable to expect the KTA scores to correlate positively with the accuracies of a classifier.

The KTA score of a vertex v at level t is computed by constructing a kernel matrix whose entries are

$$k\big(\phi(x, y_{\pi_v([t-1])}), \phi\big(x', y'_{\pi_v([t-1])}\big)\big)$$

where $k(\cdot,\cdot)$ is the kernel function, $\phi(x,z) = x \otimes z$, i.e., the Kronecker product of x and z, for cross-product features and $\phi(x,z) = x \oplus z$ for concatenated features, and $y_{\pi_v([t-1])} = (y_{\pi_v(1)}, \ldots, y_{\pi_v(t-1)})$. For linear kernels, the kernel matrix factorizes into the product of the kernel matrix defined on the input features and the kernel matrix defined on the output labels. Note that earlier, the log-likelihood scoring function led to a naturally additive objective function. While the same can be done with KTA, it is an empirical question whether this will be appropriate or not. Observe in particular that an alternative is to use the product of KTA scores, which treats the KTA as a surrogate for the raw probability score itself.

6 Discussion

This section explores several issues related to the focus of this paper, and comments on possible extensions of the algorithms above.

6.1 Is subset 0/1 loss reasonable?

The inference problem in PCC is to compute $\hat{y} = \operatorname{argmax}_{y \in \mathcal{Y}} p(y \mid x; \hat{\theta})$. This prediction is optimal if the goal is to minimize the subset 0/1 loss of the model (Dembczyński et al. 2010). However, when minimizing a different loss function, such as Hamming loss, then a different inference problem needs to be solved, and the beam search strategy is not applicable. A natural question, then, is whether subset 0/1 loss is a reasonable loss function to optimize in practice.

A disadvantage of subset 0/1 loss is that it is very strict: even if we make a single mistake on one of the individual tags, we do not get "partial credit." However, this loss may be preferable to alternatives in some cases. Specifically, suppose that the marginal distributions for each tag are highly unbalanced, i.e., $p(y_k)(1 - p(y_k)) \approx 0$ for each $k \in \{1, \ldots, K\}$. Further, suppose that the joint distribution $p(y_1, \ldots, y_K)$ is not concentrated around any particular value. Then, we may achieve a small per-tag metric like Hamming loss by predicting the majority tag value $\operatorname{argmax}_{v \in \{0,1\}} p(y_k = v)$ for each k, independent of the example x. This is intuitively dissatisfying, because the predictions are constant across examples, and so do not help to discriminate between them. (This is related to the fact that in unbalanced binary classification problems, the 0/1 loss for a majority class prediction is small.) However, such a prediction will incur a high subset 0/1 loss. Such settings are common in practical applications of multilabel learning, and in particular on many benchmark data sets (Luaces et al. 2012). Therefore, inference for subset 0/1 loss is indeed useful in some cases, and important.

6.2 Modeling non-chain dependencies with PCCs

Recall that if we write the PCC model as

$$p(y \mid x; \theta) = \frac{\exp(y^T W x + y^T V y)}{\prod_{k=1}^{K}(1 + \exp((Wx + Vy)_k))} = \prod_{k=1}^{K} \frac{\exp(y^T (Wx + Vy)_k)}{1 + \exp((Wx + Vy)_k)},$$

we impose the constraint that V is lower triangular. In fact, this constraint is essential for the model in Eq. (1) to be a valid probability distribution. If we drop the assumption, then we have to consider the model

$$p(y \mid x; \theta) = \frac{1}{Z(\theta, x)} \cdot \prod_{k=1}^{K} \frac{\exp(y^T (Wx + Vy)_k)}{1 + \exp((Wx + Vy)_k)},$$

where the normalizer is

$$Z(\theta, x) = \sum_{y' \in \{0,1\}^K} \prod_{k=1}^{K} \frac{\exp(y'^T (Wx + Vy')_k)}{1 + \exp((Wx + Vy')_k)}.$$

When V is lower triangular, each term in the summation is independent, and so $Z(\theta, x) \equiv 1$. When V does not have this structure, however, computing this normalizer efficiently is difficult. Further, it is unclear if the resulting model is log-concave, since PCC is not based on the log-linear framework. Thus, one can think of the ordering problem as a sacrifice one has to make to get around the general intractability of normalizing the probability model $p(y \mid x)$.

The idea of not using a lower triangular V is related, but not identical, to the idea of modeling each tag conditional on other tags, as used in the structured output-associative regression (SOAR) method (Bo and Sminchisescu 2009), in which

$$p\big(y_k = 1 \mid x, y^{(-k)}; \theta\big) \propto \exp\big(\langle w_k, x \rangle + \big\langle \psi_k, y^{(-k)} \big\rangle\big)$$

where $y^{(-k)}$ represents all tags but y_k, and $\psi_k \in \mathbb{R}^{K-1}$. The reason it is not identical to the PCC approach is that these equations do not uniquely determine the label distribution. In the two-tag case, for example, knowing just $p(y_1 \mid y_2, x)$ and $p(y_2 \mid y_1, x)$ will only tell us the value of the ratio $p(y_1 \mid x)/p(y_2 \mid x)$, but not the individual terms, and so it is not possible to compute the joint distribution. Thus, SOAR does not have a clear probabilistic interpretation in terms of maximizing the likelihood of a well-defined model of the label sequence.

6.3 Predicting accurate probabilities

PCCs rely on predicting accurate probabilities for each classifier in the chain. But most multilabel data sets are unbalanced, i.e., every label has very few positive instances. Using (linear) logistic regression or similar will give biased probability estimates (King and Zeng 2001). At a minimum, better results can be obtained by post-processing the scores by isotonic regression, which is a nonparametric technique to find a monotone fit to a set of target values. In a learning context, the method was used by Zadrozny and Elkan (2002)

to learn meaningful probabilities from the scores of an input classifier. This can be taken a step further by post-processing the scores produced by optimizing the area under the ROC curve directly (Menon et al. 2012). The main advantage of such an approach is that it will automatically handle the class imbalance issue, while also producing accurate probabilities.

6.4 Scalability issues in using cross-product features

Learning linear models with cross-product features may pose scalability issues for high-dimensional data sets with large number of tags. In such cases, we can efficiently compute the tensor-product kernel matrix which is the element-wise product of the kernel matrices on input features and the output labels. If the number of training instances is not high, then we can use kernel logistic regression as the base classifier in PCCs. Otherwise, we can compute a low-dimensional representation of the feature space given the tensor-product kernel matrix using, for example, kernel principal components analysis (Schölkopf et al. 1998; Ham et al. 2004), and use the extracted features to train a linear classifier. An alternative approximation is to extract features from (a subset of) the rows of the kernel matrix (Schölkopf et al. 1999; Williams and Seeger 2000; Drineas and Mahoney 2005; Balcan et al. 2006, 2008) and train a linear logistic regression on these features.

6.5 Alternatives to beam search

An alternative to beam search for learning the tag ordering and for inference is to use sampling methods (Dembczyński et al. 2011a, 2012; Read et al. 2012). It is possible to design a Markov chain Monte Carlo (MCMC) method using the Metropolis-Hastings algorithm (Metropolis et al. 1953; Hastings 1970) for inference and for learning the tag ordering. However, analyzing the mixing time of these chains is an interesting direction for future work. For combinatorial spaces such as the vertices of a hypercube and the space of all permutations, we can use tools from MCMC theory (Jerrum and Sinclair 1996; Randall 2003) to design sampling algorithms with a rigorous analysis of their mixing times. Furthermore, it may be possible to design exact sampling algorithms with runtime guarantees using techniques like coupling from the past (Propp and Wilson 1996; Huber 1998).

7 Experiments

We report experiments on the data sets listed in Table 1. We selected all the benchmark multilabel data sets from Tsoumakas et al. (2011) that have fewer than 100 tags and fewer

Table 1 Details of benchmark multilabel data sets (Tsoumakas et al. 2011)

Data set	# training inst. (m)	# test inst.	# features (d)	# tags (K)
Emotions	391	202	72	6
Scene	1211	1196	294	6
Yeast	1500	917	103	14
Genbase	463	199	1186	27
Medical	333	645	1449	45
Enron	1123	579	1001	53

than 10 K training instances. These limits allow all models to be trained easily using batch optimization methods with both concatenated and cross-product features. Note that on the majority of these data sets, inference by exhaustive enumeration is either computationally expensive or intractable. All data sets have a pre-defined test set, and our reported results are on this set. We compare the following algorithms:

(a) Binary relevance (BR): This is the baseline algorithm where we train separate independent logistic regressors for each tag.
(b) Kernel dependency estimation (KDE): This is the algorithm proposed by Weston et al. (2002). Here, a (linear) transformation using principal components analysis is applied to the original label matrix in order to decorrelate the tags. Then, independent regressors are trained in the transformed label space.
(c) Probabilistic classifier chain (PCC): We use the original formulation as described in Dembczyński et al. (2010) but with beam search as inference and the original ordering of tags found in the data sets.
(d) PCC with logistic regression using beam search for *both* learning the tag ordering and inference. To learn the tag ordering, we use kernel target alignment as the scoring function in beam search. Note that, in this setting, the output of beam search for learning is the tag ordering which is then used at a later stage to train a PCC.

We evaluate the performance of algorithms using the following loss functions, where y and $\hat{y}$ are the target and the predicted labels respectively:

(i) Subset 0/1 loss:

$$\ell_{0/1}(y, \hat{y}) = [\![y \neq \hat{y}]\!]$$

(ii) Hamming loss:

$$\ell_h(y, \hat{y}) = \sum_{i=1}^{K} [\![y_i \neq \hat{y}_i]\!] \quad \text{and}$$

(iii) Ranking loss:

$$\ell_r(y, \hat{y}) = \sum_{(i,j):y_i > y_j} \left([\![\hat{y}_i < \hat{y}_j]\!] + \frac{1}{2} [\![\hat{y}_i = \hat{y}_j]\!] \right)$$

As mentioned above, BR is theoretically optimal for Hamming and ranking losses, and it has been noted previously that it is a strong baseline for other loss functions also (Read et al. 2011; Dembczyński et al. 2010). This is especially true if the BR classifier for each tag is regularized. Some previous studies, such as Dembczyński et al. (2010), use an unregularized base classifier, for which BR may be misleadingly suboptimal. In all our experiments, we use regularized linear models and tune the regularization parameter using cross-validation.

We use the Friedman test (Friedman 1937, 1940) with Bonferroni-Dunn post-hoc analysis (Dunn 1961) to evaluate the statistical significance of results from multiple algorithms and data sets (Demšar 2006). The following subsections provide experimental results and analysis for a number of related questions.

7.1 What is the effect of beam width, used for inference and for learning tag orderings, on the performance of PCCs?

We choose three out of the six data sets for this experiment, namely, *Emotions*, *Scene* and *Yeast*, where it is computationally feasible to do inference by exhaustive enumeration of

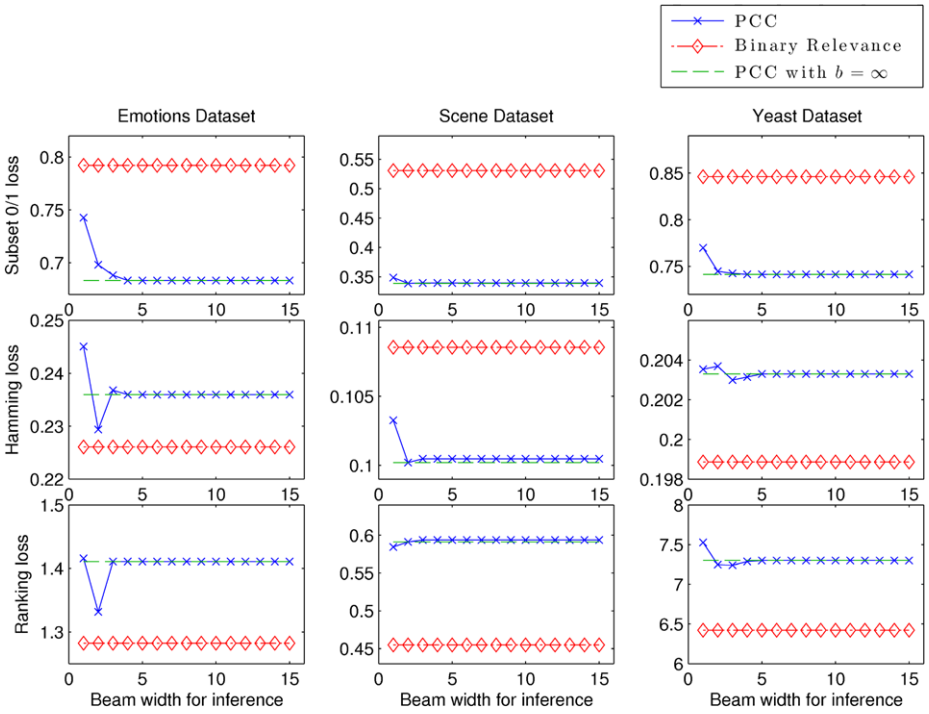

Fig. 4 Effect of beam width used in inference on the performance of PCCs

all possible labels, which allows us to study the effect of beam width on inference and learning the tag ordering. We compare the performance of (i) binary relevance (BR) and (ii) probabilistic classifier chain (PCC) with the original tag ordering in the data sets and using beam search for inference for several values of beam width b.

Figure 4 shows the performance of the algorithms measured in terms of subset $0/1$ loss, Hamming loss and ranking loss with varying beam width. From the figure, we see that the test set performance of PCC converges rapidly with $b < 15$ to the performance obtained with exhaustive enumeration, especially for the subset $0/1$ loss. We also observe that with $b = 15$, a significant fraction of the true labels in the test set are among the candidate labels found by beam search, even if the single best label found by beam search is not exactly correct. Figure 5 shows the maximum of the cosine similarities between all the labels in the beam and the true label averaged across all the examples in the test set for varying beam width. It is clear that at $b = 15$, a significant fraction of the true labels appear in the beam.

Figure 4 shows that for Hamming and ranking losses, the loss at certain values of beam width is lower than the loss obtained by exhaustive enumeration, which is surprising at first glance since exhaustive enumeration should ideally provide a lower bound for the test set loss. However, in beam search, labels are scored according to the conditional probability $p(y \mid x)$ which may not necessarily correspond to the optimal result. Indeed, one of the points made by Dembczyński et al. (2010) was that taking $\operatorname{argmax}_{y \in \mathcal{Y}} p(y \mid x)$ gives the optimal result for subset $0/1$ loss, but for Hamming and ranking losses the optimal result is for tag k, i.e., $\operatorname{argmax}_{b \in \{0,1\}} p(y_k = b \mid x)$.

We also analyze the effect of increasing the beam width used in beam search to determine the tag ordering on the classifier performance. Figure 6 shows the performance of PCCs with

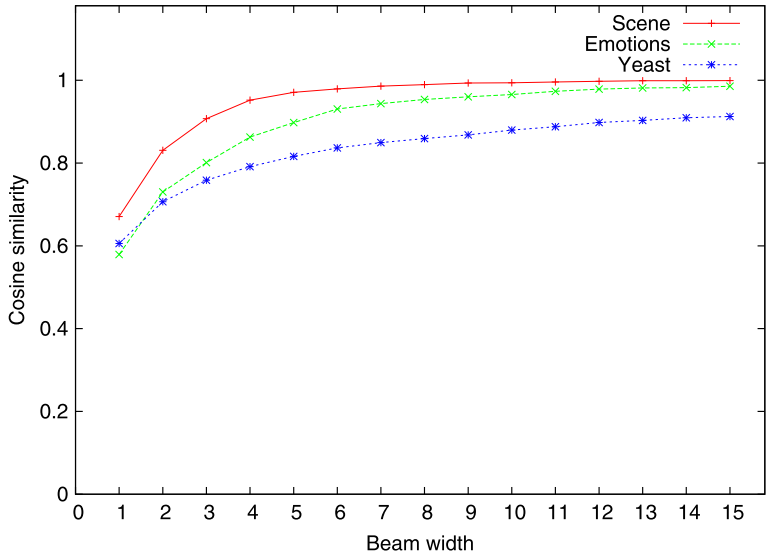

Fig. 5 Maximum of the cosine similarities between all the labels in the beam and the true label averaged across all the examples in the test set for varying beam width

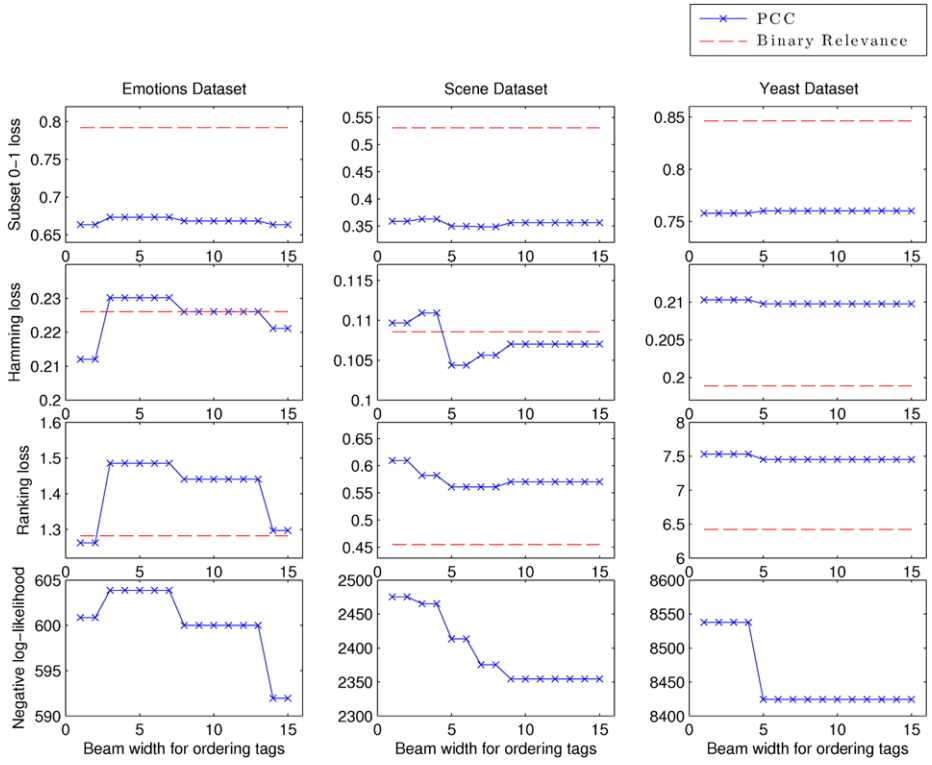

Fig. 6 Effect of beam width used to learn the tag ordering on the performance of PCCs

varying beam width on the three data sets *Emotions*, *Scene* and *Yeast*. The test time inference is done by exhaustive enumeration of all possible labels. There is no clear pattern with varying beam width for subset 0/1, Hamming and ranking losses. However, negative log-likelihood, in general, is non-increasing with increasing beam width, especially for larger widths. This confirms that beam search using KTA as the scoring function is successful at finding orderings that lead to improvements in negative log-likelihood; as the amount of search done by beam search increases, better orderings are found.

7.2 How does beam search for inference compare to the approximate inference method of Dembczyński et al. (2012)?

The results are shown in Table 2, using a beam width of 15 in beam search inference. The inference method of Dembczyński et al. (2012) with $\epsilon = 0$ and $\epsilon = 0.5$ corresponds to exact inference and approximate inference with greedy search respectively. The beam search for inference performs better than the approximate inference method for both $\epsilon = 0.5$ and $\epsilon = 0.25$ on the subset 0/1 loss. For Hamming and ranking losses, it performs slightly worse than the approximate inference method with $\epsilon = 0.25$. In all the cases, exact inference ($\epsilon = 0$) outperforms both the beam search inference and the approximate inference method with $\epsilon > 0$ as expected.

We conducted a Friedman test with Bonferroni-Dunn post-hoc analysis to evaluate the statistical significance of the results. The critical values for four classifiers at 5 % and 10 % significance levels are 2.394 and 2.128 respectively, and the corresponding critical differences are 1.784 and 1.586 respectively (Demšar 2006). For subset 0/1 loss, the difference in

Table 2 Test set performance of PCCs with beam search inference and approximate inference method of Dembczyński et al. (2012) measured in terms of subset 0/1 loss (*top*), Hamming loss (*middle*) and ranking loss (*bottom*) on the benchmark data sets. The *last column* in each table shows the ranking of algorithms averaged across all the data sets, with lower ranks being better

	Emotions	Scene	Yeast	Genbase	Medical	Enron	Avg. rank
PCC/$b = 15$	0.6832	0.3863	0.7634	0.0201	0.4	0.8497	2.17
PCC/$\epsilon = 0.5$	0.7327	0.4064	0.7884	0.0201	0.4078	0.8636	3.75
PCC/$\epsilon = 0.25$	0.698	0.3821	0.7666	0.0201	0.4	0.8532	2.5
PCC/$\epsilon = 0$	0.6832	0.3821	0.7612	0.0201	0.4	0.848	1.58
	Emotions	Scene	Yeast	Genbase	Medical	Enron	Avg. rank
PCC/$b = 15$	0.2261	0.1113	0.2092	0.001	0.012	0.0462	2.42
PCC/$\epsilon = 0.5$	0.2294	0.1165	0.2123	0.001	0.012	0.0464	3.33
PCC/$\epsilon = 0.25$	0.2285	0.1099	0.2081	0.001	0.012	0.0464	2.41
PCC/$\epsilon = 0$	0.2252	0.1099	0.208	0.0009	0.012	0.0462	1.83
	Emotions	Scene	Yeast	Genbase	Medical	Enron	Avg. rank
PCC/$b = 15$	1.3218	0.5978	7.5071	0.1859	1.7752	13.0846	2.33
PCC/$\epsilon = 0.5$	1.3564	0.6275	7.5343	0.1935	1.7395	13.0259	3.33
PCC/$\epsilon = 0.25$	1.3465	0.6033	7.3991	0.1935	1.738	13.0242	2.25
PCC/$\epsilon = 0$	1.3614	0.6024	7.3272	0.1935	1.738	13.0207	2.08

average ranks between PCC/$b = 15$ and PCC/$\epsilon = 0.5$ (greedy search) is $3.75 - 2.17 = 1.58$, which is slightly less than 1.586 as needed for statistical significance at 10 % level. For Hamming and ranking losses, the differences in average ranks are not statistically significant. We conclude that the experimental results are not sufficient to make any statistically significant statements regarding the performance of the inference methods.

7.3 Does learning to order tags improve the performance of PCCs when compared to PCCs trained using a random or predefined ordering?

The results are shown in Table 3. We use the notation PCC/$\mathcal{O}/\mathcal{I}$ to denote variants of PCC where $\mathcal{O} \in \{\text{or}, 1, 5, 15\}$ is used to denote the tag ordering determined by beam search with $b = 1, 5$ or 15 and the original tag ordering in the data sets (or), and $\mathcal{I} \in \{1, 5, 15\}$ is used to denote the beam width ($b = 1, 5$ or 15) used in inference. All variants of PCC outperform binary relevance for the subset 0/1 loss. Note that PCC/or/1 is the variant of PCC which uses the original tag ordering in the data sets and greedy search for inference, i.e., beam search with $b = 1$. All variants of PCC that use beam search for inference and/or beam search to determine the tag ordering outperform, or are on par with, PCC/or/1. This result demonstrates the advantages of using beam search for PCCs.

On a majority of the data sets, variants of PCC that use beam search to determine the tag ordering using KTA as the scoring function give the best results. For Hamming and ranking losses, binary relevance is a strong baseline and outperforms PCCs on average, especially for ranking loss, confirming the results reported by Dembczyński et al. (2010). Nevertheless, PCCs using beam search to determine the tag ordering perform, on average, better than PCCs that use the original tag ordering.

We conducted a Friedman test with Bonferroni-Dunn post-hoc analysis. The critical values for nine classifiers at the 5 % significance level is 2.724, and the corresponding critical difference is 4.307 (Demšar 2006). For subset 0/1 loss, the differences in average ranks between the pairs (BR, PCC/1/5), (BR, PCC/5/5) and (BR, PCC/5/15) are statistically significant. However, the differences in average ranks between the pairs (BR, PCC/or/1), (BR, PCC/or/5) and (BR, PCC/or/15) are not statistically significant. We conclude that using beam search for *both* inference and learning leads to statistically significant better performance by PCC. For Hamming loss, the differences are not statistically significant. For ranking loss, the performance of BR is statistically better than PCC/or/1, but not the other variants of PCC. Using beam search for inference and learning improves the performance of PCCs, and thereby makes differences between BR and PCC not statistically significant.

The beam search inference method is designed to minimize subset 0/1 loss, and not other loss functions such as Hamming and ranking loss. However, using beam search to learn the tag ordering leads to superior models for the conditional probability distribution $p(y \mid x)$ when compared to those estimated by PCCs trained with a random tag ordering. To minimize loss functions such as Hamming and ranking loss, we can perform exhaustive inference, perhaps using sampling methods to make the computation tractable. Therefore, using beam search for learning the tag ordering has the potential to improve the performance of PCCs for not only subset 0/1 loss but also other loss functions. To support this argument, we compare the negative log-likelihood of PCCs trained with the ordering determined by beam search (PCC/15/15) and the original ordering (PCC/or/15). The results are shown in Table 4. Five out of six data sets show an improvement in negative log-likelihood for the PCCs trained using the beam search ordering.

Table 3 Test set performance of binary relevance (BR), kernel dependency estimation (KDE) and probabilistic classifier chain (PCC) measured in terms of subset 0/1 loss (*top*), Hamming loss (*middle*) and ranking loss (*bottom*) on the benchmark data sets. The last column in each table shows the ranking of algorithms averaged across all the data sets, with lower ranks being better

	Emotions	Scene	Yeast	Genbase	Medical	Enron	Avg. rank
BR	0.7921	0.5309	0.8462	0.0201	0.417	0.8774	7.83
KDE	0.7822	0.6204	0.8397	0.0201	0.431	0.9016	8
PCC/or/1	0.7475	0.4022	0.7895	0.0201	0.4093	0.867	6.25
PCC/or/5	0.6832	0.3863	0.7634	0.0201	0.4	0.8497	4.08
PCC/or/15	0.6832	0.3863	0.7634	0.0201	0.4	0.8497	4.08
PCC/1/1	0.7228	0.4022	0.807	0.0201	0.4124	0.8566	6.25
PCC/1/5	0.6634	0.3813	0.7612	0.0201	0.4031	0.8411	2.83
PCC/5/5	0.6634	0.3855	0.7601	0.0201	0.4031	0.8411	2.83
PCC/15/15	0.6634	0.3813	0.7601	0.0201	0.4031	0.8428	2.83
	Emotions	Scene	Yeast	Genbase	Medical	Enron	Avg. rank
BR	0.2261	0.1086	0.1989	0.001	0.0122	0.0463	5.17
KDE	0.2236	0.1204	0.1984	0.0008	0.0127	0.0465	5.58
PCC/or/1	0.2368	0.1145	0.2131	0.001	0.0122	0.0465	7.67
PCC/or/5	0.2261	0.1113	0.2092	0.001	0.012	0.0462	5.17
PCC/or/15	0.2261	0.1113	0.2092	0.001	0.012	0.0462	5.17
PCC/1/1	0.2228	0.1104	0.2204	0.001	0.0122	0.0461	5.5
PCC/1/5	0.2112	0.1086	0.2113	0.001	0.012	0.0461	3.58
PCC/5/5	0.2129	0.1095	0.2106	0.001	0.012	0.0461	3.83
PCC/15/15	0.2129	0.1056	0.2106	0.001	0.012	0.0461	3.33
	Emotions	Scene	Yeast	Genbase	Medical	Enron	Avg. rank
BR	1.2822	0.4548	6.4209	0.1709	1.6271	12.9378	1.33
KDE	1.4307	0.5084	6.4384	0.0452	1.3659	14.7219	4.17
PCC/or/1	1.3911	0.612	7.7121	0.1859	1.7798	13.0743	7.5
PCC/or/5	1.3218	0.5978	7.5071	0.1859	1.7752	13.0846	5.42
PCC/or/15	1.3218	0.5978	7.5071	0.1859	1.7752	13.0846	5.42
PCC/1/1	1.3317	0.5397	7.5474	0.1884	1.7395	13.0708	5.42
PCC/1/5	1.3564	0.5485	7.6619	0.1884	1.7333	13.0656	6.08
PCC/5/5	1.3366	0.5293	7.6314	0.1884	1.7333	13.0639	4.92
PCC/15/15	1.3366	0.5	7.6336	0.1884	1.7333	13.0639	4.75

We conclude this subsection with an experiment where we train PCCs with orderings determined by sorting the tags based on (i) KTA and (ii) cross-validation error on the training set. The results are shown in Tables 5 and 6. When compared to the results in Table 3, we see that PCCs trained with orderings determined by beam search outperform, in the majority of cases, those trained with the heuristic of ordering tags by cross-validation error.

Table 4 Comparison of negative log-likelihood of PCCs trained with the tag ordering determined by beam search (PCC/15/15) and the original tag ordering from the data sets (PCC/or/15)

	Emotions	Scene	Yeast	Genbase	Medical	Enron
PCC/or/5	602.88	2261.51	8720.82	40.81	1222.36	4122.26
PCC/15/15	574.19	2104.53	8224.96	40.81	1219.47	4101.87

Table 5 Test set performance of PCCs trained with orderings determined by sorting the tags based on KTA on the training set

	0/1 loss	Hamming loss	Ranking loss
Emotions	0.6683	0.2178	1.3811
Scene	0.3923	0.1078	0.5410
Yeast	0.7612	0.2113	7.6641
Genbase	0.0201	0.0010	0.1884
Medical	0.4031	0.0120	1.7333
Enron	0.8497	0.0462	13.0656

Table 6 Test set performance of PCCs trained with orderings determined by sorting the tags based on cross-validation errors on the training set

	0/1 loss	Hamming loss	Ranking loss
Emotions	0.6584	0.2153	1.3713
Scene	0.3963	0.1095	0.5401
Yeast	0.7601	0.2113	8.0927
Genbase	0.0201	0.0010	0.1834
Medical	0.3969	0.0120	1.7349
Enron	0.8463	0.0464	13.5147

7.4 Does learning to order tags with beam search improve the performance of PCCs when compared to PCCs trained with an ensemble of random orderings?

We trained ensemble PCCs (EPCC) (Dembczyński et al. 2010) using a set of 15 random orderings and compared their performance with PCCs trained using the ordering estimated by the beam search method with $b = 15$ (PCC/15/15). The results are shown in Table 7. Inference in all PCCs was done using beam search with $b = 15$. We also include results from an ensemble consisting of binary relevance (BR) and PCC/15/15. PCC/15/15 performed better than BR and the ensemble methods for subset 0/1 loss. Since BR does not perform well on this loss function, the combination BR + PCC/15/15 results in no improvement. For subset 0/1 loss, we conclude that PCCs trained using beam search for learning the tag ordering perform best.

For Hamming loss, there is no major difference in performance on the large data sets *Enron*, *Medical* and *Genbase*. On two of the other three data sets, either PCC/15/15 or BR + PCC/15/15 performs better than EPCC. We therefore conclude that for Hamming loss, there is no significant gain from using an ensemble method with random orderings (EPCC). For ranking loss, BR + PCC/15/15 performs better than EPCC on four out of six data sets, and either plain BR or BR + PCC/15/15 performs better than EPCC on five out of six data sets. Therefore, for ranking loss also, there is no significant gain from using EPCC.

These findings contradict those in Dembczyński et al. (2010), where EPCC was found to perform better than competing methods. We believe that using regularized logistic regres-

Table 7 Test set performance of BR and PCCs trained with the ordering determined by beam search and an ensemble of random orderings measured in terms of subset 0/1 loss (*top*), Hamming loss (*middle*) and ranking loss (*bottom*) on the benchmark data sets. The *last row* in each table shows the ranking of algorithms averaged across all the data sets, with lower ranks being better

	BR	PCC/15/15	EPCC	BR + PCC/15/15
Emotions	0.7921	0.6634	0.7079	0.7722
Scene	0.5309	0.3813	0.4164	0.5133
Yeast	0.8462	0.7601	0.7677	0.8375
Genbase	0.0201	0.0201	0.0201	0.0201
Medical	0.417	0.4031	0.4016	0.417
Enron	0.8774	0.8428	0.8531	0.8721
Avg. rank	3.67	1.42	1.92	3

	BR	PCC/15/15	EPCC	BR + PCC/15/15
Emotions	0.2261	0.2129	0.2136	0.2219
Scene	0.1086	0.1056	0.1037	0.1061
Yeast	0.1989	0.2106	0.2049	0.1981
Genbase	0.001	0.001	0.0009	0.0009
Medical	0.0122	0.012	0.012	0.012
Enron	0.0463	0.0461	0.0461	0.047
Avg. rank	3.42	2.33	1.83	2.42

	BR	PCC/15/15	EPCC	BR + PCC/15/15
Emotions	1.2822	1.3366	1.2772	1.2425
Scene	0.4548	0.5	0.4732	0.4632
Yeast	6.4209	7.6336	6.7056	6.4285
Genbase	0.1709	0.1884	0.1909	0.1708
Medical	1.6271	1.7333	1.7194	1.8
Enron	12.9378	13.0639	12.8912	13.6787
Avg. rank	1.67	3.5	2.5	2.33

sion as the base classifier boosts the performance of binary relevance as well as of PCCs, minimizing the potential gain from ensembles.

We conducted a Friedman test with Bonferroni-Dunn post-hoc analysis. The critical values for four classifiers at 5 % and 10 % significance levels are 2.394 and 2.128 respectively, and the corresponding critical differences are 1.784 and 1.586 respectively (Demšar 2006). The differences in performance above between PCC/15/15 and both ensemble methods, EPCC and BR + PCC/15/15, are not statistically significant.

7.5 How does the choice of feature representation affect the performance of PCCs?

We compare the performance of PCCs trained with concatenated features (Table 3) and cross-product features (Table 8). The relative performance of different algorithms is similar

Table 8 Test set performance of binary relevance (BR), kernel dependency estimation (KDE) and probabilistic classifier chain (PCC) trained with *cross-product features* measured in terms of subset 0/1 loss (*top*), Hamming loss (*middle*) and ranking loss (*bottom*) on the benchmark data sets. The *last column* in each table shows the ranking of algorithms averaged across all the data sets, with lower ranks being better

	Emotions	Scene	Yeast	Genbase	Medical	Enron	Avg. rank
BR	0.7921	0.5309	0.8462	0.0201	0.417	0.8774	7.83
KDE	0.7822	0.6204	0.8397	0.0201	0.431	0.9016	8
PCC/or/1	0.7426	0.3487	0.7699	0.0201	0.3891	0.8273	6
PCC/or/5	0.6832	0.3395	0.7416	0.0201	0.3643	0.8169	3.67
PCC/or/15	0.6832	0.3395	0.7416	0.0201	0.3643	0.8169	3.67
PCC/1/1	0.6832	0.3829	0.7666	0.0201	0.3798	0.8394	6
PCC/1/5	0.6733	0.362	0.7579	0.0201	0.3597	0.8048	3.25
PCC/5/5	0.6733	0.3495	0.759	0.0201	0.3597	0.81	3.5
PCC/15/15	0.6634	0.3478	0.7601	0.0201	0.3597	0.81	3.08

	Emotions	Scene	Yeast	Genbase	Medical	Enron	Avg. rank
BR	0.2261	0.1086	0.1989	0.001	0.0122	0.0463	3.67
KDE	0.2236	0.1204	0.1984	0.0008	0.0127	0.0465	3.5
PCC/or/1	0.245	0.1033	0.2035	0.001	0.0148	0.0488	5.67
PCC/or/5	0.236	0.1005	0.2033	0.001	0.0132	0.0506	4.83
PCC/or/15	0.236	0.1005	0.2035	0.001	0.0132	0.0508	5.58
PCC/1/1	0.2351	0.1189	0.2154	0.001	0.0122	0.0507	6.42
PCC/1/5	0.2302	0.1105	0.2106	0.001	0.0114	0.0516	5.83
PCC/5/5	0.2302	0.1044	0.2094	0.001	0.0114	0.0507	4.58
PCC/15/15	0.2211	0.1058	0.2098	0.001	0.0114	0.0517	4.92

	Emotions	Scene	Yeast	Genbase	Medical	Enron	Avg. rank
BR	1.2822	0.4548	6.4209	0.1709	1.6271	12.9378	1.33
KDE	1.4307	0.5084	6.4384	0.0452	1.3659	14.7219	2.33
PCC/or/1	1.4158	0.5844	7.5267	0.191	3.6922	16.1105	5.83
PCC/or/5	1.4109	0.5936	7.3021	0.191	3.7186	16.8929	6.33
PCC/or/15	1.4109	0.5936	7.301	0.191	3.7202	16.9326	6.67
PCC/1/1	1.5347	0.6112	7.6194	0.1834	2.9093	16.4447	6.25
PCC/1/5	1.4851	0.592	7.5463	0.1834	3.0783	16.9197	6.5
PCC/5/5	1.4851	0.561	7.4297	0.1884	3.0682	16.8765	5.42
PCC/15/15	1.297	0.5284	7.4526	0.1884	3.0682	16.6123	4.33

for both feature representations. For subset 0/1 loss, we find improvements in performance when using cross-product features, $\phi(x, y) = x \otimes y$. However, for Hamming and ranking losses, cross-product features appear to degrade the performance of classifiers when compared to features formed by concatenating labels, $\phi(x, y) = x \oplus y$. The improvement in performance for subset 0/1 loss may be due to an improved estimate of the conditional probability distribution $p(y \mid x)$ resulting from the cross-product feature representation, and to the fact that beam search inference is tailored for subset 0/1 loss.

We conducted a Friedman test with Bonferroni-Dunn post-hoc analysis to evaluate the statistical significance of results in Table 8. The critical values for nine classifiers at 5 % and 10 % significance levels are 2.724 and 2.498 respectively, and the corresponding critical differences are 4.307 and 3.95 respectively (Demšar 2006). For subset 0/1 loss, the differences in average rank for the pairs (BR, PCC/1/5), (BR, PCC/5/5) and (BR, PCC/5/15) are statistically significant, similar to the findings with concatenated features. Furthermore, both PCCs with the original tag ordering perform statistically significantly better than BR at the 10 % significance level. Recall that the differences in average rank for the pairs (BR, PCC/or/1), (BR, PCC/or/5) and (BR, PCC/or/15) are not statistically significant with concatenated features. For Hamming and ranking losses, the findings are similar to those with concatenated features.

8 Conclusions

Empirical results clearly demonstrate the benefit of using beam search for test time inference, and for learning a good ordering of tags. We believe that these are important extensions to probabilistic classifier chains. We summarize the main conclusions from our experiments below.

(i) We did not find statistically significant differences between beam search inference and exact inference or the approximate inference method of Dembczyński et al. (2012). Nevertheless, when compared to exact inference, our beam search inference is computationally tractable.

(ii) Experiments show that using beam search for *both* inference and learning leads to statistically significant improvements in the performance of PCCs when compared to binary relevance for subset 0/1 loss. However, for Hamming and ranking losses the differences in performance are not statistically significant.

(iii) We did not find statistically significant improvements from using ensemble methods. We believe this is due to using fine-tuned regularized logistic regression as the base classifier, and to using the beam search method for inference and learning the tag ordering in PCCs.

(iv) Experiments with cross-product features resulted in statistically significant improvements in the performance of PCCs when compared to binary relevance for subset 0/1 loss, even when the PCCs were trained with the original tag ordering but used beam search for inference. By contrast, PCCs trained with concatenated features and the original tag ordering did not achieve statistically significant gains, thereby providing evidence for the benefit of using cross-product features in PCCs, at least when optimizing the subset 0/1 loss.

References

Balcan, M. F., Blum, A., & Vempala, S. (2006). Kernels as features: on kernels, margins, and low-dimensional mappings. *Machine Learning*, *65*(1), 79–94.

Balcan, M. F., Blum, A., & Srebro, N. (2008). A theory of learning with similarity functions. *Machine Learning*, *72*(1–2), 89–112.

Bi, W., & Kwok, J. T. (2011). Multilabel classification on tree- and DAG-structured hierarchies. In *Proceedings of the twenty-eighth international conference on machine learning*.

Bo, L., & Sminchisescu, C. (2009). Structured output-associative regression. In *Proceedings of the IEEE computer society conference on computer vision and pattern recognition*.

Breiman, L., & Friedman, J. H. (1997). Predicting multivariate responses in multiple linear regression. *Journal of the Royal Statistical Society. Series B. Statistical Methodology*, *59*, 3–54.

Cristianini, N., Shawe-Taylor, J., Elisseeff, A., & Kandola, J. S. (2001). On kernel-target alignment. In *Advances in neural information processing systems* (Vol. 14).

Dembczyński, K., Cheng, W., & Hüllermeier, E. (2010). Bayes optimal multilabel classification via probabilistic classifier chains. In *Proceedings of the twenty-seventh international conference on machine learning*.

Dembczyński, K., Waegeman, W., Cheng, W., & Hüllermeier, E. (2011a). An exact algorithm for F-measure maximization. In *Advances in neural information processing systems* (Vol. 24).

Dembczyński, K., Waegeman, W., & Hüllermeier, E. (2011b). Joint mode estimation in multi-label classification by chaining. In *Proceedings of the workshop on collective learning and inference on structured data at the European conference on machine learning and principles and practice of knowledge discovery in databases*.

Dembczyński, K., Waegeman, W., & Hüllermeier, E. (2012). An analysis of chaining in multi-label classification. In *Proceedings of the twentieth European conference on artificial intelligence*.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1–30.

Drineas, P., & Mahoney, M. W. (2005). On the Nyström method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, *6*, 2153–2175.

Dunn, O. J. (1961). Multiple comparisons among means. *Journal of the American Statistical Association*, *56*, 52–64.

Elkan, C. (2001). The foundations of cost-sensitive learning. In *Proceedings of the seventeenth international joint conference on artificial intelligence*.

Finley, T., & Joachims, T. (2008). Training structural SVMs when exact inference is intractable. In *Proceedings of the twenty-fifth international conference on machine learning*.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, *32*, 675–701.

Friedman, M. (1940). A comparison of alternative tests of significance for the problem of *m* rankings. *The Annals of Mathematical Statistics*, *11*, 86–92.

Ghamrawi, N., & McCallum, A. (2005). Collective multi-label classification. In *Proceedings of the ACM fourteenth conference on information and knowledge management*.

Ham, J., Lee, D. D., Mika, S., & Schölkopf, B. (2004). A kernel view of the dimensionality reduction of manifolds. In *Proceedings of the twenty-first international conference on machine learning*.

Hart, P., Nilsson, N., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, *4*(2), 100–107.

Hastings, W. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, *57*(1), 97–109.

Hsu, D., Kakade, S., Langford, J., & Zhang, T. (2009). Multi-label prediction via compressed sensing. In *Advances in neural information processing systems* (Vol. 22).

Huber, M. (1998). Exact sampling and approximate counting techniques. In *Proceedings of the thirtieth annual ACM symposium on the theory of computing*.

Jerrum, M., & Sinclair, A. (1996). The Markov chain Monte Carlo method: An approach to approximate counting and integration. In *Approximation algorithms for NP-hard problems* (pp. 482–520). Boston: PWS-Kent.

King, G., & Zeng, L. (2001). Logistic regression in rare events data. *Political Analysis*, *9*(2), 137–163.

Kumar, A., Vembu, S., Menon, A. K., & Elkan, C. (2012). Learning and inference in probabilistic classifier chains with beam search. In *Proceedings of the European conference on machine learning and principles and practice of knowledge discovery in databases*.

Lafferty, J. D., McCallum, A., & Pereira, F. C. N. (2001). Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning*.

Luaces, O., Díez, J., Barranquero, J., del Coz, J., & Bahamonde, A. (2012). Binary relevance efficacy for multilabel classification. *Progress in Artificial Intelligence*, *1*, 303–313.

McCallum, A., Freitag, D., & Pereira, F. C. N. (2000). Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of the seventeenth international conference on machine learning*.

Menon, A. K., Jiang, X., Vembu, S., Elkan, C., & Ohno-Machado, L. (2012). Predicting accurate probabilities with a ranking loss. In *Proceedings of the twenty-ninth international conference on machine learning*.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., & Teller, E. (1953). Equation of state calculation by fast computing machines. *Journal of Chemical Physics*, *21*, 1087–1092.

Propp, J. G., & Wilson, D. B. (1996). Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures & Algorithms*, *9*(1–2), 223–252.

Randall, D. (2003). Mixing. In *Proceedings of the fourty-fourth annual IEEE symposium on foundations of computer science*.

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

Read, J., Martino, L., & Luengo, D. (2012). Efficient Monte Carlo optimization for multi-dimensional classifier chains. arXiv:1211.2190.

Russell, S., & Norvig, P. (2003). *Artificial intelligence: a modern approach* (2nd ed.). Englewood Cliffs: Prentice-Hall.

Schölkopf, B., Smola, A. J., & Müller, K. R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, *10*(5), 1299–1319.

Schölkopf, B., Mika, S., Burges, C. J. C., Knirsch, P., Müller, K. R., Rätsch, G., & Smola, A. J. (1999). Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, *10*(5), 1000–1017.

Sorower, M. S. (2010). A literature survey on algorithms for multi-label learning. Tech. rep., Oregon State University, Corvallis, OR, USA.

Tsoumakas, G., & Katakis, I. (2007). Multi-label classification: an overview. *International Journal of Data Warehousing and Mining*, *3*(3), 1–13.

Tsoumakas, G., Katakis, I., & Vlahavas, I. P. (2010). Mining multi-label data. In *Data mining and knowledge discovery handbook* (pp. 667–685). Berlin: Springer.

Tsoumakas, G., Spyromitros-Xioufis, E., Vilcek, J., & Vlahavas, I. (2011). Mulan: a Java library for multi-label learning. *Journal of Machine Learning Research*, *12*, 2411–2414.

Weston, J., Chapelle, O., Elisseeff, A., Schölkopf, B., & Vapnik, V. (2002). Kernel dependency estimation. In *Advances in neural information processing systems* (Vol. 15).

Williams, C. K. I., & Seeger, M. (2000). Using the Nyström method to speed up kernel machines. In *Advances in neural information processing systems* (Vol. 13).

Zadrozny, B., & Elkan, C. (2002). Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining*.

Zaragoza, J., Sucar, L., & Morales, E. (2011). Bayesian chain classifiers for multidimensional classification. In *Proceedings of the twenty-second international joint conference on artificial intelligence*.