Algoritmos de Aprendizado Baseado em Grafos para Classificação Multirrótulo

Everton Alvares Cherman[1], **Newton Spolaôr**[1]
Jorge Carlos Valverde-Rebaza[1], **Maria Carolina Monard**[1]

[1]Laboratório de Inteligência Artificial (Labic)
Instituto de Ciências Matemáticas e de Computação (ICMC)
Universidade de São Paulo (USP)
Caixa Postal 668, 13560-970 – São Carlos – SP – Brasil

{echerman,newtonsp,jvalverr,mcmonard}@icmc.usp.br

Abstract. *Multi-label learning algorithms deal with problems where each example can be associated with multiple labels simultaneously. This paper evaluates the use of graph-based learning techniques in multi-label learning. To this end, three algorithms,* MLϵ*,* MLMUT *and* MLnotMUT*, are proposed and experimentally compared to other traditional multi-label learning algorithms. The experimental results show that* MLMUT *and* MLnotMUT *are competitive with the traditional multi-label classification algorithms evaluated in this work.*

Resumo. *Algoritmos de aprendizado multirrótulo tratam problemas nos quais os exemplos podem estar associados com múltiplos rótulos simultaneamente. Neste trabalho são avaliadas técnicas de aprendizado baseado em grafos na tarefa de aprendizado multirrótulo. Três algoritmos,* MLϵ*,* MLMUT *e* MLnotMUT*, são propostos e comparados experimentalmente com outros algoritmos tradicionais de aprendizado multirrótulo. Os resultados experimentais mostram que os algoritmos* MLMUT *e* MLnotMUT *são competitivos com os métodos tradicionais de classificação multirrótulo avaliados neste trabalho.*

1. Introdução

Na classificação monorrótulo cada exemplo do conjunto de treinamento está associado a um único valor da classe (rótulo), enquanto que na classificação multirrótulo cada exemplo pode estar associado a vários valores da classe, *i.e.*, a um conjunto de rótulos. A principal diferença entre aprendizado multirrótulo e monorrótulo é que o conjunto de rótulos atribuídos a um exemplo no aprendizado multirrótulo estão frequentemente correlacionados, enquanto que os possíveis valores da classe (rótulos) no aprendizado monorrótulo são mutuamente exclusivos [Tsoumakas et al. 2010].

Diversos algoritmos de aprendizado multirrótulo tem sido propostos na literatura, os quais podem ser caracterizados de diferentes maneiras [Tsoumakas et al. 2010]. Neste trabalho consideramos algoritmos das duas seguintes categorias: transformação do problema e adaptação do algoritmo. Algoritmos na primeira categoria transformam inicialmente o conjunto de dados multirrótulo em um ou vários conjuntos de dados monorrótulo. Após essas transformações, qualquer algoritmo de aprendizado monorrótulo pode ser utilizado como algoritmo base para resolver o problema multirrótulo. Algoritmos na segunda categoria são capazes de realizar aprendizado em conjuntos de dados multirrótulo

de modo direto. Na maioria dos casos, esses algoritmos consistem em adaptações de idéias utilizadas no desenvolvimento de algoritmos de aprendizado monorrótulo.

Neste trabalho propomos a adaptação de três algoritmos de aprendizado em grafos para realizar aprendizado multirrótulo, a qual não é de nosso conhecimento que tenha sido considerada e/ou publicada pela comunidade. Os três algoritmos propostos, denominados *MLMUT*, *MLnotMUT* e *ML*ϵ foram avaliados experimentalmente em 10 conjuntos de dados, juntamente com três algoritmos da categoria transformação do problema e um algoritmo da categoria adaptação do algoritmo propostos na literatura. Os resultados experimentais mostram bons resultados para os algoritmos *MLMUT* e *MLnotMUT* em comparação com os outros algoritmos avaliados.

Este trabalho está organizado da seguinte maneira: na Seção 2 são apresentados os principais conceitos sobre aprendizado multirrótulo; na Seção 3 são descritos os algoritmos propostos neste trabalho. A avaliação experimental é apresentada na Seção 4. Conclusões e trabalhos futuros são apresentados na Seção 5.

2. Aprendizado Multirrótulo

No aprendizado *monorrótulo*, a entrada do algoritmo consiste de um conjunto de N exemplos de treinamento E_i, com $i = 1, ..., N$, escolhidos de um domínio com uma distribuição $\mathcal{D}$ da forma $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$ para alguma função desconhecida $y = f(\mathbf{x})$. Os $\mathbf{x}_i$ são tipicamente vetores da forma $(x_{i1}, x_{i2}, \ldots, x_{iM})$ com valores categóricos ou numéricos, tal que x_{ij} refere-se ao valor do atributo X_j do exemplo E_i, como apresentado na Tabela 1(a). Os valores (rótulos) y_i referem-se às possíveis classes que Y pode assumir. Com esse conjunto de treinamento o algoritmo de aprendizado gera um *classificador* H, que aproxima f, *i.e.*, $H(\mathbf{x}) \approx f(\mathbf{x})$. Dado um novo exemplo $\mathbf{x}$, o classificador, ou hipótese H, prediz o valor correspondente de y. Assim, na classificação monorrótulo, cada exemplo E_i está associado a um *único* valor da classe, o qual é o rótulo y_i contido no conjunto de rótulos L, *i.e.*, $y_i \in L$, com $|L| > 1$. Quando existem mais de dois valores de classe possíveis, o problema é chamado *classificação multiclasse*. Caso o valor de classe seja sim/não, o problema é chamado *classificação binária*.

Por outro lado, na classificação *multirrótulo*, um exemplo pode estar associado a mais de um rótulo, *i.e.*, um exemplo E_i está associado a um conjunto de rótulos Y_i. O exemplo E_i é representado da forma $(\mathbf{x}_i, Y_i)$, onde $Y_i \subseteq L$, $Y_i \neq \emptyset$ e $L = \{y_1, y_2, y_3, ...y_q\}$ é o conjunto dos q rótulos simples que participam dos multirrótulos Y_i, $i = 1, \ldots, N$, como representado na Tabela 1(b). Neste caso, o classificador gerado prediz o multirrótulo Y de um novo exemplo $\mathbf{x}$.

Tabela 1. Exemplos rotulados no formato atributo-valor.

(a) Monorrótulo

	X_1	X_2	...	X_M	Y
E_1	x_{11}	x_{12}	...	x_{1M}	y_1
E_2	x_{21}	x_{22}	...	x_{2M}	y_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
E_N	x_{N1}	x_{N2}	...	x_{NM}	y_N

(b) Multirrótulo

	X_1	X_2	...	X_M	Y
E_1	x_{11}	x_{12}	...	x_{1M}	Y_1
E_2	x_{21}	x_{22}	...	x_{2M}	Y_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
E_N	x_{N1}	x_{N2}	...	x_{NM}	Y_N

Como mencionado, algoritmos multirrótulo podem ser caraterizados de diferentes maneiras. Neste trabalho utilizamos algoritmos das categorias *transformação do pro-*

blema e *adaptação de algoritmo*. Na primeira categoria, a qual transforma dados multirrótulo para um ou vários conjuntos de dados monorrótulos, várias transformações têm sido propostas. Neste trabalho utilizamos o método de transformação *Binary Relevance* (*BR*). A transformação *BR* decompõe inicialmente o problema multirrótulo em $|L|$ problemas de classificação binária, um para cada rótulo contido em L. Após a transformação, cada um dos $|L|$ conjuntos de dados monorrótulo resultante é submetido a um algoritmo de aprendizado monorrótulo. Dado um novo exemplo a ser classificado, o multirrótulo predito consiste dos rótulos preditos como positivos por cada um dos $|L|$ classificadores binários. Uma vantagem da abordagem *BR* é que qualquer algoritmo monorrótulo pode ser utilizado como algoritmo base. Porém, uma desvantagem de *BR* é que não considera a dependência de rótulos na construção do modelo de classificação multirrótulo, pois cada classificador monorrótulo é construído independentemente dos outros.

Na segunda categoria, adaptação do algoritmo, vários algoritmos monorrótulo frequentemente utilizados pela comunidade tem sido adaptados para multirrótulo, tais como: o algoritmo *C4.5 Multirrótulo* [Clare and King 2001], que é uma adaptação do algoritmo de árvore de decisão *C4.5*; o algoritmo *BP-MLL* [Zhang 2006], o qual apresenta uma adaptação da estratégia de retropropagação (*back-propagation*) para tratar diretamente dados multirrótulo; os algoritmos *MLkNN* [Zhang and Zhou 2005], *DMLkNN* [Younes et al. 2011] e *BRkNN*[1] [Spyromitros et al. 2008], que propõem diferentes adaptações do algoritmo monorrótulo *kNN*. Nos algoritmos propostos neste trabalho, o multirrótulo de um novo exemplo é construído utilizando o método proposto em uma extensão do algoritmo *BRkNN*, o qual é brevemente descrito a seguir.

2.1. Algoritmo *BRkNN*

O algoritmo *lazy* de classificação monorrótulo k Vizinhos mais Próximos ou k *Nearest Neighbors* (*kNN*), baseia-se na busca pelos k exemplos mais próximos do exemplo a ser predito, de acordo com uma medida de similaridade. O novo rótulo pode ser obtido, por exemplo, considerando o rótulo mais frequente dos k exemplos vizinhos. A simplicidade desse algoritmo pode ser um dos motivos que contribuem para que distintos trabalhos proponham adaptações e possibilitem seu uso para a classificação multirrótulo.

No caso do *BRkNN* duas maneiras foram propostas para predizer o multirrótulo de um novo exemplo nas extensões *BRkNN-a* e *BRkNN-b* [Spyromitros et al. 2008]. Ambas extensões são baseadas em um valor de confiança de cada rótulo, o qual é estimado, a partir da porcentagem dos *k* vizinhos mais próximos que contém esse rótulo. A extensão *BRkNN-a* classifica um novo exemplo E com o multirrótulo que contém os rótulos com confiança superior a $0,5$, *i.e.*, rótulos incluídos em pelo menos metade dos multirrótulos dos *k* vizinhos mais próximos de E. Se nenhum rótulo atende essa condição, o multirrótulo predito é constituído apenas pelo rótulo de maior confiança. Por outro lado, a extensão *BRkNN-b* classifica o novo exemplo E com o multirrótulo que contem os $\lfloor s \rceil$ (s arredondado) rótulos que possuem a maior confiança, em que s é o número médio de rótulos que participam dos multirrótulos dos k vizinhos mais próximos de E. Nesse mesmo trabalho foram realizados experimentos comparativos com outros algoritmos *lazy*,

[1] Apesar do *BRkNN* ser conceitualmente entendido como a aplicação do algoritmo *Binary Relevance* com *kNN* como algoritmo-base, o que caracterizaria um algoritmo de transformação de problema, na prática, o problema multirrótulo é tratado diretamente e, por isso, esse algoritmo é incluído na categoria "adaptação do algoritmo".

como o *MLkNN* [Zhang and Zhou 2005], os quais indicaram a competitividade das extensões do *BRkNN*, em particular da extensão *BRkNN-b*. Considerando o bom desempenho do *BRkNN-b*, neste trabalho utilizamos essa idéia para determinar o multirrótulo que classifica o exemplo E nos três algoritmos propostos.

2.2. Medidas de Avaliação

A complexidade dos conjuntos de dados multirrótulo e o desempenho dos classificadores construídos podem ser mensurados por distintas medidas. Para caracterizar um conjunto de dados multirrótulo D, duas medidas são frequentemente utilizadas: a Cardinalidade de Rótulo (CR), definida pela Equação 1, que é a média do número de rótulos distintos nos multirrótulos de cada exemplo; e a Densidade de Rótulo (DR), definida pela Equação 2, semelhante a anterior mas ponderada pela quantidade de rótulos simples $|L|$ que participam dos multirrótulos. Ambas as métricas estão relacionadas por $CR(D) = |L|DR(D)$.

$$CR(D) = \frac{1}{|D|}\sum_{i=1}^{|D|}|Y_i| \qquad (1) \qquad\qquad DR(D) = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i|}{|L|}. \qquad (2)$$

Quanto às medidas de desempenho de classificadores multirrótulo, elas se diferenciam em relação às medidas utilizadas no aprendizado monorrótulo, no qual a classificação de um exemplo apresenta somente dois resultados possíveis: correta ou incorreta. As medidas de desempenho de classificadores multirrótulo devem levar em conta classificações *parcialmente corretas*. Com esse objetivo, várias medidas que utilizam critérios diferentes para avaliar a correção parcial desses classificadores têm sido propostas. Em [Dembczynski et al. 2012] é estabelecida uma conexão entre esses diferentes critérios, a qual mostra que alguns deles são não correlacionados ou negativamente correlacionados. Esse problema é ilustrado em [Metz et al. 2012].

As várias medidas propostas na literatura são agrupadas em três categorias: i) baseadas em exemplo; ii) basedas em rótulo; iii) baseadas em *ranking*. Uma discussão dessas medidas foge ao escopo deste trabalho e pode ser encontradas em [Tsoumakas et al. 2010]. A seguir são descritas as medidas baseadas em exemplos utilizadas neste trabalho para avaliar os classificadores construídos, onde Y_i é o multirrótulo verdadeiro do exemplo E_i, ou seja $E_i = (\mathbf{x}_i, Y_i)$, e H é um classificador multirrótulo que prediz o multirrótulo de E_i com $Z_i = H(\mathbf{x}_i)$. As medidas baseadas em exemplos consideram as diferenças entre o multirrótulo esperado, Y_i, e o predito, Z_i, nos exemplos do conjunto de teste. O valor dessas medidas varia no intervalo [0..1].

A medida *Hamming Loss*, definida pela Equação 3, baseia-se na distância de Hamming, *i.e.*, a diferença simétrica entre os conjuntos Y_i e Z_i, denotada por Δ.

$$Hamming\,Loss(H, D) = \frac{1}{N}\sum_{i=1}^{N}\frac{|Y_i \Delta Z_i|}{|L|}. \qquad (3)$$

Hamming Loss avalia a frequência com que o multirrótulo dos exemplos contém um rótulo errado ou um rótulo que deve estar no multirrótulo não é predito. Assim, quanto menor o valor, melhor a performance do classificador.

As outras medidas utilizadas, definidas a seguir, foram inspiradas nas medidas de avaliação de classificadores monorrótulo. Para essas medidas, quanto maior o valor, me-

lhor a performance do classificador. A medida *Subset Accuracy*, definida pela Equação 4, onde $I(true) = 1$ e $I(false) = 0$, somente considera predições totalmente corretas.

$$Subset\ Accuracy(H, D) = \frac{1}{N}\sum_{i=1}^{N} I(Z_i = Y_i). \tag{4}$$

A medida *Accuracy*, definida pela Equação 5, considera basicamente os acertos obtidos durante a predição de rótulos verdadeiros e a ausência de rótulos verdadeiros.

$$Accuracy(H, D) = \frac{1}{N}\sum_{i=1}^{N} \frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|}. \tag{5}$$

A medida *F-Measure*, definida pela Equação 6, calcula a média harmônica entre as medidas de avaliação multirrótulo $Precision(H, D) = \frac{1}{N}\sum_{i=1}^{N} \frac{|Y_i \cap Z_i|}{|Z_i|}$ e $Recall(H, D) = \frac{1}{N}\sum_{i=1}^{N} \frac{|Y_i \cap Z_i|}{|Y_i|}$.

$$F\text{-}Measure(H, D) = \frac{1}{N}\sum_{i=1}^{N} \frac{2|Y_i \cap Z_i|}{|Z_i| + |Y_i|}. \tag{6}$$

2.3. Algoritmo *Baseline* $Geral_B$

Como mencionado, a classificação multirrótulo deve também levar em conta classificações parcialmente corretas. Entretanto, um classificador multirrótulo construído por um algoritmo que maximiza/minimiza uma medida multirrótulo não necessariamente maximiza/minimiza as outras medidas.

Em [Metz et al. 2012] é discutida a importância de *baselines* no aprendizado multirrótulo, os quais representam uma referência de desempenho mínimo que um algoritmo de aprendizado deveria atingir. Utilizando um algoritmo que somente tem acesso aos multirrótulos dos exemplos, *i.e.*, os valores dos atributos não são vistos pelo algoritmo para gerar o classificador, são propostos classificadores que maximizam/minimizam uma única medida multirrótulo (em detrimento das outras), bem como o $Geral_B$ que gera um classificador que pode ser utilizado como *baseline* para todas as medidas multirrótulos, pois não favorece uma medida em particular. Neste trabalho utilizamos como *baselines* das medidas consideradas na avaliação experimental os resultados obtidos por $Geral_B$.

3. Algoritmos Propostos

Segundo [Jebara et al. 2009], para o uso de grafos em tarefas de aprendizado, três passos são necessários: (i) a escolha de uma função de similaridade ou distância para estimar a semelhança ou proximidade entre exemplos (vértices) ou o peso das arestas entre pares de vértices; (ii) a escolha de um algoritmo para encontrar um subgrafo esparso a partir do grafo totalmente conectado gerado no passo anterior — esse algoritmo pode podar algumas arestas ou reponderá-las; (iii) a seleção de um algoritmo de aprendizado baseado em grafos para determinar o rótulo de exemplos não rotulados.

No aprendizado multirrótulo tratado neste trabalho, no qual o conjunto de exemplos é representado no formato atributo-valor, o conjunto de vértices $V = \{V_1, V_2, ...V_N\}$

consiste dos exemplos de treinamento —Tabela 1(b) —, tal que $V_i = E_i$, onde $E_i = (\mathbf{x}_i, Y_i)$. A seguir, os termos vértice e exemplo serão utilizados indistintamente. Para quantificar a similaridade entre pares de exemplos utilizamos a métrica *overlap* para atributos nominais e a distância normalizada, $range\,normalized\,diff$, para atributos numéricos [Wilson and Martinez 2000]. Essa abordagem, denominada *Heterogeneous Euclidean-Overlap Metric* (HEOM), é definida pela Equação 7.

$$HEOM(x_{aj}, x_{bj}) = \begin{cases} overlap(x_{aj}, x_{bj}), \text{ se } X_j \text{ for nominal} \\ range\,normalized\,diff(x_{aj}, x_{bj}), \text{ se } X_j \text{ for numérico.} \end{cases} \quad (7)$$

onde o valor da função *overlap* é 1 para atributos nominais com valores diferentes e 0 caso contrário.

Após a obtenção desse grafo totalmente conectado, o próximo passo consiste na geração de um subgrafo esparso. Esse processo visa à obtenção dos vizinhos mais relevantes para determinar o multirrótulo de novos exemplos, além de diminuir o custo computacional nos passos seguintes.

Duas das principais heurísticas para a geração desses subgrafos são: abordagem baseada na similaridade minima e abordagem baseada nos vizinhos mais próximos [Brito et al. 1997]. A primeira abordagem, também conhecida como ϵ-vizinhança, consiste em criar uma aresta entre os exemplos E_i e E_j se $\| \mathbf{x}_i - \mathbf{x}_j \| \leq \epsilon$, sendo ϵ um limiar pré-definido. Ao se utilizar esta abordagem, se ϵ for um valor muito baixo, deve-se fazer um pós-processamento para que não haja vértices isolados no grafo. Por outro lado, se ϵ for um valor muito alto, a rede gerada pode ser muito densa. Devido à propriedade de simetria contida nesta abordagem, o grafo gerado é considerado não direcionado. O algoritmo multirrótulo *ML*ϵ implementa essa abordagem.

Na abordagem baseada em vizinhos mais próximos existem duas variações para a construção do grafo: direcionado e não direcionado. No primeiro caso, escolhe-se inicialmente k vizinhos para cada exemplo E_i, e gera-se uma aresta direcionada entre E_i e E_j, se E_j estiver entre os k vizinhos mais próximos de E_i. Essa abordagem, utilizada pelo algoritmo multirrótulo *BRkNN-b*, é semelhante a do método *kNN* padrão, no qual o número n_E de arestas por vértice é k, *i.e.*, $n_E = k$. No segundo caso, grafo não direcionado, as seguintes duas estratégias podem ser utilizadas: vizinhos mútuos e vizinhos não mútuos.

Na estratégia vizinhos mútuos, uma aresta não direcionada entre E_i e E_j é criada se e somente se E_j está no conjunto de vizinhos mais próximos de E_i e E_i está no conjunto de vizinhos mais próximos de E_j. Por ser uma estratégia mais restritiva na criação de arestas, o grafo construído pode ser mais desconexo, *i.e.*, $n_E \leq k$. O algoritmo *MLMUT* implementa essa estratégia [Liu et al. 2010]. Na estratégia vizinhos não mútuos, uma aresta não direcionada entre E_i e E_j é criada s.s.s E_j está no conjunto de vizinhos mais próximos de E_i ou E_i está no conjunto de vizinhos mais próximos de E_j. Assim, o grafo criado com esta abordagem pode conter $n_E \geq k$ arestas por vértice. O algoritmo *MLnotMUT* implementa essa estratégia. Em todos os casos, para classificar um novo exemplo $E = (\mathbf{x}, ?)$, ele é inserido no grafo como um novo vértice respeitando a abordagem que o algoritmo multirrótulo utiliza. O multirrótulo predito é aquele que contem os $[s]$ rótulos que possuem a maior confiança, onde s é o número médio de rótulos que participam dos n_E exemplos conectados com E.

A complexidade do algoritmo *MLϵ* é igual ao do *BRkNN*: $O(N \times M)$, onde N é o número de exemplos no conjunto de treinamento e M o número de atributos descritores do problema. Para os algoritmos *MLMUT* e *MLnotMUT*, as complexidades são respectivamente $O(k \times N \times M)$ e $O(N^2 \times M)$. No entanto, a eficiência desses algoritmos pode ser melhorada utilizando recursos como estruturas de dados especiais [Liu et al. 2010].

As adaptações propostas para aprendizado multirrótulo nos três algoritmos baseados em grafos *MLMUT*, *MLnotMUT* e *MLϵ* foram implementadas no *framework Mulan* [Tsoumakas et al. 2011], o qual disponibiliza livremente algoritmos para aprendizado multirrótulo e é muito utilizado pela comunidade. As implementações encontram-se disponíveis em http://www.labic.icmc.usp.br/pub/mcmonard/ExperimentalResults/ENIAC2013.zip

4. Avaliação Experimental

Os algoritmos propostos foram comparados com o algoritmo *BRkNN-b*[2], bem como com a abordagem *BR* utilizando como algoritmos base SMO (Suport Vector Machine), $J48$ (árvore de decisão) e NB (Naïve Bayes). Todos os experimentos foram realizados utilizando a estratégia de validação cruzada de 10-*folds* e o *framework Mulan*, o qual é implementado utilizando *Weka*[3]. Os classificadores foram avaliados em 10 conjuntos de dados utilizando as medidas de avaliação definidas na Seção 2.2. Figuras e resultados detalhados com desvio-padrão da avaliação experimental podem ser consultados em http://www.labic.icmc.usp.br/pub/mcmonard/ExperimentalResults/ENIAC2013.pdf. Devido a restrição de espaço, nesta seção são mostrados os resultados mais relevantes.

4.1. Conjuntos de Dados

Foram utilizados 10 conjuntos de dados disponíveis para a comunidade, descritos na Tabela 2.

Tabela 2. Descrição dos conjuntos de dados

Nome	Domínio	N	M	$Tipo$	$\|L\|$	CR	DR	#Diferentes
1-Cal500	música	502	68	numérico	174	26,044	0,150	502
2-Corel5k	imagem	5000	499	nominal	374	3,522	0,009	3175
3-Emotions	música	593	72	numérico	6	1,869	0,311	27
4-Enron	texto/e-mails	1702	1001	nominal	53	3,378	0,064	753
5-Genbase	biologia	662	1186	nominal	27	1,252	0,046	32
6-Magtag5k	música	5260	68	numérico	136	4,839	0,036	4163
7-Medical	texto/medicina	978	1449	nominal	45	1,245	0,028	94
8-Scene	imagem	2407	294	numérico	6	1,074	0,179	15
9-Slashdot	texto/artigos	3782	1079	numérico	22	1,181	0,041	156
10-Yeast	biologia	2417	103	numérico	14	4,237	0,303	198

Exceto o conjunto Magtag5k[4] descrito em maiores detalhes em [Marques et al. 2011], e Slashdot[5], os demais conjuntos de dados estão no repositório *Mulan*[6]. Na Tabela 2 é especificado o nome do conjunto de dados (Nome);

[2]Ainda que várias extensões desse algoritmo foram propostas e implementadas [Reis et al. 2012], neste trabalho utilizamos a versão padrão do *Mulan*.

[3]http://www.cs.waikato.ac.nz/ml/weka/

[4]http://tl.di.fc.ul.pt/t/magtag5k.zip

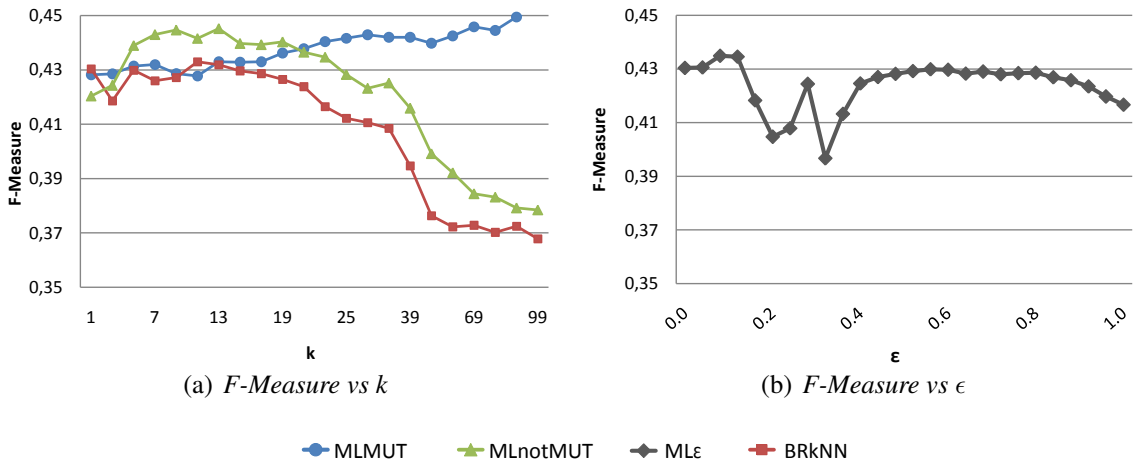
[5]http://meka.sourceforge.net/#datasets

[6]http://mulan.sourceforge.net/datasets.html

domínio de aplicação (Domínio); número de exemplos (N); número de atributos (M); tipo de atributos ($Tipo$); número de rótulos ($|L|$); cardinalidade de rótulos (CR); densidade de rótulos (DR); e número de multirrótulos diferentes (#Diferentes).

4.2. Configuração Experimental

O método de transformação *BR* utilizando três algoritmos base diferentes, denominados *BR(SMO)*, *BR(J48)* e *BR(NB)*, foi executado com valor padrão de parâmetros dos algoritmos base. Para os algoritmos *BRkNN-b*, *MLMUT* e *MLnotMUT*, é necessário escolher o valor de k. Assim, foi realizada uma análise da influência do valor de k na qualidade da predição, variando k no intervalo [1..27] com passo de 2, e no intervalo [29..99] com passo de 10. Para o algoritmo *ML*ϵ, foi realizada uma análise no intervalo [0..1] com passo de 0,02 para encontrar um valor de ϵ apropriado. Caso o número de exemplos considerados para determinar o multirrótulo de um novo exemplo (n_E) for nulo, o vizinho mais próximo é considerado. Nessa análise foi utilizada *F-Measure* como medida da qualidade de predição. Na Figura 1 são mostrados os valores de *F-Measure* em função de k para o conjunto de dados Enron. No material suplementar encontram-se as figuras correspondentes aos outros conjuntos de dados.



(a) *F-Measure vs k* (b) *F-Measure vs* ϵ

Figura 1. Desempenho de *F-Measure* na base de dados Enron

Na Tabela 3 são mostrados, para cada conjunto de dados, os valores de k e ϵ para os quais *F-Measure* atinge o maior valor. Esses valores foram utilizados nos experimentos comparativos com os outros métodos tradicionais. Para quatro conjuntos de dados, *ML*ϵ atingiu o melhor valor de *F-Measure* para mais de um valor de ϵ. Por exemplo, para o conjunto Slashdot, [0,18..0,22] na Tabela 3 indica que isso aconteceu para $\epsilon = 0,18; 0,20$ e $0,22$. Vale também lembrar que $n_E = k$ para *BRkNN-b*; $n_E \leq k$ para *MLMUT*; e $n_E \geq k$ para *MLnotMUT*. Assim, de maneira semelhante a *ML*ϵ, se para algum exemplo a ser classificado n_E for nulo, o vizinho mais próximo é considerado. Dessa maneira, nunca é predito o multirrótulo vazio.

4.3. Resultados

Na Tabela 4 são mostrados, para cada conjunto de dados, os valores das quatro medidas utilizadas para avaliar a capacidade de predição dos algoritmos, o *ranking* dos algoritmos

Tabela 3. Valores de k e ϵ com melhor valor de *F-Measure*

Nome	*MLMUT* k	*MLnotMUT* k	*ML*ϵ ϵ	*BRkNN-b* k
1-Cal500	99	79	0,4	59
2-Corel5k	99	23	[0,98..1]	21
3-Emotions	39	15	0,24	15
4-Enron	99	13	0,1	11
5-Genbase	1	1	[0..0,24]	1
6-Magtag5k	99	17	0,14	17
7-Medical	9	7	[0..0,12]	3
8-Scene	99	13	0,26	27
9-Slashdot	99	1	[0,18..0,22]	1
10-Yeast	69	29	0,56	21

em cada conjunto de dados, entre parênteses, e o *ranking* médio. A coluna $Geral_B$ mostra os respectivos *baselines*.

Pode ser observado que, em alguns casos, o valor de uma medida não atinge o *baseline* correspondente. No aprendizado multirrótulo, isso pode acontecer isoladamente para alguma medida e conjunto de dados. Porém, quando o *baseline* não é atingido em várias medidas, como no caso do algoritmo *BR(NB)* nos conjuntos de dados Cal500 e Enron, esse fato indica a incapacidade do algoritmo aprender desses conjuntos de dados.

Para analisar se há diferença significativa entre os algoritmos, o teste de Friedman, com a hipótese nula de que os algoritmos são equivalentes foi utilizado com cada medida de avaliação. Quando a hipótese nula é rejeitada pelo teste de Friedman, com confiança de 95%, o pós-teste de Nemenyi é utilizado para detectar quais diferenças entre os algoritmos são significativas [Demsar 2006]. De acordo com esse teste, a eficácia de dois algoritmos é significativamente diferente sempre que seus correspondentes *rankings* médio diferirem por pelo menos um determinado valor de diferença crítica (CD). Os resultados do pós-teste de Nemenyi podem ser representados em um diagrama nos quais o *ranking* médio dos algoritmos é representado no eixo X. Observe que valores menores do ranking médio são melhores.

Dado que em todos os casos a hipótese nula foi rejeitada pelo teste de Friedman, o pós-teste de Nemenyi foi utilizado. Na Figura 2, a diferença crítica (CD) é mostrada acima do eixo X e as linhas abaixo do eixo X conectam algoritmos que não apresentam diferença estatística significativa.

Nesse diagrama, pode ser observado que, para *Hamming Loss*, *BR(SMO)* é significativamente melhor que *ML*ϵ e *BR(NB)*, enquanto que *BR(J48)*, *MLnotMUT* e *MLMUT* são significativamente melhores que *BR(NB)*. Para *Subset Accuracy*, *MLMUT*, *BRkNN-b*, *BR(SMO)* e *MLnotMUT* são significativamente melhores que *BR(J48)* e *BR(NB)*. Para *Accuracy* e *F-Measure*, apenas os algoritmos *MLMUT* e *MLnotMUT* são significativamente melhores que *BR(NB)*. Vale lembrar que os valores de k foram selecionados utilizando *F-Measure* como critério de avaliação, o que, como é possível observar, reflitiu positivamente no *ranking* dos métodos *lazy* para essa medida. Também, não considerando *Hamming Loss*, é possível observar que *MLMUT* apresenta o melhor *ranking* médio entre todos os métodos considerados. Além disso, é importante destacar que os melhores valores obtidos por esse algoritmo ocorrem no limite superior da configuração do valor de k proposto, *i.e.*, com o valor $k = 99$, o que é um indício de que há ainda um potencial de melhora caso esse valor seja maior. O método *MLnotMUT* superou o método *lazy* padrão

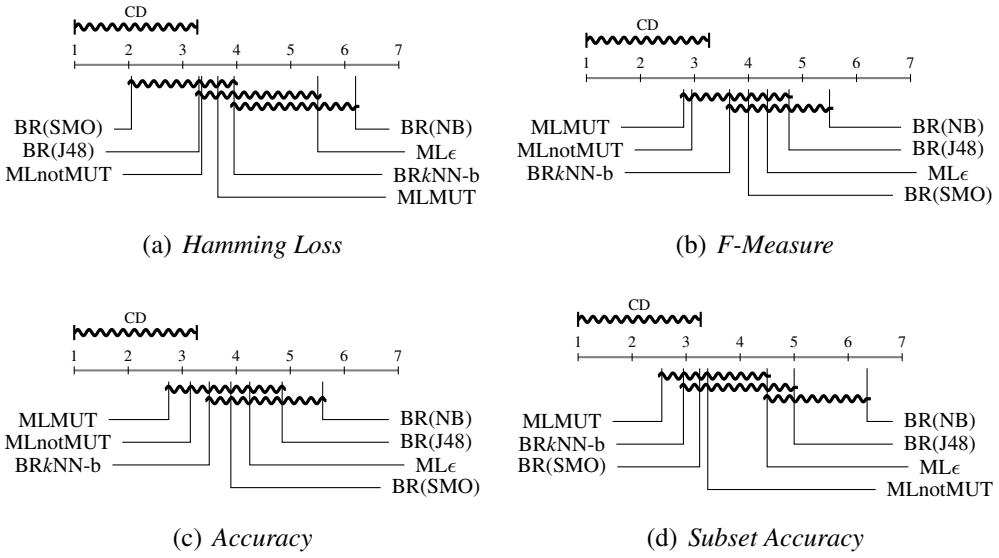
Tabela 4. Resultados da avaliação

Hamming-Loss								
Nome	*MLMUT*	*MLnotMUT*	*MLϵ*	*BRkNN-b*	*BR(SMO)*	*BR(J48)*	*BR(NB)*	$Geral_B$
1-Cal500	0,161(4,0)	0,158(2,0)	0,163(6,0)	0,159(3,0)	0,137(1,0)	0,162(5,0)	0,319(7,0)	0,219
2-Corel5k	0,015(5,0)	0,015(5,0)	0,016(7,0)	0,015(5,0)	0,011(2,0)	0,010(1,0)	0,013(3,0)	0,019
3-Emotions	0,208(2,5)	0,209(4,0)	0,222(5,0)	0,208(2,5)	0,193(1,0)	0,247(6,0)	0,252(7,0)	0,403
4-Enron	0,065(5,0)	0,062(3,5)	0,066(6,0)	0,062(3,5)	0,060(2,0)	0,052(1,0)	0,218(7,0)	0,073
5-Genbase	0,001(3,5)	0,001(3,5)	0,001(3,5)	0,001(3,5)	0,001(3,5)	0,001(3,5)	0,034(7,0)	0,075
6-Magtag5k	0,044(4,0)	0,044(4,0)	0,047(6,0)	0,044(4,0)	0,034(1,0)	0,042(2,0)	0,237(7,0)	0,104
7-Medical	0,018(4,5)	0,015(3,0)	0,019(6,0)	0,018(4,5)	0,010(1,5)	0,010(1,5)	0,026(7,0)	0,049
8-Scene	0,083(1,0)	0,086(2,0)	0,103(4,0)	0,094(3,0)	0,106(5,0)	0,137(6,0)	0,242(7,0)	0,379
9-Slashdot	0,069(4,0)	0,072(5,0)	0,073(6,5)	0,073(6,5)	0,046(2,0)	0,042(1,0)	0,068(3,0)	0,136
10-Yeast	0,200(3,0)	0,199(1,5)	0,204(5,0)	0,202(4,0)	0,199(1,5)	0,245(6,0)	0,303(7,0)	0,321
ranking médio	3,65	3,35	5,50	3,95	2,05	3,30	6,20	
Subset Accuracy								
Nome	*MLMUT*	*MLnotMUT*	*MLϵ*	*BRkNN-b*	*BR(SMO)*	*BR(J48)*	*BR(NB)*	$Geral_B$
1-Cal500	0,300(3,0)	0,311(1,0)	0,000(5,5)	0,308(2,0)	0,000(5,5)	0,000(5,5)	0,000(5,5)	0,002
2-Corel5k	0,024(1,0)	0,004(5,0)	0,000(7,0)	0,007(2,5)	0,007(2,5)	0,003(6,0)	0,005(4,0)	0,000
3-Emotions	0,282(2,0)	0,275(3,0)	0,271(4,0)	0,295(1,0)	0,270(5,0)	0,184(7,0)	0,206(6,0)	0,125
4-Enron	0,114(2,5)	0,052(6,0)	0,114(2,5)	0,105(4,0)	0,116(1,0)	0,098(5,0)	0,000(7,0)	0,064
5-Genbase	0,973(2,0)	0,970(6,0)	0,971(4,0)	0,971(4,0)	0,980(1,0)	0,971(4,0)	0,267(7,0)	0,110
6-Magtag5k	0,006(3,0)	0,006(3,0)	0,006(3,0)	0,007(1,0)	0,004(5,5)	0,004(5,5)	0,000(7,0)	0,000
7-Medical	0,522(5,0)	0,590(3,0)	0,511(6,0)	0,527(4,0)	0,657(1,0)	0,654(2,0)	0,265(7,0)	0,016
8-Scene	0,719(1,0)	0,715(2,0)	0,659(4,0)	0,689(3,0)	0,526(5,0)	0,427(6,0)	0,169(7,0)	0,016
9-Slashdot	0,284(3,0)	0,253(4,0)	0,249(5,0)	0,248(6,0)	0,363(1,0)	0,310(2,0)	0,229(7,0)	0,001
10-Yeast	0,184(3,0)	0,196(1,0)	0,183(4,0)	0,187(2,0)	0,148(5,0)	0,068(7,0)	0,095(6,0)	0,031
ranking médio	2,55	3,40	4,50	2,95	3,25	5,00	6,35	
Accuracy								
Nome	*MLMUT*	*MLnotMUT*	*MLϵ*	*BRkNN-b*	*BR(SMO)*	*BR(J48)*	*BR(NB)*	$Geral_B$
1-Cal500	0,460(3,0)	0,472(1,0)	0,297(4,0)	0,469(2,0)	0,204(6,0)	0,207(5,0)	0,203(7,0)	0,204
2-Corel5k	0,137(2,0)	0,116(4,0)	0,125(3,0)	0,109(5,0)	0,075(6,0)	0,063(7,0)	0,149(1,0)	0,048
3-Emotions	0,561(2,0)	0,556(3,0)	0,537(4,0)	0,563(1,0)	0,517(6,0)	0,462(7,0)	0,529(5,0)	0,291
4-Enron	0,360(3,0)	0,334(6,0)	0,346(4,0)	0,338(5,0)	0,408(1,0)	0,406(2,0)	0,193(7,0)	0,294
5-Genbase	0,988(3,0)	0,986(5,5)	0,988(3,0)	0,988(3,0)	0,990(1,0)	0,986(5,5)	0,289(7,0)	0,115
6-Magtag5k	0,266(2,0)	0,269(1,0)	0,233(4,0)	0,263(3,0)	0,070(7,0)	0,173(5,0)	0,101(6,0)	0,057
7-Medical	0,622(5,0)	0,679(3,0)	0,614(6,0)	0,624(4,0)	0,748(1,0)	0,744(2,0)	0,367(7,0)	0,017
8-Scene	0,754(1,0)	0,747(2,0)	0,700(4,0)	0,722(3,0)	0,596(5,0)	0,535(6,0)	0,453(7,0)	0,192
9-Slashdot	0,318(4,0)	0,287(5,0)	0,275(6,5)	0,275(6,5)	0,460(1,0)	0,361(3,0)	0,377(2,0)	0,043
10-Yeast	0,546(2,5)	0,548(1,0)	0,539(4,0)	0,546(2,5)	0,501(5,0)	0,440(6,0)	0,420(7,0)	0,308
ranking médio	2,75	3,15	4,25	3,50	3,90	4,85	5,60	
F-Measure								
Nome	*MLMUT*	*MLnotMUT*	*MLϵ*	*BRkNN-b*	*BR(SMO)*	*BR(J48)*	*BR(NB)*	$Geral_B$
1-Cal500	0,455(3,0)	0,471(1,0)	0,453(4,0)	0,467(2,0)	0,334(6,0)	0,338(5,0)	0,328(7,0)	0,329
2-Corel5k	0,193(3,0)	0,179(4,0)	0,200(2,0)	0,167(5,0)	0,106(6,0)	0,092(7,0)	0,218(1,0)	0,079
3-Emotions	0,657(1,5)	0,653(3,0)	0,629(5,0)	0,657(1,5)	0,597(6,0)	0,557(7,0)	0,633(4,0)	0,362
4-Enron	0,454(3,0)	0,445(4,0)	0,437(5,0)	0,433(6,0)	0,518(2,0)	0,519(1,0)	0,305(7,0)	0,390
5-Genbase	0,992(3,0)	0,990(5,5)	0,992(3,0)	0,992(3,0)	0,993(1,0)	0,990(5,5)	0,299(7,0)	0,116
6-Magtag5k	0,384(2,0)	0,388(1,0)	0,338(4,0)	0,377(3,0)	0,108(7,0)	0,261(5,0)	0,178(6,0)	0,102
7-Medical	0,657(5,0)	0,711(3,0)	0,648(6,0)	0,658(4,0)	0,779(1,0)	0,773(2,0)	0,403(7,0)	0,017
8-Scene	0,766(1,0)	0,757(2,0)	0,711(4,0)	0,733(3,0)	0,620(5,0)	0,573(6,0)	0,567(7,0)	0,253
9-Slashdot	0,330(4,0)	0,299(5,0)	0,284(6,5)	0,284(6,5)	0,494(1,0)	0,379(3,0)	0,435(2,0)	0,059
10-Yeast	0,654(2,5)	0,655(1,0)	0,647(4,0)	0,654(2,5)	0,611(5,0)	0,564(6,0)	0,538(7,0)	0,430
ranking médio	2,80	2,95	4,35	3,65	4,00	4,75	5,50	

BRkNN-b para *F-Measure*, *Subset Accuracy* e *Hamming Loss*, enquanto *MLϵ* ficou em uma posição intermediária. Assim, considerando os resultados experimentais, podemos concluir que os algoritmos *MLMUT* e *MLnotMUT* propostos neste trabalho mostram-se competitivos, melhorando na maioria das vezes os resultados obtidos por *BRkNN-b*.

5. Conclusão

Neste trabalho foram avaliadas técnicas de aprendizado baseado em grafos na tarefa de aprendizado multirrótulo. Foi realizada a adaptação de três dessas técnicas, as quais fo-



(a) *Hamming Loss* (b) *F-Measure*

(c) *Accuracy* (d) *Subset Accuracy*

Figura 2. Diagramas do pós-teste de Nemenyi.

ram implementadas nos algoritmos lazy de classificação multirrótulo *ML*ϵ, *MLMUT* e *MLnotMUT* propostos neste trabalho. Os três algoritmos foram experimentalmente avaliados e comparados com outros algoritmos multirrótulo propostos na literatura utilizando 10 conjuntos de dados publicamente disponíveis. Os métodos *MLMUT* e *MLnotMUT* apresentaram os melhores resultados quando comparados aos outros métodos avaliados, principalmente se considerada as medidas de avaliação *F-Measure* e *Accuracy*, nas quais obtiveram o melhor desempenho médio. Vale destacar também que o método *MLMUT* obteve um *ranking* médio superior ao algoritmo *lazy* de classificação multirrótulo *BRkNN-b* em todas as medidas de avaliação, o que sugere uma estratégia de construção de grafos mais eficaz que a usada pelo *BRkNN*. Como trabalho futuro, pretende-se avaliar outras técnicas de aprendizado baseado em grafos com o objetivo de adaptá-las para o desenvolvimento de novos algoritmos multirrótulo com intuito de considerar a dependência de rótulos, bem como explorar estratégias para identificar os melhores valores de k.

Agradecimentos: à Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) pelo auxílio recebido para a realização deste trabalho, processos n^o 2010/15992-0, 2011/02393-4 e 2011/22749-8, e também ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), processo n^o 151836 /2013-2. Os autores agradecem também aos avaliadores anônimos pelas sugestões para melhoria deste trabalho.

Referências

[Brito et al. 1997] Brito, M. R., Chávez, E. L., Quiroz, A. J., and Yukich, J. E. (1997). Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection. *Statistics & Probability Letters*, 35(1):33–42.

[Clare and King 2001] Clare, A. and King, R. D. (2001). Knowledge discovery in multi-label phenotype data. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 42–53. Springer-Verlag.

[Dembczynski et al. 2012] Dembczynski, K., Waegeman, W., Cheng, W., and Hüllermeier, E. (2012). On label dependence and loss minimization in multi-label classification. *Machine Learning*, 88(1-2):5–45.

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

[Jebara et al. 2009] Jebara, T., Wang, J., and Chang, S.-F. (2009). Graph construction and b-matching for semi-supervised learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 441–448, EUA.

[Liu et al. 2010] Liu, H., Zhang, S., Zhao, J., Zhao, X., and Mo, Y. (2010). A new classification algorithm using mutual nearest neighbors. In *Grid and Cooperative Computing (GCC), 2010 9th International Conference on*, pages 52–57.

[Marques et al. 2011] Marques, G., Domingues, M. A., Langlois, T., and Gouyon, F. (2011). Three current issues in music autotagging. In Klapuri, A. and Leider, C., editors, *ISMIR*, pages 795–800. University of Miami.

[Metz et al. 2012] Metz, J., de Abreu, L. F. D., Cherman, E. A., and Monard, M. C. (2012). On the estimation of predictive evaluation measure baselines for multi-label learning. In *IBERAMIA*, pages 189–198.

[Reis et al. 2012] Reis, D. M., Cherman, E. A., Spolaôr, N., and Monard, M. C. (2012). Extensões do algoritmo de aprendizado de máquina multirrótulo br. In *ENIA'2012: Encontro Nacional de Inteligência Artificial*, pages 1–10, Brasil.

[Spyromitros et al. 2008] Spyromitros, E., Tsoumakas, G., and Vlahavas, I. (2008). An empirical study of lazy multilabel classification algorithms. In *Hellenic conference on Artificial Intelligence*, pages 401–406, Berlin, Heidelberg. Springer-Verlag.

[Tsoumakas et al. 2010] Tsoumakas, G., Katakis, I., and Vlahavas, I. P. (2010). Mining multi-label data. In Maimon, O. and Rokach, L., editors, *Data Mining and Knowledge Discovery Handbook*, pages 667–685. Springer.

[Tsoumakas et al. 2011] Tsoumakas, G., Spyromitros, E., Vilcek, J., and Vlahavas, I. (2011). Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, 12:2411–2414.

[Wilson and Martinez 2000] Wilson, D. R. and Martinez, T. R. (2000). Reduction Techniques for Exemplar-Based Learning Algorithms. *Machine learning*, 38(3):257–286.

[Younes et al. 2011] Younes, Z., Abdallah, F., Denoeux, T., and Snoussi, H. (2011). A dependent multilabel classification method derived from the k-nearest neighbor rule. *EURASIP Journal on Advances in Signal Processing*, 2011:1–14.

[Zhang 2006] Zhang, M.-L. (2006). Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1338–1351.

[Zhang and Zhou 2005] Zhang, M.-L. and Zhou, Z.-H. (2005). A k-nearest neighbor based algorithm for multi-label classification. *IEEE International Conference on Granular Computing*, 2:718–721.