Metric Learning for Multi-label Classification

Marco Brighi(✉), Annalisa Franco, and Dario Maio

DISI, University of Bologna, Cesena, Italy
{marco.brighi6,annalisa.franco,dario.maio}@unibo.it

Abstract. This paper proposes an approach for multi-label classification based on metric learning. The approach has been designed to deal with general classification problems, without any assumption on the specific kind of data used (images, text, etc.) or semantic meaning assigned to labels (tags, categories, etc.). It is based on clustering and metric learning algorithm aimed at constructing a space capable of facilitating and improving the task of classifiers. The experimental results obtained on public benchmarks of different nature confirm the effectiveness of the proposal.

Keywords: Metric learning · Multi-label classification · Clustering · Supervised learning.

1 Introduction and Related Works

Multi-label classification plays an important role in the context of data analysis in many different applications, ranging from text classification to multimedia annotation or bioinformatics [9]. The general problem can be stated as follows. Let $Y = \{y_j, j = 1, .., m\}$ be a finite set of m class labels and let $D = \{(\mathbf{x}_i, Y_{\mathbf{x}_i}), i = 1, .., n\}$ be a generic set of labeled patterns, where $\mathbf{x}_i$ is a data pattern and $Y_{\mathbf{x}_i} \subseteq Y$ the set of related labels. In multi-label classification each pattern $\mathbf{x}_i$ is therefore generally assigned to more than one class. The objective of multi-label classification is to determine a function $f(\mathbf{x}_i)$ able to correctly identify all the labels $Y_{\mathbf{x}_i}$ that can be associated to each pattern in the domain. Traditional classifiers are not feasible in this case, and an extension to the multi-label case is needed. Most of the commonly used algorithms have their own multi-label variant; as an alternative, some works in the literature propose approaches able to transform multi-label problems into more canonical multi-class ones [4]. Interested readers can refer to [5] for a description of the existing approaches. Recently, some approaches for multi-label classification have been proposed, based on neural networks and deep learning, such as [18] and [13]. These techniques achieve in general good performance but require, for a proper training, a large amounts of data that are not always available. The solution proposed in this work is aimed at designing a method for improving performance of natively multi-label classifiers for general pattern classification problems, in the presence of datasets of limited dimensions.

A. Torsello et al. (Eds.): S+SSPR 2020, LNCS 12644, pp. 24–33, 2021.
https://doi.org/10.1007/978-3-030-73973-7_3

In an ideal situation, just like in a traditional multi-class problem, the patterns sharing the same labels should lay in the same sub-region of the feature space. Unfortunately, this desirable situation seldom occurs in real cases, since labels can generally represent high-level concepts used by humans for classification that do not have a direct and immediate counterpart in terms of feature similarity; of course, the multi-label scenario further complicates the problem. A common approach to derive a better representation space is based on the use of algorithms capable of learning a specific metric, which can properly group patterns belonging to the same classes. To this purpose, several techniques based on supervised learning have been proposed such as Local Fisher Discriminant Analysis [16] (LFDA) or Large Margin Nearest Neighbor Metric Learning [19] (LMNN). Unfortunately, such approaches cannot be easily extended to the multi-label case. An easy and intuitive solution to this problem is to consider super-sets of labels by transforming the multi-label problem into a multi-class one. This approach is commonly referred to as *label powerset* (see for instance [17]) and it maps each combination of labels to a unique class changing the nature of the problem; each new class created corresponds to a list of labels of the original problem. This approach is well-suited for problems where the number of labels and their possible combinations are limited, otherwise the risk is to obtain an extremely high number of super-sets, i.e. to many classes to deal with.

This paper proposes a novel approach aimed at improving the multi-label classifier accuracy by the adoption of a metric learning algorithm. The authors of [11] present a method in order to learn a new distance metric maximizing the margin present between different instances. To achieve the expected result, they project data and labels into the same embedding space imposing constraints on distances such that instances with very different multiple labels will be moved far away. Unlike the previous solutions, [8] proposes an innovative loss function based on the Jaccard index able to determine how different two instances are based on their labels set. Using the Adam optimisation algorithm, they learn a new metric able to improve the classification performance.

The approach proposed in this work is based on label clustering, meaning that the patterns in the dataset are clustered according to their labels rather than on the basis of the related feature vectors. The information produced by clustering is exploited to learn a metric aimed at improving classification accuracy. In [9] the authors use clustering algorithms for pruning infrequent multi labels. They assume that the elimination of infrequently multi label instances from the training set leads to the identification of better label power sets. Another interesting use of clustering can be found in [20]. The authors address the multi-label classification problem using the classifier chain approach, training a single classifier for each label. The use of a clustering algorithm, *k-means* in this case, allows to discover any correlations present between the labels. This information allows to identify the correct order in which to arrange the classifiers, maximizing the final performance.

The proposed approach has been designed as a general solution capable of dealing with multi-label classification problems. For this reason the experimental evaluation will consider heterogeneous datasets containing data of different nature. The paper is organized as follows: Sect. 2 presents the proposed approach, its

characteristics and steps, Sect. 3 shows a complete overview of the results achieved and Sect. 4 draws some conclusions and presents possible future research directions.

2 The Proposed Approach

The main steps of the approach can be summarized as follows (detailed description in the following subsections):

1. *Clustering:* the patterns within the dataset are grouped on the basis of label sharing;
2. *Metric Learning:* the original space is replaced thanks to the learning of a metric capable of bringing together patterns belonging to the same cluster;
3. *Multi-label Classification:* a suitably trained classifier identifies the labels to be associated with each pattern inside the new space.

2.1 Clustering

The idea of exploiting label clustering in this work is motivated by the observation that, in real life scenarios, patterns sharing the same set of labels not always can be considered similar from the point of view of their respective feature descriptors. Labels represent very general high-level concepts assigned by humans, especially in case of multi-label classification; clustering techniques based on data descriptors may work well for specific applications (e.g. image retrieval in [14]), but could present limited generalization capabilities when dealing with classification problems of different nature. Label clustering, on the contrary, allows to group together data with similar labels, even when they are spread over the feature space.

The membership of a generic pattern $\mathbf{x}$ to the available labels can be described through a binary vector $\mathbf{l} = \{l_i, i = 1, .., m\}$ of size m equal to the number of possible labels; each element of the vector $l_i \in \{0, 1\}$, representing the membership of $\mathbf{x}$ to class y_i. This kind of encoding defines a real space of the labels, different from that of the features used to describe the single patterns.

The proposed approach applies a clustering algorithm to the label space, ignoring at all the feature vectors associated to patterns, thus ensuring that clusters are defined according to class membership regardless of the descriptors used to encode them. The metric used for clustering is the *hamming distance*, more suited than the Euclidean distance to deal with the specific label encoding adopted. Clustering is performed by the HDBSCAN [3] technique, a density-based clustering algorithm proposed by the same authors of DBSCAN, of which it represents an evolution. The algorithm identifies clusters as high-density areas in space. The traditional DBSCAN algorithm uses a single density value to identify all clusters, while HDBSCAN extends this concept allowing to identify clusters with different density levels.

The first step of the algorithm requires the estimation of the local density for each pattern $\mathbf{x}_i$. The easiest way to get a density estimate is to evaluate

the neighborhood of a point. If a large number of neighbors are placed in a very narrow radius, that particular area can be considered dense. The distance $d_{core-k}(\mathbf{x}_i)$ is simply defined as the distance between $\mathbf{x}_i$ and its k-th nearest neighbor, using any metric. Based on this measure, the *mutual reachability distance* $d_{reach-k}(\mathbf{x}_i, \mathbf{x}_j)$ between two points $\mathbf{x}_i$ and $\mathbf{x}_j$ is defined as follows:

$$d_{reach-k}(\mathbf{x}_i, \mathbf{x}_j) = \max\{d_{core-k}(\mathbf{x}_i), d_{core-k}(\mathbf{x}_j), d(\mathbf{x}_i, \mathbf{x}_j)\} \tag{1}$$

The practical effect of this metric is to preserve the distances of points laying in dense regions and increasing the reciprocal distances of the points in sparse ones. Clustering is then based on a weighted graph representation of the points in the dataset. The points assume the role of the vertices, while the arc connecting two points $\mathbf{x}_i$ and $\mathbf{x}_j$ is assigned a weight equal to $d_{reach-k}(\mathbf{x}_i, \mathbf{x}_j)$. The idea is to eliminate the weakest arches, thus identifying the high density areas. This can be accomplished extremely efficiently by calculating the minimum spanning tree using a proper algorithm. Given the minimum spanning tree, the next step is to convert that representation into a hierarchy of connected components by grouping neighboring points. DBSCAN cuts the hierarchy tree horizontally by identifying a single threshold used to locate clusters. This represents a strong limitation if there are areas with heterogeneous densities in the dataset. HDBSCAN uses a different approach using the notion of *minimum cluster size*, which is taken as input parameter for the algorithm. Each split in the tree represents a subdivision of the data into multiple clusters. In fact, the root of the tree treats all data as a single cluster. The deeper you go down, the more clusters you divide your data with. Browsing through the tree, HDBSCAN makes an evaluation on the validity of each split present. Only splits into two clusters each at least as large as the minimum cluster size can be considered valid. Otherwise, that split is eliminated by condensing that part of the tree. After walking through the whole hierarchy, we end up with a much smaller condensed tree with a reduced number of nodes. Given the tree, it is only a matter of choosing those nodes/clusters to save.

2.2 Metric Learning

The output of the previous clustering stage is a set of clusters containing data sharing large part of the respective labels. These new labels, found out by a clustering algorithm, are used to learn a new metric in a supervised fashion from the feature space. The metric learning step is aimed at reducing intra-cluster distances while maximizing inter-cluster distances. The literature proposed several supervised metric learning algorithms. The approach adopted in this work is Neighbourhood components analysis (NCA) [7]. It is an algorithm that learns a linear transformation, in a supervised fashion, to improve the classification accuracy of a stochastic nearest neighbors rule in the transformed space. The goal of NCA is to learn an optimal linear transformation matrix A such that the average *leave-one-out (LOO)* classification performance is maximized. It identifies the optimal transformation matrix by maximizing the sum over all samples

of the probability of being correctly classified according to LOO classification. This type of classification tries to predict the class label of a single data point by consensus of its k-nearest neighbours, using a given distance metric. Unfortunately, it is not so simple to identify the optimal matrix as any objective function based on neighborhood points would be not differentiable. In particular, the set of neighbors for a point may undergo discrete changes in response to regular changes in the elements of A. This difficulty is overcome by adopting an approach based on stochastic gradient descent. The entire transformed dataset is considered as stochastic nearest neighbours using a softmax function of the squared euclidean distance between a point and each other point in the space. Considering C_i as the set of points in the same class as sample i, the probability of sample i being correctly classified is:

$$p_i = \sum_{j \in C_i} p_{ij} \quad where \quad p_{ij} = \frac{\exp(-||A\mathbf{x}_i - A\mathbf{x}_j||^2)}{\sum_{z \neq i} \exp -(||A\mathbf{x}_i - A\mathbf{x}_z||^2)}, \quad p_{ii} = 0 \tag{2}$$

At each iteration the algorithm adequately modifies the parameters of matrix A in order to approach patterns belonging to the same class. It is possible to proceed until convergence or until a maximum number of iterations is reached. By limiting the number of iterations, the impact of the algorithm on the new metric can be modulated, preserving part of the old space. NCA also offers the possibility of reducing the dimensionality of the data if deemed excessive.

2.3 Multi-label Classification

The final step of the approach is represented by multi-label classification, which takes place in the new space created by the metric learning algorithm. From the results of this last phase it is possible to verify the effectiveness of the entire proposal, which should lead to an improvement in the classification performance. Among the different approaches in the literature, we decided to adopt in this work two classifiers, Multi-label K-Nearest Neighbors [21] and Random Forest [2], to evaluate the effectiveness of the proposed approach.

Multi-label K-Nearest Neighbors Classifier (ML-kNN). It is a multi-label classification algorithm based on an extension of the well known K-Nearest Neighbor. It assigns labels to a pattern by evaluating the labels of its neighborhood based on a simple approach: it identifies the k closest patterns present in the training set and then uses Bayesian inference to select the labels to assign. Considering the main characteristics of this algorithm, it is essential to have an adequate metric able to aggregate the patterns with the same labels in space.

Random Forest. It is a meta classifier algorithm that fits a number of decision tree classifiers on various sub-samples of the dataset. The final result of the classifier is the average of the results obtained by individual trees. This strategy

allows to increase the classification accuracy and reduce the occurrence of overfitting. The random forest algorithm uses the technique of *bagging*. Every tree is trained selecting a random sample with replacement of the training set. Feature bagging is also used to reduce correlation between trees: during the learning process only a random subset of the available features is used. The use of these techniques leads to better model performance because it decreases the variance of the model, without increasing the bias. While the predictions of a single tree are very sensitive to noise in its training set, the average of many trees is not, as long as the trees are not correlated. It is important to notice that this algorithm is natively multi-label. Given a generic pattern $\mathbf{x}$, each of the trees available will return the predicted class $y \in Y$. The set of these labels represents the output of the entire algorithm. Some more robust implementations only consider those labels voted by a minimal number of classifiers.

3 Experiments and Results

3.1 Evaluation Protocol

To evaluate the effectiveness of this approach, it was chosen to use the main and most known indicators in the field of multi-label classification. They represent a multi-label variant of the most commonly used indicators in classification problems [6]. Let D be a multi-label test set. Let $Y_{\mathbf{x}}$ be the set of labels associated with the generic pattern $\mathbf{x}$ and $Z_{\mathbf{x}}$ those predicted by the classification algorithm. Accuracy, Precision and Recall are three of the most important indicators for evaluating a classification approach. In their multi-label version they are defined as:

$$Accuracy = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_{\mathbf{x}_i} \bigcap Z_{\mathbf{x}_i}|}{|Y_{\mathbf{x}_i} \bigcup Z_{\mathbf{x}_i}|} \tag{3}$$

$$Precision = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_{\mathbf{x}_i} \bigcap Z_{\mathbf{x}_i}|}{|Z_{\mathbf{x}_i}|} \tag{4}$$

$$Recall = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_{\mathbf{x}_i} \bigcap Z_{\mathbf{x}_i}|}{|Y_{\mathbf{x}_i}|} \tag{5}$$

Accuracy measures the overall ability of the classifier of correctly classifying patterns. Precision measures the portion of predicted labels that are correct, while recall measures the portion of real labels that were correctly predicted. Another key indicator within the multi-label classification is the hamming loss. Its definition recalls the concept of hamming distance:

$$HammingLoss = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_{\mathbf{x}_i} \oplus Z_{\mathbf{x}_i}|}{|Y_{\mathbf{x}_i}|} \tag{6}$$

where $\oplus$ is the XOR operator. This indicator is extremely significant in the multi-label context as it summarizes the difference between the real label set and the one predicted by the classifier. Despite the presence of all these specific indicators for multi-label classification, it is still possible to evaluate the quality of the results by counting the percentage of *exact matches*, i.e. the percentage of test data for which all the required labels have been correctly predicted.

3.2 Datasets

Extensive experiments have been carried out to evaluate the effectiveness of our proposal as well as its generality. To this purpose, three heterogeneous multi-label datasets (see Table 1) have been selected:

- *enron* [15], a dataset of emails labelled with a set of categories; each email is encoded using the Bag Of Words encoding;
- *scene* [1], an image dataset; the six labels available identify the characteristics of the landscape depicted in the image itself; each image is described with visual numeric features;
- *bibtex* [10], containing bibliographic data from the BibSonomy social bookmark and publication sharing system, annotated with a subset of the tags assigned by BibSonomy users; it uses the Bag Of Words model.

The three datasets are quite different from different points of view such as content (text/image), features representation, size, number of labels, and how they are distributed. Each dataset assigns a different meaning to the labels (category, description, tag), so that they represent an interesting test bed. Table 1 also reports the training/testing partitioning suggested in the literature for the datasets. Of course clustering, metric learning and classifier training were done on the training set, while the test set was used to measure classification results.

Table 1. The main characteristics of the datasets.

Dataset	Domain	Instances (train/test)	Attributes	Labels	Avg/Max labels per instance
Enron	Text	1702 (1123/579)	1001	53	3.38 / 12
Scene	Image	2407 (1211/1196)	294	6	1.07 / 3
Bibtex	Text	7395 (4880/2515)	1836	159	2.4 / 28

3.3 Results

Before analysing the classification performance, we briefly report the result produced in the clustering phase by the HDBSCAN algorithm. The minimum cluster size required as an input parameter by the algorithm was arbitrarily fixed to 5 for our experiments. Table 2 shows some results related to clustering such as the number of clusters identified in each dataset and their average size in terms of

patterns. The number of clusters obtained is in line with the size and the number of labels in the datasets. It is interesting to note the limited number of clusters identified in the *scene* dataset, quite close to the number of labels. Another important information reported by the table is the average number of clusters in which a single label ends. It should provide an indication of how the patterns presenting that label have been grouped. For example, if a label is present on the majority of the dataset is highly unlikely that all patterns indicating the label end up in a single cluster. Conversely, if a single or a group of labels characterize a certain portion of the dataset, those data will probably shape a single cluster. Of course, the occurrence of these situations is closely related to the meaning that each dataset assigns to its labels. On average, each label in the *enron* and *scene* datasets ends up in a very similar number of clusters (4/3) despite the huge difference in the number of labels available. This is due to the different distribution of the labels within the dataset.

Table 2. The results of clustering.

Dataset	No. of clusters	Cluster average size	Avg No. of Clusters per label
Enron	39	16.38	4.04
Scene	11	89.82	3.00
Bibtex	226	19.11	19.01

The second step of the method is focused on the NCA algorithm for learning a better metric. The maximum number of iterations to be performed during learning has been fixed for all datasets to 30, to avoid excessive flattening of data on clusters, which proved to be counterproductive in terms of results. No dimensionality reduction has been applied in our experiments. As regards the classification, the two algorithms illustrated above were used: ML-kNN and Random Forest. The results obtained by ML-kNN using the proposed approach are compared with the benchmarks indicated in [12], following the same protocol. In particular, the best results in the new space were obtained by assigning the value 10 to the hyper-parameter k. As for Random Forest, it was made a comparison between the original space of the datasets and the built one using a model with 100 decision trees. Thanks to these comparisons, it is possible to evaluate the effectiveness of the proposed method. The classification results, obtained using the indicators shown above, are reported in Table 3.

The results clearly show that the proposed method is able to improve classification performance. In particular, ML-kNN improves significantly, especially for the *bibtex* dataset. Given the characteristics of the classifier, it was reasonable to expect a positive effect from the construction of a space more suitable for classification. The results obtained show the ability of the previously identified clusters to group patterns with the same labels. In general, Random Forest also seems to get benefits, though in a limited form. While on the one hand the increase in performance on *scene* is remarkable, on the other hand the results

Table 3. Classification results on selected datasets.

		Accuracy	Precision	Recall	Hamming D.	Exact match
Enron	ML-kNN (Benchmark)	0.319	0.587	0.358	0.051	0.062
	ML-kNN (Our method)	0.413	0.66	0.465	0.048	0.123
	Random forest (Original Space)	0.401	0.694	0.449	0.047	0.119
	Random forest (Our method)	0.413	0.7	0.456	0.046	0.123
Scene	ML-kNN (Benchmark)	0.629	0.661	0.655	0.099	0.573
	ML-kNN (Our method)	0.698	0.732	0.715	0.090	0.646
	Random forest (Original space)	0.540	0.565	0.540	0.093	0.514
	Random forest (Our method)	0.657	0.688	0.659	0.087	0.624
Bibtex	ML-kNN (Benchmark)	0.129	0.254	0.132	0.014	0.056
	ML-kNN (Our method)	0.257	0.390	0.271	0.013	0.156
	Random forest (Original space)	0.217	0.362	0.219	0.013	0.135
	Random forest (Our method)	0.186	0.316	0.188	0.013	0.114

on the other datasets are not so good. This is not accidental if we consider the characteristics of the two datasets and those of the classifier. On average, *enron* and *bibtex* have a much higher number of labels for each instance than *scene*. The new space, by grouping different patterns, has evidently reduced the ability of the trees to discriminate with respect to a single label, causing not exactly exciting results. In summary, contrary to what happened with ML-kNN, the new learned metric hindered the task of the Random Forest classifier.

Overall, the results obtained are very good and prove the capability of the proposed method to build a more effective space for multi-label classification.

4 Conclusions and Future Works

In this work a metric-learning approach aimed at improving the performance of a multi-label classifier has been presented. In particular, the use of a cluster algorithm applied to pattern labels allowed to identify groups of data that share most of their labels, subsequently used to build a new, more effective, representation space. The results obtained on three different multi-label datasets show how the new space built by the proposed approach is able to significantly increase the classification results. Future researches will be devoted to investigate the scalability of this approach to larger datasets. Another future development involves the introduction, in the metric learning step, of a neural network capable of building a more suitable space for classifiers.

References

1. Boutell, M.R., Luo, J., Shen, X., Brown, C.M.: Learning multi-label scene classification. Pattern Recogn. **37**(9), 1757–1771 (2004)
2. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)

3. Campello, R.J.G.B., Moulavi, D., Sander, J.: Density-based clustering based on hierarchical density estimates. In: Pei, J., Tseng, V.S., Cao, L., Motoda, H., Xu, G. (eds.) PAKDD 2013. LNCS (LNAI), vol. 7819, pp. 160–172. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37456-2_14
4. Cherman, E.A., Monard, M.C., Metz, J.: Multi-label problem transformation methods: a case study. CLEI Electron. J. **14**(1), 4–4 (2011)
5. Ganda, D., Buch, R.: A survey on multi label classification. Recent Trends Program. Lang. **5**(1), 19–23 (2018)
6. Godbole, S., Sarawagi, S.: Discriminative methods for multi-labeled classification. In: Dai, H., Srikant, R., Zhang, C. (eds.) PAKDD 2004. LNCS (LNAI), vol. 3056, pp. 22–30. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24775-3_5
7. Goldberger, J., Hinton, G.E., Roweis, S.T., Salakhutdinov, R.R.: Neighbourhood components analysis. Adv. Neural Info. Process. Syst. **17**, 513–520 (2005)
8. Gouk, H., Pfahringer, B., Cree, M.: Learning distance metrics for multi-label classification. In: Asian Conference on Machine Learning, pp. 318–333. PMLR (2016)
9. Gupta, P., Anand, A.: Multi label classification using label clustering. In: Appearing in Proceedings of the 1st Indian Workshop on Machine Learning, IIT Kanpur, India (2013)
10. Katakis, I., Tsoumakas, G., Vlahavas, I.: Multilabel text classification for automated tag suggestion. In: Proceedings of the ECML/PKDD, vol. 18, p. 5 (2008)
11. Liu, W., Tsang, I.W.: Large margin metric learning for multi-label prediction. In: Proceedings of the National Conference on Artificial Intelligence (2015)
12. Madjarov, G., Kocev, D., Gjorgjevikj, D., Džeroski, S.: An extensive experimental comparison of methods for multi-label learning. Pattern Recogn. **45**(9), 3084–3104 (2012)
13. Nam, J., Kim, J., Loza Mencía, E., Gurevych, I., Fürnkranz, J.: Large-scale multi-label text classification — revisiting neural networks. In: Calders, T., Esposito, F., Hüllermeier, E., Meo, R. (eds.) ECML PKDD 2014. LNCS (LNAI), vol. 8725, pp. 437–452. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44851-9_28
14. Nasierding, G., Tsoumakas, G., Kouzani, A.Z.: Clustering based multi-label classification for image annotation and retrieval. In: IEEE SMC, pp. 4514–4519 (2009)
15. Read, J., Pfahringer, B., Holmes, G.: Multi-label classification using ensembles of pruned sets. In: IEEE ICDM, pp. 995–1000. IEEE (2008)
16. Sugiyama, M.: Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis. J. Mach. Learn. Res. **8**, 1027–1061 (2007)
17. Tsoumakas, G., Vlahavas, I.: Random k-labelsets: an ensemble method for multilabel classification. In: Kok, J.N., Koronacki, J., Mantaras, R.L., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 406–417. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74958-5_38
18. Wang, J., Yang, Y., Mao, J., Huang, Z., Huang, C., Xu, W.: CNN-RNN: a unified framework for multi-label image classification. In: IEEE CVPR (2016)
19. Weinberger, K.Q., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. J. Mach. Learn. Res. **10**(2), 1–38 (2009)
20. Yu, Z., Wang, Q., Fan, Y., Dai, H., Qiu, M.: An improved classifier chain algorithm for multi-label classification of big data analysis. In: 2015 IEEE 17th HPCC, 2015 IEEE 7th CSS, and 2015 IEEE 12th ICESS, pp. 1298–1301 (2015)
21. Zhang, M.L., Zhou, Z.H.: ML-KNN: a lazy learning approach to multi-label learning. Pattern Recogn. **40**(7), 2038–2048 (2007)