UMA ESTRATÉGIA HÍBRIDA PARA O PROBLEMA DE CLASSIFICAÇÃO MULTIRRÓTULO

Tiago Amador Coelho[1], **Ahmed Ali Abdalla Esmin**[2], **Wagner Meira Júnior**[1]

[1]Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte, MG – Brasil

[2]Universidade Federal de Lavras (UFLA)
Lavras, MG – Brasil

{tiago,meira}@dcc.ufmg.br, ahmed@dcc.ufla.br

Abstract. *Multi-label classification learning first arose in the context of text categorization, where each document may belong to several classes simultaneously and has attracted significant attention lately, as a consequence of both the challenge it represents and its relevance in terms of application scenarios. In this paper, we propose a new hybrid approach, Multi Label K-Nearest Michigan Particle Swarm Optimization (ML-KMPSO). It is based on two strategies. The first strategy is the Michigan Particle Swarm Optimization (MPSO), which breaks the multi-label classification task into several binary classification problems, but it does not take into account the correlations among the various classes. The second strategy is ML-KNN, which is complementary and takes into account the correlations among classes. We evaluated the performance of ML-KMPSO using two real-world datasets and the results show that our proposal matches or outperforms well-established multi-label classification learning algorithms.*

Resumo. *A aprendizagem de classificadores multirrótulo se originou na categorização de textos, onde cada documento podia pertencer a mais de uma classe simultaneamente e tem atraído muita atenção como consequência tanto do desafio de pesquisa que representa quanto da relevância dos seus cenários de aplicação. Neste artigo, propomos uma nova abordagem híbrida, Multi Label k-Nearest Michigan Particle Swarm Optimization (ML-KMPSO), que é baseada em duas estratégias. A primeira é a divisão do problema multirrótulo em diversos problemas binários, para tal foi utilizado o Michigan Particle Swarm Optimization (MPSO) para resolve-los, mais esta estratégia não leva em consideração as correlações existentes entre as classes. Já a segunda estratégia tem como objetivo considerar as correlações existentes entre as classes utilizando o Multi Label K-Nearest Neighbor (ML-KNN). Avaliamos o desempenho do ML-KMPSO utilizando duas bases de dados reais e os resultados mostram que a nossa proposta se iguala ou supera algoritmos estado-da-arte de classificação multirrótulo.*

1. Introdução

Aplicações reais como classificação semântica de cenas, classificação funcional de proteínas, categorização de textos, categorização de músicas, entre outras, vem reque-

rendo cada vez mais métodos que realizem a classificação multirrótulo, uma vez que cada instância a ser classificada nesses cenários é associada a mais de uma classe.

No problema de classificação unirrótulo, que é o mais tradicional, cada instância é associada a um sub-conjunto de rótulos unitário Y de um conjunto de rótulos $\mathcal{Y}$, $|\mathcal{Y}| > 1$. Quando $|\mathcal{Y}| = 2$, o problema é chamado de problema de classificação binária, e se $|\mathcal{Y}| > 2$ o problema é chamado de problema de classificação multiclasse. Entretanto há diversos problemas nos quais as instâncias podem estar associadas a mais de uma classe simultaneamente, $|Y| \geq 2$, em que as classes não são disjuntas, esses problemas são conhecidos como problemas de classificação multirrótulo (MLC). Essa generalização do problema de classificação o torna mais difícil de se resolver.

Uma estratégia intuitiva de se resolver esse problema é a decomposição do problema em múltiplos e independentes problemas de classificação binários, o que não considera as correlações que podem existir entre as diferentes classes, tornando-o ineficiente. Uma outra forma de se resolver esse problema é adaptar um algoritmo de classificação unirrótulo para realizar a classificação multirrótulo, como categorização de texto multirrótulo [Schapire and Singer 2000], árvores de decisão multirrótulo [Quinlan 1993], métodos de kernel multirrótulo [Elisseeff and Weston 2001], redes neurais multirrótulos [Zhang and Zhou 2006] e ML-KNN [Zhang and Zhou 2005]. Nesses casos, os resultados obtidos ainda não são muito bons e por isso essa área continua em aberto e atraindo muita atenção de pesquisadores.

Neste trabalho é proposta uma nova estratégia híbrida, ML-KMPSO, que se baseia em duas estratégias existentes. A primeira é o Michigan Particle Swarm Optimization (MPSO), que divide o problema de classificação multirrótulo em diversos problemas de classificação binário e os resolve, mas não considera as possíveis correlações existentes entre as classes. A segunda estratégia é o Multi-Label K-Nearest Neighbor (ML-KNN), que é complementar à primeira e considera a correlação existente entre as classes.

Na prática, o ML-KMPSO funciona da seguinte forma. Inicialmente, para cada instância de treinamento, identifica as instâncias vizinhas mais próximas do conjunto de treinamento e calcula a sua probabilidade *a priori*. Então, um modelo de classificação é gerado pelo MPSO e um conjunto de partículas especializadas em classificação é identificado. Baseado nas informações estatísticas obtidas do conjunto de rótulos dos vizinhos das instâncias, incluindo as partículas de classificação geradas pelo MPSO, é utilizado então o princípio do *maximun a posteriori* (MAP) [Zhang and Zhou 2007] para se determinar o conjunto de rótulos das instâncias de teste (resultado da classificação).

O trabalho está organizado da seguinte forma. A Seção 2 discute o problema de classificação multirrótulo. Na Seção 3 é feito um resumo do *Particle Swarm Optimization* ou Método de Enxame de Partículas (PSO). A Seção 4 apresenta o PSO e o MPSO (*Michigan Particle Swarm Optimization*) para o problema de classificação. Na Seção 5 são apresentados e discutidos os experimentos. Finalmente na Seção 6 são colocadas as conclusões e os trabalhos futuros.

2. Classificação Multirrótulo

Pesquisas em classificação multirrótulo foram inicialmente motivadas pelo desafio representado na categorização de textos, onde vários rótulos podem ser associados a um mesmo documento [Zhang and Zhou 2007].

No problema de classificação multirrótulo, cada instância está associada a um conjunto de rótulos Y, onde Y $\subseteq \mathcal{Y}$, sendo $\mathcal{Y}$ um conjunto de todos os rótulos do problema.

Em [Li et al. 2006], o problema de classificação multirrótulo é definido da seguinte forma: Sendo $\mathcal{X}$ um conjunto de treinamento, $\mathcal{Y} = \{1, 2, ..., k\}$ o conjunto de rótulos. Dado um conjunto de treinamento da forma $\langle x_i, Y_i \rangle, x_i \in \mathcal{X}, Y_i \in 2^{|\mathcal{Y}|}$, onde $2^{|\mathcal{Y}|}$ são todas as combinações possíveis em $\mathcal{Y}$. O objetivo é aproximar uma função $f(x)$, tal que $f(x)$ retorne valores de $2^{|\mathcal{Y}|}$ com o menor erro. A dificuldade de definir o erro em multirrótulo é que diversas combinações de rótulos são possíveis.

Na maioria dos casos, a abordagem multirrótulo induz uma ordenação dos possíveis rótulos de uma dada instância de acordo com $f(x, l_n)$. Então, formalmente, podemos definir $rank_f(x, l)$ como $rank_f$ do rótulo l para a instância x, e se $f(x, l_1) \leq f(x, l_2)$ então $rank_f(x, l_1) \leq rank_f(x, l_2)$.

Tanto [Trohidis et al. 2008] como [Zhang and Zhou 2007], em seus trabalhos, agrupam em duas categorias os métodos para resolver o problema multirrótulo:

- a primeira categoria decompõe o problema em múltiplos e independentes problemas de classificação binária ou unirrótulo, mas não leva em consideração as correlações existentes entre as classes;
- a segunda categoria é a realização de alterações nos algoritmos de classificação existentes para permitir a classificação multirrótulo.

Para ilustrar os diferentes métodos que são encontrados na literatura para o problema múltirrótulo, [de Carvalho and Freitas 2009] organizaram de maneira hierárquica todos esses métodos, como mostrado na Figura 1.

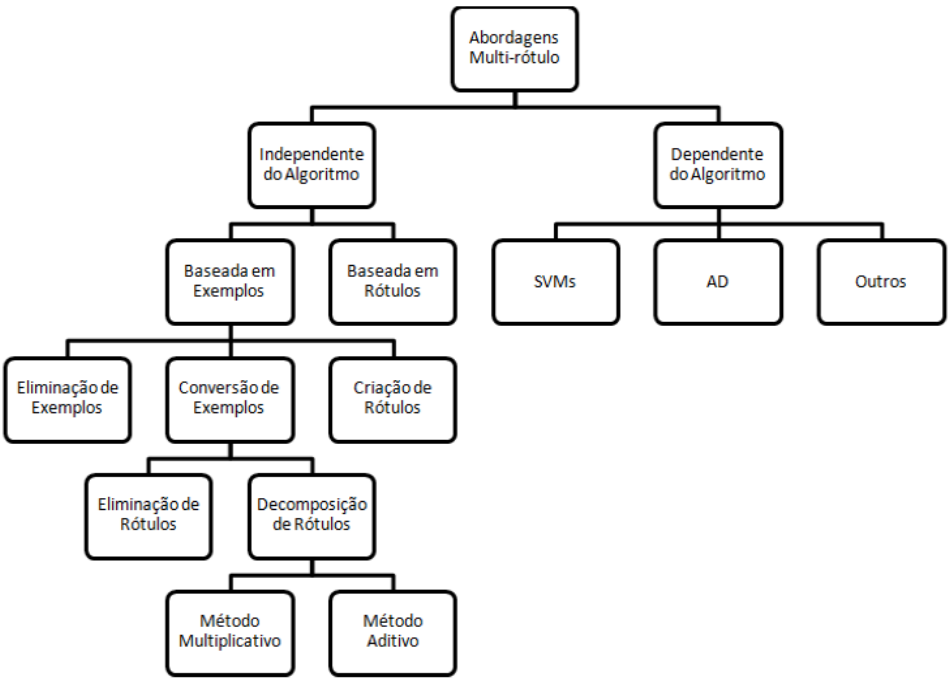

Figura 1. Técnicas para Classificação Multirrótulo (adaptado de [de Carvalho and Freitas 2009])

Um dos métodos estado da arte para classificação multirrótulo é o BoosTexter. Proposto por [Schapire and Singer 2000], o método mantém um conjunto de pesos para cada par exemplo-rótulo do conjunto de treinamento. Os pares que são difíceis de se prever corretamente têm os seus pesos incrementados e, para os pares facilmente previstos, os seus pesos são decrementados.

[Elisseeff and Weston 2001] propuseram o RANK-SVM, um método baseado em *kernels* para classificação, que maximiza a soma das margens de todas as categorias simultaneamente, ordenando as categorias relevantes de cada instância da base de treinamento e penalizando as irrelevantes.

No trabalho de [Zhang and Zhou 2007] foi proposto o método Multi Label K-Nearest Neighbors (ML-KNN), derivado do método KNN tradicional e diferenciado pelo uso de probabilidades *a priori* e *a posteriori*. Inicialmente o método identifica, para cada instância na base de treinamento, os seus k vizinhos mais próximos. Então, baseado na informação estatística *a priori* obtida do conjunto de rótulos destes vizinhos, é utilizado o princípio *Maximum a Posteriori* (MAP) para determinar o conjunto de rótulos da instância de teste.

Neste trabalho propomos uma nova abordagem híbrida para resolver o problema de classificação multirrótulo, que reúne a abordagem independente do algoritmo (MPSO) com a abordagem dependente do algoritmo (ML-KNN), como ilustrado na Figura 2.

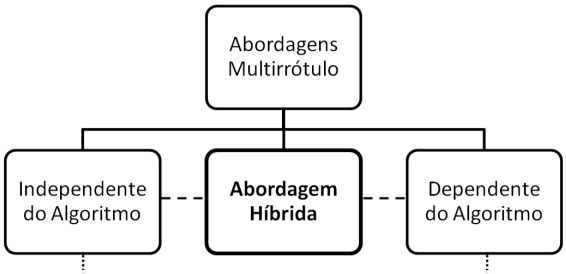

Figura 2. Nova organização dos métodos para resolver o problema multirrótulo

3. Método de Enxame de Partícula - PSO

O *Particle Swarm Optimization* (PSO) ou Método de Enxame de Partícula é um método de otimização baseado em comportamento social de um bando de pássaros, proposto por [Kennedy and Eberhart 1995]. O método PSO encontra a solução ótima do problema usando uma população de partículas, onde cada partícula é uma solução candidata do problema. Basicamente o PSO simula um bando de pássaros em um espaço multidimensional ([Kennedy and Eberhart 1995, Eberhart and Kennedy 1995]). Podemos definir o PSO como:

- Cada partícula i tem as seguintes propriedades: Uma posição atual no espaço de busca, x_i, uma velocidade atual, v_i, e uma melhor posição individual, y_i.
- A melhor posição individual, y_i, corresponde à posição no espaço de busca onde a partícula i apresenta o menor erro determinado pela função f.

- A melhor posição global é ilustrada por $\breve{y}$, representando a posição que teve o menor erro dentre todas as y_i.

Durante cada iteração do algoritmo, cada partícula no enxame é atualizada conforme as equações 1 e 2. A atualização da velocidade é dada por:

$$\begin{aligned} v_{i,j}(t+1) \quad &= wv_{i,j}(t) + c_1 r_{1,j}(t)[y_{i,j}(t) - x_{i,j}(t)] + \\ &\quad c_2 r_{2,j}(t)[\breve{y}_j(t) - x_{i,j}(t)] \end{aligned} \tag{1}$$

onde c_1 and c_2 são duas constantes positivas, r_1 e r_2 são variáveis aleatórias dentro do intervalo [0,l], e w é o valor da inércia.

A atual posição da partícula é atualizada com a equação:

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{2}$$

A equação 1 consiste em três partes. A primeira parte é a velocidade atual da partícula, o que demonstra seu estado atual; a segunda parte é o termo de cognição, que expressa o "conhecimento ou aprendizado"da própria partícula; a terceira parte é chamada de fator social, e reflete as informações compartilhadas no enxame. Estas três partes em conjunto, determinam a habilidade da partícula no espaço de busca, e sob a influência delas, as partículas podem atingir uma posição melhor e mais eficaz.

As equações 3 e 4 definem como os melhores valores individual e global são atualizados no instante t, respectivamente. É assumido que o enxame é formado por s partículas (onde $i \in 1..s$).

$$y_i(t+1) = \begin{cases} y_i(t) & se \quad f(y_i(t)) \leq f(x_i(t+1)) \\ x_i(t+1) & se \quad f(y_i(t)) > f(x_i(t+1)) \end{cases} \tag{3}$$

$$\begin{aligned} &\breve{y}(t) = min f(y), f(\breve{y}(t)) \\ &y \in y_0(t), y_1(t), ..., y_s(t) \end{aligned} \tag{4}$$

O algoritmo consiste na execução iterativa das equações apresentadas. O pseudocódigo do PSO é apresentado na Figura 3.

Na próxima seção será descrito a adaptação do PSO para o problema de classificação.

4. PSO para Problemas de Classificação

Recentemente, alguns trabalhos apresentaram uma adaptação do PSO para os problemas de classificação, sendo a maioria deles baseada em regras de classificação. Em

Inicializar as posições e velocidades das partículas aleatoriamente no espaço n-dimensional;
Encontrar o $pbest$ de cada partícula;
Encontrar o $gbest$;
para cada *partícula* $i \in [1..s]$ **faça**
    se x_i *melhor que* $pbest_i(t)$ **então**
        $pbest_i(t) = x_i$
    fim se
    se $pbest_i(t)$ *melhor que* $gbest(t)$ **então**
        $gbest(t) = pbest_i(t)$
    fim se
    Atualiza a velocidade (v_i) da partícula i
    Atualiza a posição (x_i) da partícula i
fim para cada

Figura 3. Pseudo-código do PSO

[Sousa et al. 2004], o PSO é usado para extrair regras de indução para classificar dados, sendo o algoritmo do PSO executado diversas vezes, gerando em cada execução uma regra e utilizando apenas os padrões não classificados em suas próximas iterações. Em [Wang et al. 2006] e [Wang et al. 2007], o PSO é utilizado a extração de regras em problemas de classificação discretos. Já em [Esmin 2007] o PSO foi usado para extrair regras *fuzzy*.

Na estratégia padrão do PSO, a solução em potencial é codificada em apenas uma partícula, sendo que toda a informação a ser codificada é o conjunto de protótipos e as suas classes. Há um outro método, chamado de *Michigan Particle Swarm Optimization* (MPSO) que utiliza a abordagem *Michigan*. Os resultados encontrados na literatura do método MPSO são bastante competitivos em comparação com outros métodos. Na abordagem MPSO, cada membro da população não codifica toda a solução para o problema, mas apenas parte dele. Todo o enxame é uma potencial solução do problema ([Cervantes et al. 2009b, Sousa et al. 2004])

O pseudo-código do método MPSO é mostrado na Figura 4 e descrito em detalhes em [Cervantes et al. 2009b, Sousa et al. 2004]. A equação de posição foi modificada quando comparado ao método PSO padrão, no qual foi introduzido a força de repulsão e o uso de vizinhança dinâmica da partícula. Quando está se movendo, cada partícula seleciona uma outra que é chamada de ”não competidora”para ser o centro de atração, e uma segunda partícula para ser o centro de repulsão.

Durante cada iteração, as partículas ”competidoras”e ”não competidoras”são definidas dinamicamente e, para isso, considera as classes das partículas. Neste caso, as partículas ”não competidoras”são aquelas de classes diferentes e as ”competidoras”possuem a mesma classe. O MPSO utiliza o conceito de *fitness* local, sendo cada partícula é avaliada como “boa”se classifica corretamente os padrões próximos. O cálculo do *fitness* local não considera como o restante dos padrões são classificados ([Cervantes et al. 2009b]).

O MPSO é muito eficiente na geração de um enxame altamente especializado em reconhecimento de padrões e pode ser facilmente adaptado a diversas classes de problemas ([Cervantes et al. 2009a, Cervantes et al. 2009b]). Nestra trabalho, apresentamos

Carregue as instâncias de treino;
Inicializa o enxame;
Inserir N partículas para cada classe das instâncias de treino;
enquanto *(max iteração ou taxa de sucesso de 100%)* **faça**
    Calcular para cada classe, as partículas são competidoras e não competidoras ;
    para cada *partícula* **faça**
        Calcular o *Fitness* local;
        Calcular o fator de adaptação social;
        Encontrar a partícula de mesma classe e que esteja no conjunto de partículas competidoras (centro de repulsão);
        Encontrar a partícula de mesma classe e que esteja no conjunto de partículas não competidoras (centro de atração);
        Calcular a próxima posição da partícula baseada em sua velocidade anterior, sua melhor posição, centros de atração e repulsão;
    fim para cada
    Movimente as partículas;
    Atribuir classes às instâncias de treino, utilizando a partícula mais próxima;
    Avaliar a taxa de acerto da classificação feita pelo enxame;
    Se o enxame conseguiu um melhor resultado, armazena todas as posições das partículas como ”melhor enxame”;
fim enqto
Apagar do enxame de partículas aquelas partículas que, ao serem retiradas, não afetam a taxa de acerto do enxame.;

Figura 4. Pseudo Algoritmo do MPSO

uma adaptação do MPSO para resolver o problema de classificação multirrótulo chamado de ML-KMPSO. Como o próprio nome indica, o ML-KMPSO é derivado do ML-KNN e do MPSO. Na Figura 5 é demonstrado o pseudo código do algoritmo.

1. Calcula a probabilidade a priori de todos os rótulos do conjunto de treinamento
2. Encontra o enxame de partículas usando o MPSO (Figura 4)
3. Agrega o enxame de partículas aos dados de treinamento
4. Calcula a probabilidade a posteriori de todos os rótulos usando os k vizinhos
5. Determina os rótulos de cada instância de teste utilizando o MAP

Figura 5. Pseudo código do ML-KMPSO

Como é descrito na Figura 5, inicialmente é calculada a probabilidade *a priori* dos rótulos do conjunto de treinamento, então é gerado o modelo de classificação usando o MPSO (passo 2) e identificado um conjunto de partículas especializadas em classificação (o enxame). Note que no passo 5 do PSO (Figura 4), partículas que não são utilizadas são marcadas para serem retiradas da solução (enxame), iniciando por aquela que detém o pior *fitness*. Após a verificação de que esta ação não irá interferir na qualidade da classificação do enxame, a partícula é então retirada. Finalmente, utilizando as informações estatísticas obtidas dos rótulos do conjunto de vizinhos, incluindo as partículas resultantes do MPSO, é aplicado o princípio do *Maximum a Posteriori* (MAP) para determinar os rótulos que

serão aplicados à instância de teste.

Na próxima seção avaliaremos o desempenho experimental do ML-KMPSO usando bases de dados reais.

5. Experimentos

Nesta seção avaliaremos o desempenho do ML-KMPSO usando duas diferentes bases de dados: a base *Yeast* e a base *Scene*. Os resultados obtidos serão comparados com os dos métodos ML-KNN ([Zhang and Zhou 2005, Zhang and Zhou 2007]), BoosTexter ([Schapire and Singer 2000]) e RankSVM ([Elisseeff and Weston 2001]).

5.1. Base de Dados

Como mencionado, para experimentos foram utilizadas duas bases de dados reais multirrótulos: a base *Yeast*([Elisseeff and Weston 2005]) e a base *Scene* ([Boutell et al. 2004]).

Yeast é uma base de dados multirrótulos biológica obtida através de estudos realizados sobre a levedura *Saccharomyces cerevisiae*, um dos organismos mais estudados da literatura. A base é descrita como um conjunto de micro vetores que expressam as características e os perfis filogenéticos de 2417 genes, onde cada gene está associado a um conjunto de 14 classes funcionais (Figura 6). [Elisseeff and Weston 2001] realizaram o pre-processamento da base, a fim de tornar fácil o seu uso e somente as classes funcionais conhecidas serão usadas.

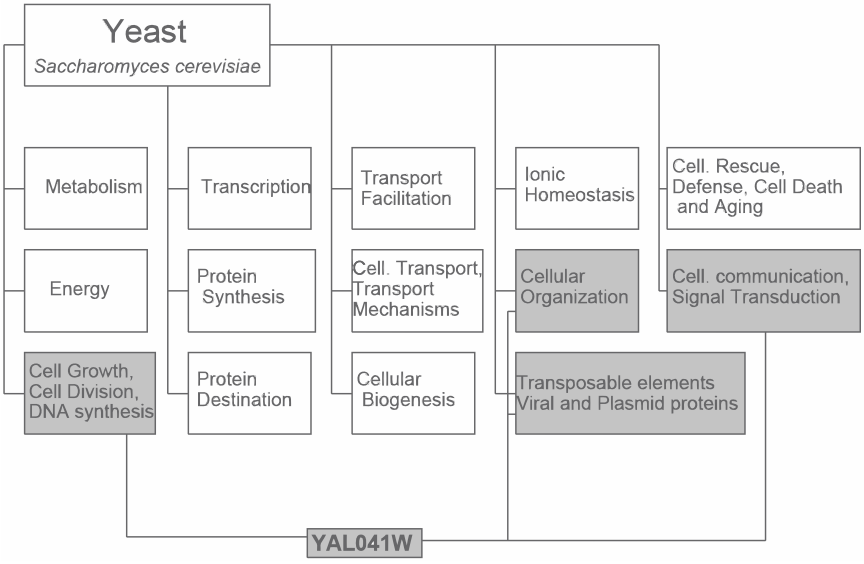

Figura 6. Classes Funcionais dos Genes da Levedura *Saccharomyces cerevisiae* [Elisseeff and Weston 2005]

A base *Scene* contém, no total, 2000 imagens naturais, todas elas são classificadas manualmente e representadas por um vetor de 294 dimensões e 5 possíveis rótulos, cada imagem possuindo em média 1.24 rótulos. O objetivo do classificador nesta base é que, após o treinamento, possa classificar automaticamente imagens que ainda serão apresentadas a ele.

A Tabela 1 mostra algumas informações estatísticas das duas bases que foram utilizadas nos experimentos.

Tabela 1. Informações Sobre as Bases Multirrótulos

Bases	Instâncias	Atributos	Rótulos	Cardinalidade	Densidade
Yeast	2417	103	14	4.327	0.302
Scene	2000	294	5	1.24	0.248

5.2. Métricas de Avaliação

Para avaliar o desempenho dos métodos de aprendizagem foram escolhidas as métricas comumente utilizadas na literatura nos problemas de classificação multirrótulo e que foram propostas por [Schapire and Singer 2000].

Hamming Loss: avalia quantas vezes o par exemplo-rótulo é erroneamente classificado, ou seja, um rótulo não pertence ao exemplo e foi previsto, ou um rótulo que pertença ao exemplo e não foi previsto. Nesta métrica, quanto menor o valor do *Hamming Loss* melhor é a acurácia do classificador.

$$HammingLoss_S(h) = \frac{1}{P}\sum_{i=1}^{P}|h(x_i)\triangle Y_i| \tag{5}$$

onde $\triangle$ representa a diferença simétrica entre os conjuntos dos rótulos previstos $(h(x_i))$ e os rótulos corretos (Y_i).

One-error: avalia quantas vezes o melhor rótulo não pertence ao conjunto de possíveis rótulos.

$$One - error_S(f) = \frac{1}{P}\sum_{i=1}^{P}\left[[\texttt{arg}_{y\in\mathcal{Y}}\texttt{max}\ \ f(x_i,y)] \notin Y_i\right] \tag{6}$$

onde $i \in \mathcal{Y}$. O termo $[\texttt{arg}_{y\in\mathcal{Y}}\texttt{max}\ \ f(x_i,y)]$ representa o valor máximo da função $f()$. Quanto menor o valor do $One - Error_S(f)$ melhor é o desempenho.

Coverage: é definido como a distância em termos de rótulos no seu *ranking* para cobrir todos os possíveis rótulos de uma instância x.

$$Coverage_S(f) = \frac{1}{P}\sum_{i=1}^{P}\texttt{max}_{y\in Y}rank_f(x_i,y) - 1 \tag{7}$$

onde $i \in \mathcal{Y}$. Quanto menor o valor do *Coverage*, melhor é o desempenho do classificador.

Ranking Loss: avalia a quantidade média dos rótulos que estão inversamente ordenados para a instância. Quanto menor o valor do *Ranking Loss* melhor é o desempenho do classificador.

$$RankLoss_S(f) = \frac{1}{P}\sum_{i=1}^{P}\frac{1}{|Y_i||\overline{Y}_i|}|(y_1,y_2)|f(x_i,y_1) \not< f(x_i,y_2),(Y_1,y_2) \in Y_i \times \overline{Y}_i| \tag{8}$$

Average Precision: avalia a proporção média de rótulos que ocupam, no *ranking*, uma posição superior a um dado rótulo, e que pertencem ao conjunto de rótulos desejados ([de Carvalho and Freitas 2009]). Quanto maior o valor desta métrica melhor é o desempenho do classificador.

$$AveragePrecision_S(f) = \frac{1}{P}\sum_{i=1}^{P}\frac{1}{|Y_i|}\times\sum_{y\in Y_i}\frac{|\{y'|rank_f(x_i,y') \not< rank_f(x_i,y'), y' \in Y_i\}|}{rank_f(x_i,y)} \tag{9}$$

5.3. Análise dos Resultados

A fim de garantir resultados mais confiáveis, todos os experimentos foram realizados utilizando o *ten fold cross validation method*. Neste método de avaliação experimental, o conjunto de dados é dividido em 10 *fold*, e são realizados 10 experimentos, onde cada experimento usa um *fold* diferente para teste e os restantes nove *fold* para treinamento.

A Tabela 2 apresenta a comparação do ML-KMPSO com os algoritmos do estado da arte para o problema de classificação multirrótulo utilizando a base *Yeast*. Para cada métrica de avaliação, a seta para baixo indica que quanto menor, melhor, enquanto a seta para cima indica que quanto maior, melhor.

Tabela 2. Resultados na Base *Yeast*

Métricas	Algoritmos			
	ML-KNN	BoosTexter	Rank-SVM	ML-KMPSO
Hamming Loss↓	**0.194±0.010**	0.220±0.011	0.207±0.013	0.198±0.008
One-Error↓	0.230±0.030	0.278±0.034	0.243±0.039	**0.225±0.045**
Coverage↓	**6.275±0.240**	6.550±0.243	7.090±0.503	6.355±0.232
Ranking Loss↓	**0.167±0.016**	0.186±0.015	0.195±0.021	0.172±0.009
Average Precision↑	**0.765±0.021**	0.737±0.022	0.749±0.026	0.762±0.012

Como visto nesta tabela, o ML-KMPSO apresentou melhor desempenho em todas as cinco métricas em relação ao BoosTexter e SVM-Rank. Comparando com o ML-KNN, foi melhor apenas na métrica *One-Error*, e similar nas demais métricas. O ranking dos melhores métodos para cada métrica é mostrada na Tabela 3.

Tabela 3. Relação de Performance dos Métodos de Classificação Multirrótulo para a Base *Yeast*

Métricas	Algoritmos
HL	ML-KNN > ML-KMPSO > Rank-SVM > BoosTexter
OE	ML-KMPSO > ML-KNN > Rank-SVM > BoosTexter
Co	ML-KNN > ML-KMPSO > BoosTexter > Rank-SVM
RL	ML-KNN > ML-KMPSO > BoosTexter > Rank-SVM
AP	ML-KNN > ML-KMPSO > Rank-SVM > BoosTexter

Os resultados para a base Scene estão na Tabela 4. O ML-KMPSO se mostrou muito promissor, sendo o melhor nas métrias *Hamming Loss* e *Coverage metrics*, o segundo em *One-Error* e terceiro no *Ranking Loss* e *Average Precision*. Os bons resultados associados ao *Coverage* (Co) e ao *Hamming Loss* (HL) são consequência da baixa cardinalidade e densidade da base, que favorece ao MPSO, que resulta em classificadores especializados e é capaz de prever com precisão um grande número de tuplas (instâncias x_i - rótulos Y_i). Por outro lado, o Rank-SVM obteve os piores resultados.

Tabela 4. Resultados na Base *Image*

Métricas	Algoritmos			
	ML-KNN	BoosTexter	Rank-SVM	ML-KMPSO
Hamming Loss↓	0.169±0.016	0.179±0.015	0.253±0.055	**0.153±0.010**
One-Error↓	**0.300±0.036**	0.311±0.041	0.491±0.135	0.311±0.036
Coverage↓	0.939±0.100	0.939±0.092	1.382±0.381	**0.872±0.274**
Ranking Loss↓	**0.168±0.024**	0.168±0.020	0.278±0.096	0.173±0.013
Average Precision↑	**0.803±0.027**	0.798±0.024	0.682±0.093	0.796±0.018

Tabela 5. Relação de Performance dos Métodos de Classificação Multirrótulo para a Base *Image*

Métricas	Algoritmos
HL	ML-KMPSO > ML-KNN > BoosTexter > Rank-SVM
OE	ML-KNN > ML-KMPSO > BoosTexter > Rank-SVM
Co	ML-KMPSO > ML-KNN > BoosTexter > Rank-SVM
RL	ML-KNN > BoosTexter > ML-KMPSO > Rank-SVM
AP	ML-KNN > BoosTexter > ML-KMPSO > Rank-SVM

Os resultados dos experimentos mostram que o algoritmo proposto supera outras abordagens como Rank-SVM, BoosTexter e empata com o ML-KNN. Acreditamos que ajustes nos parâmetros do MPSO dentro da estratégia proposta, podem resultar em resultados melhores que o ML-KNN.

6. Conclusão

O trabalho apresentou uma nova abordagem híbrida para resolver o problema de classificação multirrótulo chamado ML-KMPSO, que é uma adaptação do Michigan Particle Swarm Optimization para o problema de classificação multirrótulo. O ML-KMPSO foi comparado com outros métodos do estado da arte para esse problema (ML-KNN, RankSVM e BoosTexter) e os resultados dos experimentos, no qual foram utilizadas bases de dados reais, mostrou que o novo método superou ou ao menos se igualou aos métodos existentes, demonstrando que estratégias que representam um compromisso entre soluções globais e locais são interessantes.

Como trabalhos futuros tem-se como meta a realização de novos experimentos ajustando parâmetros e utilizando novas bases de dados multirrótulos (como a base Yahoo de categorização de páginas web) para a completa avaliação da eficiência do ML-KMPSO.

7. Agradecimentos

Os autores agradecem ao CNPq, à FAPEMIG pelo auxílio dado a esta pesquisa.

Referências

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

Cervantes, A., Galván, I. M., and Isasi, P. (2009a). Ampso: a new particle swarm method for nearest neighborhood classification. *Trans. Sys. Man Cyber. Part B*, 39:1082–1091.

Cervantes, A., Galván, I. M., and Viñuela, P. I. (2009b). Michigan particle swarm optimization for prototype reduction in classification problems. *New Generation Comput.*, 27(3):239–257.

de Carvalho, A. and Freitas, A. (2009). *A tutorial on multi-label classification techniques*, volume Foundations of Computational Intelligence Vol. 5 of *Studies in Computational Intelligence 205*, pages 177–195. Springer.

Eberhart, R. and Kennedy, J. (1995). A new optimizer using particle swarm theory. pages 39–43.

Elisseeff, A. and Weston, J. (2001). A kernel method for multi-labelled classification. In *In Advances in Neural Information Processing Systems 14*, pages 681–687. MIT Press.

Elisseeff, A. and Weston, J. (2005). A kernel method for multi-labelled classification. In *Annual ACM Conference on Research and Development in Information Retrieval*, pages 274–281.

Esmin, A. A. A. (2007). Generating fuzzy rules from examples using the particle swarm optimization algorithm. In *HIS*, pages 340–343.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4.

Li, T., Zhang, C., and Zhu, S. (2006). Empirical studies on multi-label classification. In *ICTAI '06: Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence*, pages 86–92, Washington, DC, USA. IEEE Computer Society.

Quinlan, J. R. (1993). *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Schapire, R. E. and Singer, Y. (2000). BoosTexter: A Boosting-based System for Text Categorization. *Machine Learning*, 39(2/3):135–168.

Sousa, T., Silva, A., and Neves, A. (2004). Particle swarm based data mining algorithms for classification tasks. *Parallel Comput.*, 30(5-6):767–783.

Trohidis, K., Tsoumakas, G., Kalliris, G., and Vlahavas, I. (2008). Multilabel classification of music into emotions. In *Proc. 9th International Conference on Music Information Retrieval (ISMIR 2008), Philadelphia, PA, USA, 2008*.

Wang, Z., Sun, X., and Zhang, D. (2006). Classification rule mining based on particle swarm optimization. In *RSKT*, pages 436–441.

Wang, Z., Sun, X., and Zhang, D. (2007). A pso-based classification rule mining algorithm. In Huang, D.-S., Heutte, L., and Loog, M., editors, *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence*, volume 4682 of *Lecture Notes in Computer Science*, pages 377–384. Springer Berlin / Heidelberg.

Zhang, M.-L. and Zhou, Z.-H. (2005). A k-nearest neighbor based algorithm for multi-label classification. In *GrC*, pages 718–721.

Zhang, M.-L. and Zhou, Z.-H. (2006). Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18:1338–1351.

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