A Two-Stage Conditional Random Field Model Based Framework for Multi-Label Classification

Abhiram Kumar Singh(✉) and C. Chandra Sekhar

Department of Computer Science and Engineering,
Indian Institute of Technology Madras, Chennai, India
{abhi,chandra}@cse.iitm.ac.in

Abstract. Multi-label classification (MLC) deals with the task of assigning an instance to all its relevant classes. This task becomes challenging in the presence of the label dependencies. The MLC methods that assume label independence do not use the dependencies among labels. We present a two-stage framework which improves the performance of MLC by using label dependencies. In the first stage, a standard MLC method is used to get the confidence scores for different labels. A conditional random field (CRF) is used in the second stage that improves the performance of the first-stage MLC by using the label dependencies among labels. An optimization-based framework is used to learn the structure and parameters of the CRF. Experiments show that the proposed model performs better than the state-of-the-art methods for MLC.

Keywords: Label dependence · Conditional Random Field · Multi-label Classification

1 Introduction

In the single-label classification (SLC) problem, each data instance is assigned to one class out of two or more classes. However, in real world tasks, an object can have multiple labels. For example, a news article may have multiple topics, an image may have multiple labels and a medical diagnosis may lead to multiple diseases. Multi-label classification (MLC) [1] deals with the task of assigning such instances to all its relevant classes.

Traditional methods for MLC either transform the MLC problem into several SLC problems (problem transformation methods) or adapt an SLC method for multi-label datasets (algorithm adaptation methods). These methods assume the label independence and may give inconsistent output. For example, an instance may be assigned to two mutually exclusive labels. A method that can correct these errors due to inconsistencies by exploiting the label dependencies is likely to give an improved performance.

We present a framework based on the conditional random field (CRF) that tries to correct the erroneous output from a multi-label classifier by using the dependencies among labels. Results of our studies show that capturing dependencies among the class labels significantly improves the performance of MLC.

The rest of the paper is organised as follows. In Sect. 2, we present a brief review of methods for using the label dependencies in MLC. Section 3 presents the proposed framework that uses the CRF to capture the label dependencies and then use the dependencies to correct the errors in the output of an MLC model. In Sect. 4, we present our experimental studies and results.

2 Approaches to Capture Label Correlations

Capturing label correlations and using them for multi-label learning is important for MLC. We review some of the methods for capturing the correlations among labels.

Classifier chain [2] is based on the chain rule decomposition of the joint probability distribution where each factor in the chain decomposition is realized using a binary classifier. The input to a classifier in the chain is augmented with the output from the previous binary classifiers in the chain. The limitation of this method is that the performance depends on the chain order. Ensemble of classifier chains [2] mitigate the problem of performance dependence on the chain order by taking the average over predictions obtained using different chain orders. A Bayesian network is used in [3] to learn the relationship among the labels. Then, it uses the classifier chain method where the topological ordering of labels in the Bayesian network is considered as the chain order and the feature vector is augmented with the output from the parent class classifier. In [4], a cyclic directed graphical model is used to capture the relationships among labels. The model is built by learning a binary classifier for a label given all other labels and input features. Then the Gibbs sampling is used for inference. In [5], a two stage binary relevance method is used. In this method, the input to the second stage of binary classifiers is augmented with the output from the binary classifiers in the first stage.

Methods for MLC using the undirected graphical model have been proposed in [6–9]. In [6], a pairwise Markov random field is used for joint prediction of labels. Similarly, in [7,8], a pairwise CRF is used where a tree-structured graph is constructed to identify the set of informative label pairs in [7]. In [8], a fully connected graph with the pairwise clique potentials is used.

3 Enhancing Multi-label Classification Using Label Dependencies

We propose a two-stage framework for multi-label classification. In the first stage, one of the MLC classifiers such as Binary Relevance (BR) [1], ML-kNN [10] or an ensemble of classifiers chains (ECC) is used. In the second stage, the output of MLC in the first stage is refined by using the dependencies among labels captured by a CRF model.

Let $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n), 1 \leq n \leq N\}$ be the multi-label data where $\mathbf{x}_n \in \Re^d$ is the d-dimensional input instance and $\mathbf{y} = \{y_1, y_2, ..., y_m\}$ is the m-dimensional desired output vector. Here, m is the number of class labels and $y_j \in \{0,1\}$. MLC deals with learning the mapping $h : \Re^d \rightarrow \{0,1\}^m$.

In the BR method for MLC, the multi-label dataset is transformed into m binary classification datasets. In the j^{th} dataset, the instances are considered as the positive instances if they belong to the j^{th} class, otherwise they are considered as the negative instances. Any SLC method can be used to build each of the m classifiers. Prediction for a test instance is obtained from the outputs of the m classifiers. The ML-kNN method is an algorithm adaptation method based on the k-nearest neighbour (kNN) classification for SLC. For a given test instance, the ML-*kNN* first identifies its k-nearest neighbours. Then the prediction is obtained using the Bayes rule based on the statistical information obtained from the neighbours.

Let $\mathbf{s} = \{s_1, s_2, ..., s_m\}$ be the set of confidence scores obtained from the first stage where $s_j \in [0,1]$ is the output of the classifier corresponding to the j^{th} class for a given instance $\mathbf{x}$.

3.1 Conditional Random Field

Conditional Random Field (CRF) [11] is a discriminative undirected probabilistic graphical model that directly models the conditional probability distribution $p(\mathbf{y}|\mathbf{s})$, where $\mathbf{y}$ is the set of output variables and $\mathbf{s}$ is the set of observed input variables as shown in Fig. 1. In the proposed method, the set of confidence scores $\mathbf{s}$ obtained from the first stage are used as the input to the CRF. The graph associated with the CRF encodes the dependencies among the output variables. An edge between two nodes in the graph indicates that the corresponding variables are dependent on each other. The conditional probability distribution $p(\mathbf{y}|\mathbf{s})$ is given by the normalized product of clique potentials.

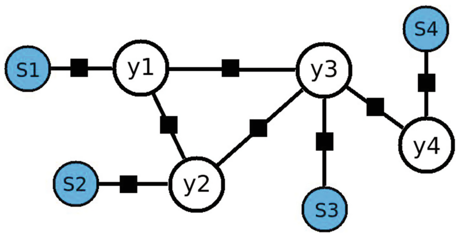

Fig. 1. A factor graph representation of the proposed CRF based model. The unshaded circles represent the class variables $\mathbf{y}$. The shaded circles represent the input variables $\mathbf{s}$. The edges amongst nodes represent the dependencies among class variables. The solid blocks represent the factors associated with those variables.

We use a CRF with the pairwise potentials to model the dependencies among the labels $\mathbf{y}$ using the output $\mathbf{s}$ from the first stage. Let $G = (V, E)$ be the graph associated with the CRF. The nodes V of the graph represents the class variables and the edges E represents the dependence relationships among class variables. The conditional distribution $p(\mathbf{y}|\mathbf{s})$ is given by

$$p(\mathbf{y}|\mathbf{s}) = \frac{1}{Z(\mathbf{s})} \prod_{i \in V} \Phi_i(y_i, \mathbf{s}) \prod_{(i,j) \in E} \psi_{ij}(y_i, y_j, \mathbf{s}) \tag{1}$$

where Φ_i is the node potential associated with i^{th} node and ψ_{ij} is the edge potential associated with the (i,j) edge. The normalization constant $Z(\mathbf{s})$, also known as the partition function is given by

$$Z(\mathbf{s}) = \sum_{\mathbf{y}} \left[\prod_{i \in V} \Phi_i(y_i, \mathbf{s}) \prod_{(i,j) \in E} \psi_{ij}(y_i, y_j, \mathbf{s}) \right] \tag{2}$$

For the binary variable $y_i \in \{0,1\}$, the node potential Φ_i for different assignments of y_i is given by

$$\Phi_i(y_i, \mathbf{s}) = \left(e^{f_i(\mathbf{s}) v_i^0}, e^{f_i(\mathbf{s}) v_i^1} \right) \tag{3}$$

where v_i^0 and v_i^1 are the node parameters corresponding to the state $y_i = 0$ and $y_i = 1$ respectively, and $f_i(\mathbf{s}) = s_i$ is the node feature.

Similarly, the edge potential ψ_{ij} for different assignments of edge $(i,j) = \{00, 01, 10, 11\}$ is defined by

$$\psi_{ij}(y_i, y_j, \mathbf{s}) = \begin{pmatrix} e^{\mathbf{f}_{ij}(\mathbf{s}) \mathbf{w}_{ij}^{0,0}} & e^{\mathbf{f}_{ij}(\mathbf{s}) \mathbf{w}_{ij}^{0,1}} \\ e^{\mathbf{f}_{ij}(\mathbf{s}) \mathbf{w}_{ij}^{1,0}} & e^{\mathbf{f}_{ij}(\mathbf{s}) \mathbf{w}_{ij}^{1,1}} \end{pmatrix} \tag{4}$$

where $\mathbf{f}_{ij}(\mathbf{s}) = [s_i, s_j]^T$ are the edge features and $(\mathbf{w}_{ij}^{0,0}, \mathbf{w}_{ij}^{0,1}, \mathbf{w}_{ij}^{1,0}, \mathbf{w}_{ij}^{1,1})$ are the edge parameters.

Let $\boldsymbol{\theta} = [\mathbf{v}, \mathbf{w}]$ be the combined parametric vector and the respective feature functions be combined as $F(\mathbf{s}, \mathbf{y})$. The Eq. (1) can now be written succinctly as

$$p(\mathbf{y}|\mathbf{s}) = \frac{1}{Z(\boldsymbol{\theta}, \mathbf{s})} exp \left(\boldsymbol{\theta}^T F(\mathbf{s}, \mathbf{y}) \right) \tag{5}$$

3.2 Objective Function

The objective function for learning the CRF parameters, the negative log likelihood (nll) is given as

$$nll(\boldsymbol{\theta}) = -\sum_{n=1}^{N} log\ p(\mathbf{y}_n|\mathbf{s}_n) = -\sum_{n=1}^{N} \left[\boldsymbol{\theta}^T F(\mathbf{s}_n, \mathbf{y}_n) - log\ Z(\boldsymbol{\theta}, \mathbf{s}_n) \right] \tag{6}$$

The gradient for the negative log likelihood [12] is given by

$$\nabla nll(\boldsymbol{\theta}) = -\sum_{n=1}^{N} \left[F(\mathbf{s}_n, \mathbf{y}_n) - E_{\mathbf{y}'} \left[F(\mathbf{s}, \mathbf{y}') \right] \right] \tag{7}$$

where $E_{\mathbf{y}'}\left[F\left(\mathbf{s},\mathbf{y}'\right)\right] = \sum_{\mathbf{y}'} p\left(\mathbf{y}'|\mathbf{s}\right) F\left(\mathbf{s},\mathbf{y}'\right)$ are the expectations for the feature functions. To find these expectations, we have to run an inference algorithm to compute model distribution $p\left(\mathbf{y}'|\mathbf{s}\right)$ for all values of $\mathbf{y}'$. This makes computing gradient very expensive. Two main solutions to address this issue are: (a) use an approximate inference algorithm such as loopy belief propagation and (b) use a surrogate objective function such as pseudo-likelihood. We consider the second method that uses the pseudo-likelihood. The negative log pseudo-likelihood (nlpl) for a CRF is given by

$$nlpl\left(\boldsymbol{\theta}\right) = -\sum_{n=1}^{N} log\ PL\left(\mathbf{y}_n|\mathbf{s}_n\right) = -\sum_{n=1}^{N}\sum_{i\in V} log\ p\left(y_{i,n}|\mathbf{y}_{\mathcal{N}_{i},n},\mathbf{s}_n;\boldsymbol{\theta}\right) \quad (8)$$

where $\mathbf{y}_{\mathcal{N}_i,n}$ is the set of neighbours $\mathcal{N}_i$ for the i^{th} node and the n^{th} instance. The negative log pseudo-likelihood is a convex function in parameters $\boldsymbol{\theta}$ and known to be a consistent estimator, i.e., it returns the same set of parameters as the maximum likelihood estimate for $\boldsymbol{\theta}$ when the number of instances goes to infinity [15].

Using the concise notation,

$$p\left(y_i|\mathbf{y}_{\mathcal{N}_i},\mathbf{s};\boldsymbol{\theta}\right) = \frac{1}{Z_i\left(\boldsymbol{\theta}_i,\mathbf{s}\right)} exp\left(\boldsymbol{\theta}_i^T F_i\left(\mathbf{s},\mathbf{y}\right)\right) \quad (9)$$

where $\boldsymbol{\theta}_i = \left(\mathbf{v}_i, \{\mathbf{w}_{ij}\}_{j\in\mathcal{N}_i}\right)$ are the parameters corresponding to i^{th} node and its neighbours, Z_i is the local partition function, and F_i is the local feature vector. The local partition function Z_i can be computed by summing only over the values of y_i.

3.3 CRF Structure and Parameter Learning

The structure of a CRF can be learnt by minimizing the regularized negative log pseudo-likelihood function with L_1 regularization [13]. The L_1 norm based regularization is known to give a sparse solution. We impose L_1 regularization for each set of parameters associated with the edges in the graph [14]. This causes sparsity in the edge weight parameters where all parameters associated with a specific edge go to zero simultaneously. Using L_2 regularizer for the node parameters, the regularization term $R(\boldsymbol{\theta})$ can be written as

$$R(\boldsymbol{\theta}) = \lambda_1 \left\|\mathbf{v}\right\|_2^2 + \lambda_2 \sum_{b\in E} \left\|\mathbf{w}_b\right\|_2 \quad (10)$$

where $\mathbf{w}_b = (\mathbf{w}_{ij}^{0,0}, \mathbf{w}_{ij}^{0,1}, \mathbf{w}_{ij}^{1,0}, \mathbf{w}_{ij}^{1,1})$ is the set of weight parameters for different configuration of the edge $b = (i,j)$. Parameters of the CRF are found by minimizing the regularized loss function as given below

$$\boldsymbol{\theta}^* = argmin_{\boldsymbol{\theta}}(nlpl\left(\boldsymbol{\theta}\right) + R(\boldsymbol{\theta})) \quad (11)$$

We use the projected quasi-Newton [16] method to solve the above optimization problem. The structure of the CRF then corresponds to all edges in the graph that has non-zero weight parameters. After fixing the structure of the CRF, the L_2 norm regularization is used over the edge parameters. The limited-memory BFGS [17] method is used to further fine-tune the model's parameters for the given structure. After training the model, the loopy belief propagation method is used to obtain the final predictions.

4 Experiments

We performed the experiments on the following multi-label datasets; Emotion, Enron, Medical, Scene and Yeast from Mulan [18].

The evaluation metrics used to compare the various methods are: Accuracy, Subset-accuracy (exact match) and Hamming loss [1].

Table 1. Accuracy comparison of different single-stage MLC methods(BR, ML-kNN and ECC) with the proposed two-stage method using CRF.

Dataset	Method					
	BR	CRF_{BR}	ML-kNN	CRF_{ML-kNN}	ECC	CRF_{ECC}
Emotions	0.5360	0.5701	0.3366	0.4745	0.5850	0.6163
Enron	0.4059	0.4704	0.3321	0.3853	0.4620	0.4701
Medical	0.6450	0.6877	0.4428	0.5674	0.7410	0.7615
Scene	0.5836	0.7099	0.6353	0.7333	0.7030	0.7274
Yeast	0.5270	0.5416	0.5202	0.5435	0.5660	0.5692

We compared the performance of the proposed method with different existing methods for MLC. The BR, ML-kNN and ECC based MLC are used in the first stage. Logistic regression with L_2 regularization is used as the base classifier for BR method. SVMs were used as base classifiers for ECC. For ML-kNN, we used the code released on the internet by the author. We used the UGM-toolbox [19] for CRF implementation. Other MLC methods were implemented using *MEKA*[1]. All hyper-parameters are tuned using the cross-validation method.

The performance of proposed two-stage method using different MLCs in the first stage is presented in Table 1. For all the three MLC methods, the CRF based two-stage method is able to enhance the performance. The improvement is more significant in datasets that have a high correlation among class labels. Table 2 presents the comparison of the proposed method against the other existing methods. The proposed method performs better than all other methods. This shows the effectiveness of capturing label dependencies for MLC.

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

Table 2. Performance comparison of our proposed method (CRF_{ECC}) with other state-of-the-art-methods: Collective Multi-Label classification (CML) [8], Meta Binary Relevance (MBR) [5] and Conditional Dependency Network (CDN) [4]

Dataset	Method	Accuracy	Exact-match	Hamming loss
Emotions	CML	0.5664	0.3465	0.2244
	MBR	0.5850	0.3470	0.1910
	CDN	0.5840	0.3230	**0.1820**
	CRF_{ECC}	**0.6163**	**0.3861**	0.1914
Enron	CML	0.4319	0.1399	0.0575
	MBR	0.4370	0.1490	**0.0490**
	CDN	0.4670	0.1360	0.0540
	CRF_{ECC}	**0.4701**	**0.1606**	0.0508
Medical	CML	0.7209	0.6450	0.0113
	MBR	0.6990	0.6140	0.0120
	CDN	0.6460	0.5190	0.0150
	CRF_{ECC}	**0.7615**	**0.6698**	**0.0109**
Scene	CML	0.6198	0.5493	0.1282
	MBR	0.6090	0.5730	0.0860
	CDN	0.6580	0.5680	0.1020
	CRF_{ECC}	**0.7274**	**0.6706**	**0.0842**
Yeast	CML	0.4662	0.1897	0.2565
	MBR	0.5300	0.2070	**0.1900**
	CDN	0.5170	0.1620	0.2170
	CRF_{ECC}	**0.5692**	**0.2225**	0.1967

5 Conclusion

In this paper, we proposed a two-stage framework for multi-label classification using the conditional random field. It captures the dependencies among labels to improve the MLC performance. An optimization-based framework is used for learning the structure of the CRF. Experimental results shows the effectiveness of the proposed method for benchmark multi-label datasets.

References

1. Zhang, M.-L., Zhou, Z.-H.: A review on multi-label learning algorithms. IEEE Trans. Knowl. Data Eng. **26**(8), 1819–1837 (2014)
2. Read, J., Bernhard, F.P., Holmes, G., Frank, E.: Classifier chains for multi-label classification. Mach. Learn. **85**(3), 333–359 (2011)
3. Zhang, M.-L., Zhang, K.: Multi-label learning by exploiting label dependency. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data mining, pp. 999–1008. ACM (2010)

4. Guo, Y., Gu, S.: Multi-label classification using conditional dependency networks. In: IJCAI Proceedings-International Joint Conference on Artificial Intelligence, vol. 22, p. 1300 (2011)
5. Godbole, S., Sarawagi, S.: Discriminative methods for multi-labeled classification. In: Dai, H., Srikant, R., Zhang, C. (eds.) PAKDD 2004. LNCS, vol. 3056, pp. 22–30. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24775-3_5
6. Arias, J., Gamez, J.A., Nielsen, T.D., Puerta, J.M.: A scalable pairwise class interaction framework for multidimensional classification. Int. J. Approximate Reasoning **68**, 194–210 (2016)
7. Li, X., Zhao, F., Guo, Y.: Multi-label image classification with a probabilistic label enhancement model. In: Proceedings of Uncertainty in Artificial Intelligence (2014)
8. Ghamrawi, N., McCallum, A.: Collective multi-label classification. In: Proceedings of the 14th ACM International Conference on Information and Knowledge Management, pp. 195–200. ACM (2005)
9. Naeini, M.P., Batal, I., Liu, Z., Hong, C., Hauskrecht, M.: An optimization-based framework to learn conditional random fields for multi-label classification. In: Proceedings of the 2014 SIAM International Conference on Data Mining, pp. 992–1000. Society for Industrial and Applied Mathematics (2014)
10. Zhang, M.-L., Zhou, Z.-H.: ML-KNN: a lazy learning approach to multi-label learning. Pattern Recogn. **40**(7), 2038–2048 (2007)
11. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proceedings of the Eighteenth International Conference on Machine Learning, ICML, vol. 1, pp. 282–289 (2001)
12. Murphy, K.P.: Machine Learning: A Probabilistic Perspective. MIT press, Cambridge (2012)
13. Schmidt, M.W., et al.: Structure learning in random fields for heart motion abnormality detection. In: CVPR, vol. 1(1) (2008)
14. Yuan, M., Lin, Y.: Model selection and estimation in regression with grouped variables. J. R. Stat. Soc. Ser. B (Stat. Methodol.) **68**(1), 49–67 (2006)
15. Besag, J.: Efficiency of pseudolikelihood estimation for simple Gaussian fields. Biometrika **64**(3), 616–618 (1977)
16. Schmidt, M.W., Van Den Berg, E., Friedlander, M.P., Murphy, K.P.: Optimizing costly functions with simple constraints: a limited-memory projected quasi-Newton Algorithm. In: AISTATS, vol. 5 (2009)
17. Liu, D.C., Nocedal, J.: On the limited memory BFGS method for large scale optimization. Math. Prog. **45**(1), 503–528 (1989)
18. Tsoumakas, G., Spyromitros-Xioufis, E., Vilcek, J., Mulan, I.V.: A Java library for multi-label learning. J. Mach. Learn. Res. **12**, 2411–2414 (2011)
19. Schmidt, M.: UGM: a Matlab toolbox for probabilistic undirected graphical models (2007). http://www.cs.ubc.ca/~schmidtm/Software/UGM.html