Multi-Label Classification using Sticky Brownian Motion

Sepehr Abbasi-Zadeh
University of Toronto
sepehr@cs.toronto.edu

Christodoulos Karavasilis
University of Toronto
ckar@cs.toronto.edu

Abstract

Multi-label learning and classification deals with problems where a single instance can be associated with multiple class labels. Binary classifiers are often used to predict each label separately but can be misled by not exploiting potential dependencies between different labels. A common research theme in this area focuses on learning these label dependencies in order to improve performance [27]. In this work, we propose a new architecture based on Brownian motion [13] that uses binary classifiers in addition to pair-wise correlations between the labels, and produces a label set that adheres to these dependencies. We provide experimental results that show our architecture is comparable and sometimes outperforms several state-of-the-art methods on real-world datasets and we identify directions for possible improvement.

1 Introduction

Multi-label classification (MLC) is a type of a structured prediction problem [3] that has attracted a lot of attention and has applications spanning information retrieval [20], bioinformatics [4], content annotation [16] and web mining [22]. The goal is to learn a classifier that annotates instances with all the relevant labels. More formally, we have an input domain $\mathcal{X} = \mathbb{R}^d$, an output domain $\mathcal{Y} = \{y_1, .., y_n\}$ where $\forall i \in [n],\ y_i \in \{0, 1\}$, and a score function $\mathcal{S} : (\mathcal{X}, \{0, 1\}^n) \to \mathbb{R}$. Given an instance $x \in \mathcal{X}$, the goal is to find $argmax_{y \in \{0,1\}^n} \mathcal{S}(x, y)$. The solution space is large enough for computing a brute-force solution to be infeasible. Many different approaches have been proposed to tackle this problem and they can be divided ([23, 24]) into two main categories: a) *problem transformation methods* that find a solution using one or more binary classifiers, and b) *algorithm adaptation methods* that deal with the data in its entirety to predict multiple labels. A third category of algorithms, called *ensemble methods* [12], was later identified as a general approach to performing MLC, and it works by combining a number of multi-label classifiers. Recent work comparing state-of-the-art ensembles of multi-label classifiers can be found in [14].

It is generally accepted that to achieve strong generalization in MLC models, one should exploit potential correlations between the labels [28, 26, 24, 9]. Given n binary classifiers that output the probability of the corresponding label being true (e.g. logistic regression), and a correlation matrix $W \in \mathbb{R}^{n \times n}$ that models pair-wise dependencies between the labels, we introduce a technique which we call Brownian Inference (BI) that predicts the final labels while maintaining these dependencies. The core idea is using sticky brownian motion (SBM) [1] as a randomized rounding method that works by performing a correlated random walk in an $n-$dimensional space.

To get an intuition, assume we have the three following labels: golf_pitch, football_field, and golf_flag. We now want to annotate the image of the golf pitch shown in Figure 1. Let's assume our binary classifiers returned the following probabilities for each label: $P[golf\_pitch] = 0.6$, $P[football\_field] = 0.4$, $P[golf\_flag] = 0.85$. The model is slightly agnostic between golf pitch

Learning to Search (LTS 2019), Toronto, Canada.

Figure 1: Image of a golf pitch.

Figure 2: Image of a football field.

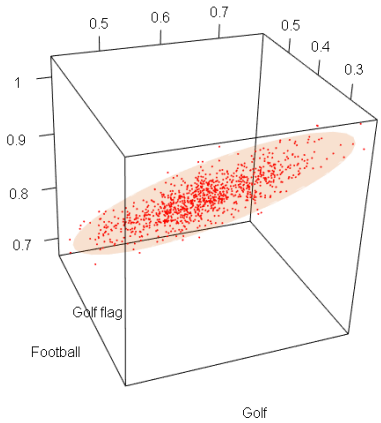

Figure 3: Correlations between three labels.

and football field as they can look quite similar. Additionally, assume we have access to a correlation matrix (learned by an independent process) with the following entries:

- corr(golf_pitch, football_field) = -0.97
- corr(golf_pitch, golf_flag) = 0.89
- corr(football_field, golf_flag) = -0.92

A visualization of the correlation matrix can be found in Figure 3. Guided by these dependencies, our method is unlikely to return a vector containing both gold_pitch and football_field. Intuitively, this is due to the fact that in each step it uniformly samples a point on the surface of the shown ellipsoid shown as the correlation matrix of Figure 3. Therefore, the motion is more likely to go toward the directions that have stronger dependencies. Running BI on this toy example we actually return the correct vector 70% of the time and never do we get those two labels together. Additionally, using the randomized rounding method of including a label with its corresponding probability (e.g. the label golf_pitch is added with probability 0.6), returns the correct vector 30% of the time. This example is an easy way to see why taking advantage of label correlations can vastly improve prediction accuracy.

Our architecture is explained in detail in section 3. The correlation matrix W can be heuristically computed (e.g. using the Pearson correlation coefficient between each two label vectors) or learned using stochastic gradient descent. We have experimented with both and, as shown in section 4, the latter performs much better but with an additional overhead on running time. We showcase the advantage of BI that uses label correlations over independent binary classification and also compare it against various state-of-the-art methods on real datasets.

2 Related work

Binary relevance [6] is the approach of using a binary classifier independently for each label. This is the most common problem transformation method that has attracted a lot of attention [27] because of its simplicity and elegance. In order to utilize correlations among the labels, many augmentations have been proposed (see [27] and references therein). A different family of methods [18, 25] considers the label powerset. A new single-label dataset is generated with a different class for every combination of labels. This approach effectively considers label correlations but the issue is that the the complexity is exponential in the number of labels. A popular approach that considers label dependencies is that of using *classifier chains* [19]. Assuming n labels, a chain of n binary datasets is generated, with the feature space of each classifier being augmented with the label predictions of previous classifiers. The chain-induced label ordering has a direct impact on performance and different chain ordering techniques have been considered in the literature [10], with ensembles of classifier chains trying to overcome the problem of choosing the right chain ordering altogether. In contrast, we aim to capture label correlations in a more intuitive and abstract way, by having our correlation matrix be independent of the core binary learning process, and making it easier to test and search for the underlying dependencies. There have also been efforts to create techniques (e.g. by pruning infrequently occurring labels) for scalable MLC [17, 8]. The special problem of *extreme multi-label classification* [2] deals with cases where the number of labels is extremely large but only a small subset is relevant for every instance.

3 Model description

Brownian motion [13] in an n-dimensional domain U is defined by an $n \times n$ correlation matrix W and a starting point $Z(0) \in U$.
The diffusion process proceeds as follows:

$$\begin{aligned} &\text{while } Z(t) \notin \partial U : \\ &\quad Z(t+1) = Z(t) + \delta W^{1/2} B_t \qquad \text{where } B_t \sim \mathcal{N}(0, I_n) \\ &\quad t = t + 1. \end{aligned} \tag{1}$$

Here $\mathcal{N}(0, I_n)$ is a multivariate (n-dimensional) Gaussian vector, and $\delta > 0$ is an infinitesimal step size. In our problem, we assume that the set of all possible labels is of size n. Accordingly, we assume $U = [0, 1]^n$ in the Brownian motion description, in which each vertex of U corresponds to a possible vector of labels. That is, since by running the sticky[1] Brownian procedure it is guaranteed that the final solution is a vertex $\mathbf{v}$ of U, we set the $i-$th label to 1 if $\mathbf{v_i} = \mathbf{1}$, and 0 otherwise. Further, for solving the inference problem on a given instance x, the starting point $Z(0)$ is given to us as follows: $Z(0)_i = \mathbb{P}[\ell_i = 1|x] \quad \forall i \in [n]$. Therefore, for an instance x, to solve the inference problem 1, we just need to obtain $Z(0)$ from n different (possibly uncorrelated) learners, and also train a correlation matrix W based on a loss function $L(y, \hat{y}_W(x))$, where $y, \hat{y}_W(x) \in \{0, 1\}^n$. We do not impose any constraint on the loss function, but we assume that for any pair of $(y, \hat{y}_W(x)) \in \{0, 1\}^n \times \{0, 1\}^n$ we can compute the loss. An overview of the model is presented in Figure 4.

The following theorem relates the expected loss that we get by running the sticky Brownian motion to the correlation matrix W.

Theorem 1. *For a given instance $s \in \mathcal{X}$, let $\hat{y}_W(s)$ be the output of running the sticky Brownian motion with correlation matrix W, starting from point $Z(0) \in [0, 1]^n$. Also, let $u_W : [0, 1]^n \to \mathbb{R}$ be the (unique) solution to the following boundary value partial differential equation:*

$$\begin{aligned} \sum_{i,j \in [n]} w_{ij} \frac{\partial^2 u_W}{\partial x_i \partial x_j} &= 0 \\ s.t. \quad u_W(a) &= L_{ext}(a) \quad \forall a \in \partial [0, 1]^n, \end{aligned}$$

where L_{ext} is a unique extension of the loss function L on the boundaries of the cube U yet to be defined. Then we have:

$$\mathbb{E}_{Z(0)} [L(y, \hat{y}_W(s))] = u_W (Z(0)) .$$

[1] It's called sticky because as soon as we hit a surface, it is fixed and we continue the movement in a lower dimension.

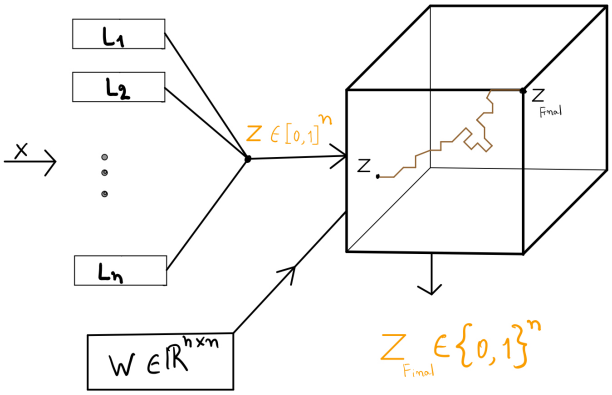

Figure 4: Model overview.

The extended function L_{ext} is defined recursively as the solution to the same PDE projected to the corresponding spanned subspace.

It turns out that the solution to the previous PDE is unique and it is a harmonic function [13]. Therefore, it is infinitely differentiable. Thus, Theorem 1 gives us enough information for updating the correlation matrix W which minimizes the expected loss L using a simple stochastic gradient descent method which have access to the gradients of the form $\frac{\partial L}{\partial W}$. However, since we do not know the closed-form solution to the mentioned PDE, it is not possible to find the gradient analytically. Therefore, we use a Monte Carlo sampling method to find an unbiased gradient estimator numerically, and note that this method does not require to find the extended function L_{ext}. To make sure that the correlation matrix W remains positive definite, we assign each label a k-dimensional vector, and we learn the k-dimensional representation of that label. This representation in turn gives us the correlation between each two labels by taking the dot product between their corresponding vectors and normalizing it. This way, the resulting correlation matrix by the definition remains a positive definite matrix.

4 Experiments

Our evaluation metrics are Hamming distance and Accuracy. Hamming distance is defined as the average symmetric difference between predicted and true vectors normalized by the total number of labels. Accuracy is defined as the average ratio of correctly predicted labels over the number of true labels in addition to wrongly predicted labels. More formally, let Y be the true labels, Z the predicted labels, and Δ be the binary operator computing the symmetric difference between two sets. Hamming distance is given by $\frac{1}{m_{test}} \sum_{1=i}^{m_{test}} \frac{1}{q} |Z_i \Delta Y_i|$, and Accuracy is given by $\frac{1}{m_{test}} \sum_{1=i}^{m_{test}} \frac{|Z_i \cap Y_i|}{|Z_i \cup Y_i|}$.

The datasets[2] we used along with some statistics can be found in table 1. The total number of instances is m, number of features is d, and number of labels is n. The density ($dens$) is defined as the mean number of relevant labels for each example divided by the total number of labels, and diversity (div) is the ratio of distinct labelsets that appear over the total number of possible labelsets. These datasets contain a pre-defined division between train and test data, making it easier to compare our results against those reported in other papers.

We compare our model against four state-of-the-art methods that, as reported in [14], seem to achieve the best performance over many datasets when using the Hamming distance and Accuracy metrics. The *Ensemble of Binary Relevance classifiers* (EBR) [19] is generated using bagging [7] of binary classifiers, each trained on a random subset of the instances. The *Ensemble of Classifier Chains* (ECC) [19] consists of multiple classifier chains, each using a random chain and trained on randomly selected instances. The final predictions are obtained by averaging the confidence values for each

[2]Detailed description and origin of datasets can be found in http://www.uco.es/kdis/mllresources

label. The AdaBoost.MH method [21] is based on the AdaBoost algorithm and works by maintaining different sets of weights over the instances and the labels. The *Random Forest of Predictive Clustering Trees* (RF-PCT) [11] method uses predictive clustering trees [5] as base classifiers trained on instances sampled by bagging.

The Random method corresponds to the independent inclusion of a label randomly with probability proportional to the confidence returned by the corresponding binary classifier. BI_{heur} corresponds to Brownian inference with the correlation matrix being heuristically computed using the Pearson correlation coefficient between each two label vectors in the training set. BI_{train} uses a correlation matrix initialized heuristically but then improved over many iterations using stochastic gradient descent.

For the BI_{train} and BI_{heur}, we implemented the Walk-on-Spheres algorithm [15] which is a faster simulation of the Brownian motion. For the binary classifiers, we have used logistic regression learners.

The reported results for BI_{train} are not as good as BI_{heur} for the CHD_49 dataset, and this is due to the fact that the experiments did not converge on our laptops before the deadline. We used a quad-core core i7 2.7GHz laptop with 16GB of RAM for doing these experiments. Time constraints were also the reason we didn't include additional datasets.

Table 1: Datasets

Dataset	Domain	m	d	n	$dens$	div
CHD_49	Medicine	555	49	6	0.430	0.531
EMOTIONS	Music	593	72	6	0.311	0.422

Table 2: Hamming Distance Results (lower is better)

Dataset	EBR	ECC	AdaB.MH	RF-PCT	BI_{heur}	BI_{train}	Random
CHD_49	0.301	0.295	0.307	0.312	0.345	0.351	0.355
EMOTIONS	0.202	0.204	0.302	0.209	0.316	0.291	0.320

Table 3: Accuracy Results (higher is better)

Dataset	EBR	ECC	AdaB.MH	RF-PCT	BI_{heur}	BI_{train}	Random
CHD_49	0.513	0.540	0.464	0.533	0.521	0.513	0.504
EMOTIONS	0.513	0.539	0.045	0.548	0.553	0.572	0.546

5 Conclusion

We have verified that using label correlations in addition to binary classifiers can greatly increase prediction accuracy. We have introduced a new architecture that uses underlying label dependencies in addition to binary learners and predicts the final labels while maintaining those connections. The way we model these dependencies (i.e. pair-wise correlation matrix) is independent of the binary learning process and this makes it easier to experiment, exploit, and further refine the correlations in a way that increases the accuracy of our model. Being able to extract these correlations using methods of varying complexity (e.g. simple heuristics or learned over time), we have the additional flexibility of choosing different methods to balance the performance trade-off between different parts of our architecture (binary learning, correlation computation and brownian rounding). By conducting experiments, we have shown that our approach is comparable and in fact sometimes outperforms existing state-of-the-art methods in standard real-world datasets.

One disadvantage of our method is its seeming difficulty to scale. The most promising results are given by BI_{train} that uses a learned correlation matrix. The cost to learn and improve this matrix might be too high for certain applications and even though a more specialized implementation would surely help, it is not clear if it would affect overall scalability. Estimating the gradient (as discussed in section 3) in an efficient way is a promising direction to making our method more scalable. The issue of scalability is also amplified if we consider higher order correlations (instead of just pair-wise)

between variables. Even though our model in theory can support correlations over multiple variables, it is not clear how feasible it is in practice to consider more complicated dependencies.

Future work could be experimenting with different ways (both heuristic and learned) of computing the correlation matrix and better understanding the trade-offs between correlation quality and overall performance, both in terms of predictive accuracy and efficiency. We would also like to test more specialized implementations to better compare the running time and scalability of our approach compared to state-of-the-art methods.

References

[1] S. Abbasi-Zadeh, N. Bansal, G. Guruganesh, A. Nikolov, R. Schwartz, and M. Singh. Sticky brownian rounding and its applications to constraint satisfaction problems. *arXiv preprint arXiv:1812.07769*, 2018.

[2] R. Babbar and B. Schölkopf. Data scarcity, robustness and extreme multi-label classification. *Machine Learning*, pages 1–23, 2019.

[3] G. BakIr, T. Hofmann, B. Schölkopf, A. J. Smola, and B. Taskar. *Predicting structured data*. MIT press, 2007.

[4] Z. Barutcuoglu, R. E. Schapire, and O. G. Troyanskaya. Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7):830–836, 2006.

[5] H. Blockeel, L. De Raedt, and J. Ramon. Top-down induction of clustering trees. *arXiv preprint cs/0011032*, 2000.

[6] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown. Learning multi-label scene classification. *Pattern recognition*, 37(9):1757–1771, 2004.

[7] L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[8] J. Deng, O. Russakovsky, J. Krause, M. S. Bernstein, A. Berg, and L. Fei-Fei. Scalable multi-label annotation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3099–3102. ACM, 2014.

[9] E. Gibaja and S. Ventura. A tutorial on multilabel learning. *ACM Computing Surveys (CSUR)*, 47(3):52, 2015.

[10] E. C. Goncalves, A. Plastino, and A. A. Freitas. A genetic algorithm for optimizing the label ordering in multi-label classifier chains. In *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*, pages 469–476. IEEE, 2013.

[11] D. Kocev, C. Vens, J. Struyf, and S. Džeroski. Ensembles of multi-objective decision trees. In *European conference on machine learning*, pages 624–631. Springer, 2007.

[12] G. Madjarov, D. Kocev, D. Gjorgjevikj, and S. Džeroski. An extensive experimental comparison of methods for multi-label learning. *Pattern recognition*, 45(9):3084–3104, 2012.

[13] P. Mörters and Y. Peres. *Brownian motion*, volume 30. Cambridge University Press, 2010.

[14] J. M. Moyano, E. L. Gibaja, K. J. Cios, and S. Ventura. Review of ensembles of multi-label classifiers: models, experimental study and prospects. *Information Fusion*, 44:33–45, 2018.

[15] M. E. Muller et al. Some continuous monte carlo methods for the dirichlet problem. *The Annals of Mathematical Statistics*, 27(3):569–589, 1956.

[16] G.-J. Qi, X.-S. Hua, Y. Rui, J. Tang, T. Mei, and H.-J. Zhang. Correlative multi-label video annotation. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 17–26. ACM, 2007.

[17] J. Read. *Scalable multi-label classification*. PhD thesis, University of Waikato, 2010.

[18] J. Read, B. Pfahringer, and G. Holmes. Multi-label classification using ensembles of pruned sets. In *8th IEEE international conference on data mining*, pages 995–1000. IEEE, 2008.

[19] J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains for multi-label classification. *Machine learning*, 85(3):333, 2011.

[20] C. Sanden and J. Z. Zhang. Enhancing multi-label music genre classification through ensemble techniques. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 705–714. ACM, 2011.

[21] R. E. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. *Machine learning*, 39(2-3):135–168, 2000.

[22] L. Tang, S. Rajan, and V. K. Narayanan. Large scale multi-label classification via metalabeler. In *Proceedings of the 18th international conference on World wide web*, pages 211–220. ACM, 2009.

[23] G. Tsoumakas and I. Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1–13, 2007.

[24] G. Tsoumakas, I. Katakis, and I. Vlahavas. Mining multi-label data. In *Data mining and knowledge discovery handbook*, pages 667–685. Springer, 2009.

[25] G. Tsoumakas, I. Katakis, and I. Vlahavas. Random k-labelsets for multilabel classification. *IEEE Transactions on Knowledge and Data Engineering*, 23(7):1079–1089, 2010.

[26] J. Zhang, X. Wu, and V. S. Sheng. Learning from crowdsourced labeled data: a survey. *Artificial Intelligence Review*, 46(4):543–576, 2016.

[27] M.-L. Zhang, Y.-K. Li, X.-Y. Liu, and X. Geng. Binary relevance for multi-label learning: an overview. *Frontiers of Computer Science*, 12(2):191–202, 2018.

[28] M.-L. Zhang and Z.-H. Zhou. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837, 2013.