PROPOSTA DE UM ALGORITMO DE *CLUSTERING* SEMI-SUPERVISIONADO PARA ROTULAR EXEMPLOS A PARTIR DE POUCOS EXEMPLOS ROTULADOS

Marcelo Kaminski Sanches[1]
Maria Carolina Monard[2]

RESUMO

Recentemente, a comunidade científica tem mostrado um grande interesse no aprendizado semi-supervisionado. Esse interesse deve-se ao fato que, em muitas aplicações do mundo real, conjuntos de exemplos não rotulados são facilmente encontrados, ou muito baratos para serem coletados, quando comparados aos conjuntos de exemplos rotulados. Os exemplos não rotulados podem ser utilizados de diversas maneiras. Neste trabalho é proposto um algoritmo semi-supervisionado, denominado $k\text{-}means_{ki}$, o qual viabiliza o uso de exemplos não rotulados em aprendizado supervisionado. A técnica utilizada pelo algoritmo proposto está baseada em duas premissas. A primeira delas é que os exemplos tendem a se agrupar naturalmente em clusters, ao invés de se distribuirem uniformemente no espaço de descrição dos exemplos. Além disso, cada exemplo do conjunto inicial de exemplos rotulados deve estar localizado perto do centro de um dos clusters existentes no espaço de descrição de exemplos. A segunda premissa diz que a maioria dos exemplos nos clusters pertencem a uma classe específica. Obviamente, a validade dessas premissas é dependente do conjunto de dados utilizado. Experimentos realizados demonstram que o algoritmo $k\text{-}means_{ki}$ funciona bem nos casos em que os dados estão em conformidade com ambas as premissas.

Introdução

A abordagem utilizada pelo aprendizado não supervisionado, especificamente no processo de *clustering*, sofre de algumas limitações significantes. Diferentemente do que ocorre com o aprendizado supervisionado, os resultados de um processo de *clustering* não fornecem uma explicação ou descrição, mas apenas clusters. Porém, muitas vezes, não se está interessado apenas nos clusters, mas também em alguma explicação ou descrição conceitual dos exemplos que foram agrupados em um mesmo cluster, o que não é uma tarefa fácil. Dessa forma, existe a preferência, sempre que exista a possibilidade, de se escolher aprendizado supervisionado. Entretanto, em muitos casos do mundo real, o número de exemplos rotulados é muito pequeno, quando não inexistente, e modelos preditivos induzidos a partir de um pequeno conjunto de exemplos rotulados apresentam, geralmente, baixa precisão.

O aprendizado semi-supervisionado, uma recente área de pesquisa em AM, consiste em utilizar algoritmos que aprendem a partir de exemplos rotulados e não rotulados. A grande motivação para esse tipo de aprendizado se dá pelo fato de exemplos não rotulados existirem em abundância e exemplos rotulados serem geralmente escassos. Além disso, a rotulação de exemplos pode ser custosa, como nos casos de indexação de vídeo, categorização de textos e diagnósticos médicos, entre outros. Uma outra motivação é que classificadores induzidos exclusivamente a partir de um pequeno conjunto de exemplos rotulados, geralmente, não apresentam boa precisão. A idéia do aprendizado semi-supervisionado é então utilizar os poucos exemplos rotulados para se obter informações sobre o problema e utilizá-las para guiar o processo de aprendizado a partir dos exemplos não rotulados [4].

Aprendizado semi-supervisionado pode ser utilizado tanto em tarefas de classificação como em tarefas de *clustering* [1]. Em uma classificação semi-supervisionada, a idéia é rotular, com uma certa margem de segurança, alguns dos exemplos no conjunto de exemplos não rotulados, os quais são posteriormente utilizados durante a fase de treinamento do classificador, freqüentemente resultando em uma classificação mais precisa [3]. Já em *clustering* semi-supervisionado, os exemplos rotulados são utilizados no processo de formação dos clusters, servindo geralmente como um conhecimento de fundo, expresso, por exemplo, na forma de restrições, e resultando em melhores clusters.

Vários algoritmos de *clustering* semi-supervisionado têm sido propostos. A grande maioria deles tem como base algum algoritmo existente na literatura, o qual é modificado para tratar exemplos rotulados e não rotulados. Nesses algoritmos, os exemplos rotulados são utilizados para guiar o

[1]DSIN - Gerência de Business Intelligence. CPqD Telecom & IT Solutions. Rod. Campinas—Mogi–Mirim (SP340), km 118, 5. Campinas – São Paulo – Brasil. CEP 13086-902. e-mail msanches@cpqd.com.br.

[2]Instituto de Ciências Matemáticas e de Computação. Universidade de São Paulo. Caixa Postal 668. São Carlos – São Paulo – Brasil. CEP 13560-970. e-mail mcmonard@icmc.usp.br.

processo de formação dos clusters, como nos algoritmos *COP–k-means* [9], *SEEDED–k-means* e *CONSTRAINED–k-means* [1].

Neste trabalho é proposto um outro algoritmo de *clustering* semi-supervisionado, denominado *k-means*$_{ki}$, que pode ser utilizado no processo de rotulação de exemplos, com o intuito de aumentar o conjunto de exemplos rotulados, viabilizando assim o uso de algoritmos supervisionados nesses dados. O algoritmo proposto tem dois requisitos básicos. O primeiro tem a ver com a existência de um pequeno conjunto de exemplos rotulados, muito representativos, rotulados por um especialista, *i.e.* tais exemplos devem representar uma instância de um determinado conceito com elevado nível de certeza. Esse requisito pode ser considerado bastante simples, pois dado a um especialista um conjunto de dados não rotulados, é uma tarefa pouco complexa para o especialista realizar a rotulação de um pequeno conjunto de exemplos protótipos. O segundo requisito, o qual está relacionado com a hipótese induzida por um algoritmo supervisionado, após a rotulação dos exemplos usando *k-means*$_{ki}$, é que a maioria dos exemplos nos clusters sejam da mesma classe.

Este trabalho está organizado da seguinte forma: na Seção 2 é apresentado o algoritmo *k-means*$_{ki}$, fruto do presente trabalho; na Seção 3 são apresentados os resultados experimentais obtidos com o algoritmo e na Seção 4 são apresentadas algumas conclusões a respeito do trabalho realizado.

O Algoritmo *k-means*$_{ki}$

O processo de rotulação aqui proposto está baseado em uma idéia muito simples, composta praticamente de quatro etapas:

1. **Pré-Avaliação**: consiste na indução de um classificador, ou hipótese $\mathbf{h}_1$ inicial, a partir do pequeno conjunto de exemplos rotulados disponível e estimar sua precisão $ca(\mathbf{h}_1)$ utilizando 10–*fold cross–validation*, por exemplo.

2. ***Clustering***: consiste em encontrar clusters utilizando apenas os poucos exemplos rotulados, sendo que exemplos com classes diferentes não poderão ser agrupados em um mesmo cluster. Esse é um ponto muito importante, pois o rótulo dos exemplos não tem significado especial em aprendizado não supervisionado. Clusters em aprendizado não supervisionado e classes em aprendizado supervisionado, não são, necessariamente, equivalentes. A idéia então é utilizar o atributo classe, tal que esses clusters agrupem apenas exemplos com o mesmo valor desse atributo.

3. **Rotulação**: consiste em analisar o conjunto de exemplos não rotulados com o intuito de determinar os exemplos que pertencem aos clusters encontrados, mas com um *alto grau de similaridade*. Uma vez escolhidos esses exemplos, eles serão rotulados com a classe dos exemplos do cluster correspondente, incrementando assim o conjunto de exemplos rotulados.

4. **Pós-Avaliação**: O novo conjunto de exemplos rotulados, *i.e.*, incrementado pelos exemplos rotulados no passo anterior, é fornecido ao mesmo indutor utilizado no passo 1 e uma nova hipótese $\mathbf{h}_2$ é induzida. Essa hipótese $\mathbf{h}_2$ terá então sua precisão $ca(\mathbf{h}_2)$ avaliada e, caso $ca(\mathbf{h}_2) > ca(\mathbf{h}_1)$ é admitido que o processo de rotulação obteve êxito, uma vez que a segunda hipótese teve sua precisão incrementada, e o processo pode ser repetido a partir do passo 2. Caso $ca(\mathbf{h}_1) > ca(\mathbf{h}_2)$, o processo de rotulação pára.

Como uma simples ilustração do processo proposto, considere a Figura 1. Nessa figura os exemplos não rotulados são representados por $\bullet$, e os exemplos rotulados são representados pela própria classe, cujos valores são, nesse caso, $+$ e $*$. Em (a) encontra-se representado o conjunto original de exemplos (rotulados e não rotulados) e em (b) são ilustrados os clusters encontrados utilizando somente os exemplos rotulados. Já em (c) os exemplos não rotulados estão, sempre que possível, alocados em seus respectivos clusters. Em (d) são escolhidos somente os exemplos "mais" similares, os quais receberão o rótulo do atributo classe dos exemplos do cluster correspondente (esses exemplos estão representados por $\bullet$). O resultado final do processo, *i.e.* o novo conjunto de exemplos rotulados, é ilustrado em (e). Os novos exemplos rotulados estão destacados pelas setas.

Como pode ser observado na Figura 1, há a exigência de que os clusters encontrados a partir dos exemplos rotulados *não* agrupem exemplos com classes diferentes. Isso porque os exemplos a serem rotulados receberão a classe dos exemplos que formaram o cluster no qual ele foi inserido. Duas classes diferentes originando um só cluster inviabilizaria o uso do mesmo no processo de rotulação.

Uma vez definido como se dará o processo de rotulação, faz-se necessário escolher o algoritmo a ser utilizado no processo de *clustering*. Para

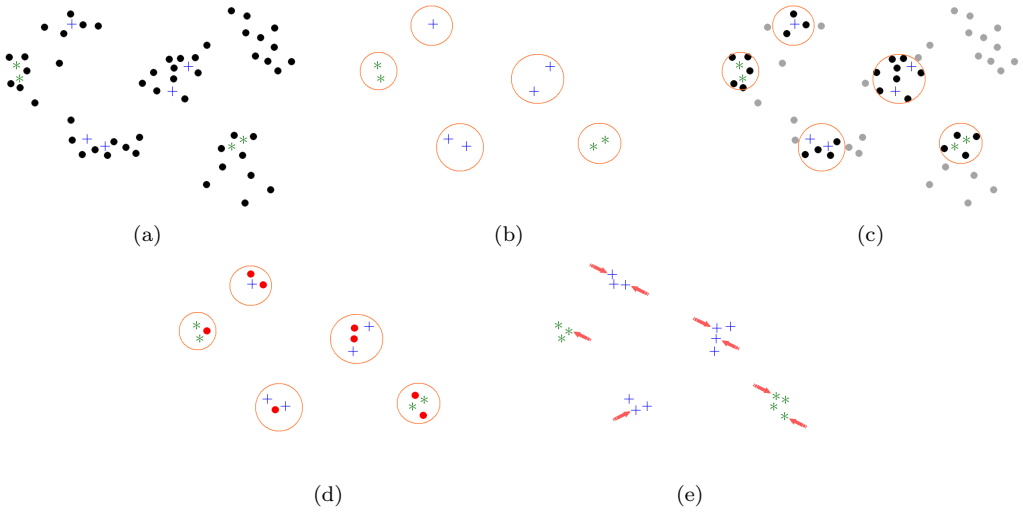

Figura 1: Processo de rotulação proposto: (a) conjunto original de exemplos (b) *clustering* (c) exemplos alocados em clusters (d) escolha dos exemplos mais similares (e) rotulação.

este trabalho optou-se por desenvolver um algoritmo "tipo *k-means*", *i.e.* um algoritmo parecido com o *k-means* [5], porém, com algumas alterações necessárias para se atingir o objetivo proposto.

Esse algoritmo, denominado *k-means*$_{ki}$, tem por objetivo rotular exemplos a partir de um pequeno conjunto de exemplos rotulados, com o intuito de melhorar o processo de classificação. A principal diferença entre os dois algoritmos é encontrada na fase de seleção de centróides. Ao passo que o *k-means* seleciona aleatoriamente seus centróides iniciais, o *k-means*$_{ki}$ define, como centróides iniciais, cada um dos exemplos rotulados disponíveis. Garante-se assim que exemplos rotulados com classes diferentes não serão agrupados em um mesmo cluster. Uma outra diferença é quanto ao processo de *clustering* propriamente dito. Quando o *k-means* é utilizado, cada exemplo é associado ao cluster (centróide) mais próximo. No caso do *k-means*$_{ki}$, é estipulado *a priori* um *threshold* t. Esse *threshold* será o responsável pela associação exemplo/cluster. Ou seja, o exemplo somente poderá ser associado a um dado cluster caso esteja a uma distância menor ou igual a t de seu respectivo centróide.

Uma preocupação constante durante a fase de desenvolvimento do algoritmo, foi a de apenas rotular exemplos com um alto grau de certeza. Assim, ficou estipulado que caso um exemplo esteja a uma distância menor ou igual a t de dois ou mais centróides, ele apenas será rotulado caso os respectivos clusters forem formados por exemplos de mesma classe.

O algoritmo *k-means*$_{ki}$ necessita, quando da sua execução, de três parâmetros, a serem definidos pelo usuário. O primeiro deles é a medida de similaridade a ser utilizada no processo de *clustering*. O segundo parâmetro, já citado, é o valor de t, *i.e.* o valor do *threshold* a ser utilizado no processo de rotulação dos exemplos. O último parâmetro diz respeito às diferentes versões do algoritmo *k-means*$_{ki}$. Tais parâmetros são detalhados a seguir.

Medida de Similaridade: há várias formas de se calcular a similaridade entre exemplos. No algoritmo *k-means*$_{ki}$, a medida de similaridade a ser utilizada na formação dos clusters e na definição do *threshold* t é definida pelo usuário. No presente trabalho foram implementadas as medidas de distância Euclideana e Chebychev. Como as medidas de similaridade foram implementadas em módulos independentes, a inclusão de novas métricas é uma tarefa fácil de ser realizada.

O algoritmo *k-means*$_{ki}$ foi concebido com o intuito de tratar exemplos cujos atributos possuem valores discretos, contínuos e/ou ausentes. Para tanto, todos os atributos contínuos são normalizados no início da execução do algoritmo. Na implementação corrente, para dois atributos discretos $(x_i - y_i)$, é atribuído o valor 1 (um), caso eles possuam valores diferentes, ou 0 (zero), caso sejam iguais. No caso de um valor ausente, é atribuído o valor 1 (um).

***Threshold*:** um fator importante no algoritmo *k-means*$_{ki}$ é a definição do valor do *threshold*. Com certeza, definir um valor que represente

um alto grau de similaridade não é uma tarefa simples. Essa definição torna-se ainda mais complicada quando da rotulação de exemplos a serem utilizados posteriormente por um algoritmo de aprendizado supervisionado, uma vez que rotular erroneamente um exemplo pode ter conseqüências indesejáveis. No algoritmo *k-means*$_{ki}$, o *threshold* é estipulado pelo próprio usuário como um parâmetro de entrada. Assim, quanto maior for o valor de t, maior a probabilidade de se rotular erroneamente um exemplo.

O usuário não estipula um valor absoluto para t, mas sim relativo. O valor absoluto do *threshold*, aqui denominado $t^{'}$, é calculado pelo *k-means*$_{ki}$ baseado em um vetor ordenado v contendo todas as distâncias entre todos os centróides (exemplos rotulados) e todos os exemplos não rotulados. Por exemplo, caso o usuário estabeleça o valor 5% para t, significa que serão passíveis de rotulação todos aqueles exemplos que estiverem a uma distância d de um ou mais centróides, se e somente se d estiver entre as 5% menores distâncias, *i.e.*, as 5% primeiras posições de v. Denominando o vetor formado pelas 5% primeiras posições do vetor ordenado v por $v^{'}$, o valor absoluto do *threshold*, denominado $t^{'}$, é dado pelo valor contido na última posição de $v^{'}$. Assim, serão então passíveis de rotulação todos aqueles exemplos que estiverem a uma distância d de algum centróide, tal que $d \leq t^{'}$. Nesse caso, para receber efetivamente um rótulo, basta que o exemplo não pertença simultaneamente a clusters formados por exemplos com classes diferentes.

Versões Adicionalmente, foram estudadas possíveis variações do algoritmo *k-means*$_{ki}$, o que resultou em duas versões do mesmo. Tais versões possuem sutis diferenças, dentre as quais pode-se citar:

- **Versão 1:** é a versão mais simples, na qual os centróides são calculados utilizando somente os exemplos iniciais rotulados pelo especialista. Nesse caso, o algoritmo *k-means*$_{ki}$ é executado somente uma vez, já que os novos exemplos rotulados não influenciam o processo de rotulação;
- **Versão 2:** difere da primeira versão pelo fato de que os centróides são atualizados a cada iteração. Nessa versão, os novos exemplos rotulados passam a influenciar no processo de rotulação. O perigo, nesse caso, é a existência de um rótulo errado, que pode acarretar em rótulos errados para exemplos a serem rotulados em cada iteração;

As versões 1 e 2 do *k-means*$_{ki}$ são descritas pelo Algoritmo 1. Deve ser observado que o algoritmo *k-means*$_{ki}$, independentemente da versão considerada, possui um *bias* muito característico. Ele considera que exemplos que expressam o mesmo conceito, *i.e.* possuem a mesma classe, estão próximos, segundo a medida de similaridade utilizada na sua execução. Sendo assim, o seu uso é aconselhável em conjuntos de dados que possuem tal característica.

Algoritmo 1 *k-means*$_{ki}$

Require: E $=\{(\mathbf{x}_i, y_i), i = 1, ..., N\}$: conjunto de exemplos rotulados; F $=\{\mathbf{a}_i, i = 1, ..., M\}$: conjunto de exemplos não rotulados; d: medida de distância a ser utilizada (Euclideana, Manhattan, etc); t: *threshold* (distância mínima para rotular um exemplo); v: versão do algoritmo; $MD[N][M]$: matriz de distâncias entre os exemplos de E e F, onde $MD[i][j]$ especifica a distância entre $\mathbf{x}_i \in E$ e $\mathbf{a}_j \in F$;
procedure *k-means*$_{ki}$ (E, F, d, t, v, MD)

```
1:  R = ∅: conjunto de exemplos rotulados pelo algoritmo;
2:  escolha, como centróides iniciais dos clusters C_1, C_2, ..., C_N, os exemplos x_1, x_2, ..., x_N, respectivamente;
3:  repeat
4:    for all a_i ∈ F do
5:      L_clusters : lista ordenada com todos os clusters a uma distância d de a_i, tal que d ≤ t;
6:      if todos os clusters em L_clusters tiverem sido originados por exemplos rotulados com o mesmo rótulo r then
7:        associe a_i ao cluster C_j mais próximo, tal que C_j ∈ L_clusters;
8:      end if
9:    end for
10:   if v = 2 then
11:     atualize os centróides dos clusters que tiveram algum novo exemplo associado;
12:     atualize MD;
13:   end if
14: until não ocorra nenhuma nova associação exemplo/cluster;
15: for all C_i do
16:   for all a_j associado a C_i do
17:     Rotule a_j com a classe do exemplo que originou o cluster C_i;
18:     R = R ∪ a_j;
19:   end for
20: end for
    return R;
```

Experimentos e Análise dos Resultados

Nesta seção são descritos os experimentos realizados utilizando o conjunto de dados *Breast-Cancer*, obtido do repositório da UCI [6]. Experimentos semelhantes utilizando outros conjuntos de dados encontram-se em [7]. Os experimentos têm os seguintes dois objetivos:

1. analisar a validade dos rótulos encontrados pelo algoritmo *k-means*$_{ki}$;
2. analisar a utilidade dos novos exemplos rotulados na indução de um classificador.

Para o primeiro objetivo, o conjunto de dados *Breast-Cancer*, que possui apenas exemplos rotulados, é particionado em dois subconjuntos. Um grande, cujos exemplos têm sua classe extraída, e outro pequeno, cujos exemplos mantém o rótulo. Assim, simula-se a situação para a qual o *k-means*$_{ki}$ foi concebido.

Após submeter esses dois subconjuntos ao *k-means*$_{ki}$, a saída do algoritmo, *i.e.* o novo conjunto de exemplos rotulados, é analisada. Essa análise é uma tarefa simples, e consiste da comparação do rótulo atribuído pelo algoritmo *k-means*$_{ki}$ a um dado exemplo com o seu rótulo original, que é conhecido. Nas Tabelas 1, 3 e 5 são ilustrados os resultados obtidos. Nela, as colunas ilustram a medida de similaridade utilizada, o *threshold* estipulado, a versão do algoritmo utilizada, bem como o total de novos exemplos rotulados, a percentagem aproximada de novos exemplos rotulados, considerando o número total de exemplos não rotulados, o número de exemplos rotulados erroneamente e a percentagem aproximada de exemplos rotulados erroneamente, considerando o número médio de exemplos rotulados pelo algoritmo (**Total**). Foram utilizados 34, 69 e 169 exemplos, respectivamente, para o conjunto de exemplos rotulados e o restante (665, 630 e 530 exemplos respectivamente) para o conjunto de exemplos não rotulados. Como os exemplos rotulados são escolhidos aleatoriamente, cada experimento foi repetido 5 (cinco) vezes. Assim, os resultados apresentados nas Tabelas 1, 3 e 5 representam a média e o desvio-padrão obtidos a partir das 5 (cinco) repetições. Foram utilizados também diferentes valores para a medida de similaridade — Euclideana e Chebychev — , para o *threshold* — 5%, 10% e 20% — e para a versão do algoritmo — versão 1 (v1) e versão 2 (v2).

Já nas tabelas 2, 4 e 6 é mostrado o erro dos classificadores induzidos a partir do pequeno conjunto de exemplos inicialmente rotulados (**Pré-Rotulação**) e a partir desse mesmo conjunto adicionado dos exemplos rotulados pelo algoritmo *k-means*$_{ki}$ (**Pós-Rotulação**). Foi utilizado para medir o erro **Pré-Rotulação** o mesmo conjunto de teste utilizado para medir o erro **Pós-Rotulação**. Em todos os casos, os números entre parênteses referem-se ao desvio padrão.

Pode ser visto que a percentagem de exemplos rotulados pelo algoritmo *k-means*$_{ki}$ (**% Rotulados**) é, na maioria dos casos, maior quando a versão 2 é utilizada. Pode também ser observado que a percentagem de exemplos erroneamente rotulados pelo algoritmo (**% Errados**) é, quando não nula, sempre maior para a versão 2. Também pode ser notado que conforme o *threshold* aumenta, a percentagem de exemplos rotulados pelo algoritmo também aumenta.

Pode-se notar também que a percentagem de exemplos rotulados erroneamente é sempre superior para a distância Chebychev. Isso mostra, como esperado, a influência da função de distância utilizada. Ainda assim, em ambos os casos, o número de exemplos rotulados erroneamente é muito baixo, mostrando a potencialidade do algoritmo proposto.

Entretanto, em uma situação de uso real do algoritmo, não são conhecidos os rótulos dos exemplos não rotulados. Nesse caso, inicialmente, o erro dos classificadores induzidos com os exemplos rotulados deve ser estimado, utilizando, por exemplo, *k–fold cross–validation* e, após a aplicação do algoritmo, os novos exemplos rotulados devem ser adicionados ao conjunto de treinamento. O erro dos classificadores induzidos com esse novo conjunto de treinamento deve ser estimado e, se for o caso, o processo deve ser repetido. Na Tabela 7 são mostrados os resultados obtidos realizando essa experiência real.

Nessa experiência foram escolhidos, para atuarem como conjunto de exemplos inicialmente rotulados, 69 ($\sim$ 10%) e 139 ($\sim$ 20%) exemplos do conjunto *Breast-Cancer*. Nessa experiência foi utilizado 10–*fold cross-validation*, e os resultados obtidos são ilustrados na coluna **Pré-Rotulação** da Tabela 7.

O próximo passo da experiência consistiu na execução do *k-means*$_{ki}$ utilizando como conjunto de exemplos inicialmente rotulados os 69, e 139 exemplos utilizados na indução do classificador. Como conjunto de exemplos não rotulados foi utilizado o resto dos exemplos, *i.e.* 630 e 560 exemplos respectivamente, visto que o conjunto *Breast-Cancer* possui 699 exemplos.

Para o primeiro caso — 69 exemplos no conjunto de exemplos inicialmente rotulados — o *k-means*$_{ki}$ foi executado utilizando um *threshold* de 20% com

Distância	*Threshold*	Versão	Rotulados			
			Total	**% Rotulados**	**Errados**	**% Errados**
Euclideana	5%	v1	301.8 (8.1)	45.4	0	0
		v2	305.6 (9.7)	45.9	0	0
	10%	v1	352.2 (19.0)	53.0	0	0
		v2	358.4 (20.6)	53.9	0	0
	20%	v1	389.4 (12.9)	58.5	0.4 (0.5)	0.1
		v2	395.8 (12.4)	59.5	0.4 (0.5)	0.1
Chebychev	5%	v1	337.2 (6.8)	50.7	0	0
		v2	337.2 (6.8)	50.7	0	0
	10%	v1	395.0 (36.8)	59.4	3.2 (2.2)	0.8
		v2	403.0 (40.0)	60.6	4.2 (3.0)	1.0
	20%	v1	410.8 (10.7)	61.8	4.0 (1.2)	1.0
		v2	419.2 (5.8)	63.0	5.2 (1.9)	1.2

Tabela 1: *Breast–Cancer* — Resultados com 34 ($\sim 5\%$) exemplos rotulados

Erro						
			Pós-Rotulação		**Pré-Rotulação**	
Distância	*Threshold*	**Versão**	$\mathcal{C}4.5$rules	$\mathcal{CN}2$	$\mathcal{C}4.5$rules	$\mathcal{CN}2$
Euclideana	5%	v1	13.1 (0.7)	13.5 (2.1)	17.5 (1.0)	17.6 (2.4)
		v2	13.3 (0.8)	13.6 (2.2)	17.6 (1.1)	17.8 (2.4)
	10%	v1	16.1 (0.9)	20.5 (3.4)	18.9 (1.4)	20.3 (3.2)
		v2	17.0 (1.0)	20.8 (3.4)	19.0 (1.4)	20.7 (3.3)
	20%	v1	20.5 (2.2)	20.2 (2.1)	18.4 (2.5)	22.0 (3.8)
		v2	23.2 (2.9)	24.3 (4.3)	18.3 (2.7)	22.3 (3.9)
Chebychev	5%	v1	17.6 (1.1)	14.6 (2.5)	18.3 (1.4)	19.4 (2.5)
		v2	17.6 (1.1)	14.6 (2.5)	18.3 (1.4)	19.4 (2.5)
	10%	v1	18.6 (3.5)	20.9 (5.2)	17.1 (3.0)	22.4 (4.3)
		v2	20.1 (2.8)	21.3 (4.7)	17.3 (2.8)	22.1 (4.1)
	20%	v1	17.4 (3.6)	21.9 (5.0)	17.8 (3.2)	22.9 (4.2)
		v2	18.8 (3.0)	22.3 (4.5)	17.9 (2.9)	22.6 (3.9)

Tabela 2: *Breast–Cancer* — Performance dos classificadores antes (com 34 exemplos rotulados) e depois da rotulação

Distância	*Threshold*	Versão	Rotulados			
			Total	**% Rotulados**	**Errados**	**% Errados**
Euclideana	5%	v1	313.6 (6.1)	49.8	0	0
		v2	317.6 (5.5)	50.4	0	0
	10%	v1	357.2 (2.8)	56.7	0	0
		v2	363.8 (3.7)	57.7	0	0
	20%	v1	383.6 (2.7)	60.9	0.4 (0.5)	0.1
		v2	387.2 (2.2)	61.5	0.8 (0.8)	0.2
Chebychev	5%	v1	347.2 (3.8)	59.4	0	0
		v2	347.2 (3.8)	59.4	0	0
	10%	v1	405.2 (13.4)	64.3	5.2 (1.5)	1.3
		v2	414.2 (4.9)	65.7	6.2 (1.1)	1.5
	20%	v1	405.2 (13.4)	64.3	5.2 (1.5)	1.3
		v2	414.2 (4.9)	65.7	6.2 (1.1)	1.5

Tabela 3: *Breast–Cancer* — Resultados com 69 ($\sim 10\%$) exemplos rotulados

a versão 2 do algoritmo. Já no segundo caso — 139 exemplos no conjunto de exemplos inicialmente rotulados — foi utilizado um *threshold* de 5% com a versão 1 do algoritmo. Para ambos os casos foram utilizadas as medidas Chebychev e Euclideana.

Após a execução do *k-means*$_{ki}$, para analisar os rótulos encontrados, foram induzidos classificadores utilizando o conjunto de exemplos inicialmente rotulados adicionado do conjunto de exemplos rotulados pelo algoritmo. Esses classificadores também tiveram sua precisão estimada utilizando 10–*fold cross–validation*, a qual é mostrada na coluna **Pós-Rotulação** da Tabela 7.

Na Tabela 8 é mostrado o número de exemplos rotulados pelo algoritmo em cada um dos casos, bem como a percentagem de exemplos rotulados con-

Erro						
Pós-Rotulação					Pré-Rotulação	
Distância	*Threshold*	**Versão**	$\mathcal{C}$4.5rules	$\mathcal{CN}2$	$\mathcal{C}$4.5rules	$\mathcal{CN}2$
Euclideana	5%	v1	12.8 (1.6)	17.2 (2.2)	13.5 (0.4)	16.5 (1.7)
		v2	13.0 (1.5)	17.4 (2.2)	13.6 (0.3)	16.7 (1.7)
	10%	v1	17.6 (3.0)	18.0 (2.5)	15.0 (0.6)	19.2 (1.9)
		v2	18.8 (3.3)	18.7 (2.5)	15.2 (0.7)	19.6 (2.0)
	20%	v1	18.7 (3.1)	20.6 (2.7)	15.0 (0.9)	21.0 (2.3)
		v2	18.9 (3.2)	21.9 (2.5)	14.8 (1.0)	21.1 (2.3)
Chebychev	5%	v1	18.1 (3.0)	19.0 (2.4)	14.5 (0.5)	18.4 (1.8)
		v2	18.1 (3.0)	19.0 (2.4)	14.5 (0.5)	18.4 (1.8)
	10%	v1	14.0 (0.9)	21.9 (1.4)	13.9 (0.8)	19.7 (2.6)
		v2	14.1 (0.8)	20.1 (2.7)	13.8 (0.7)	19.7 (2.7)
	20%	v1	14.0 (0.9)	21.9 (1.4)	13.9 (0.8)	19.7 (2.7)
		v2	14.1 (0.8)	20.1 (2.7)	13.8 (0.7)	19.7 (2.7)

Tabela 4: *Breast–Cancer* — Performance dos classificadores antes (com 69 exemplos rotulados) e depois da rotulação

			Rotulados			
Distância	*Threshold*	**Versão**	**Total**	% **Rotulados**	**Errados**	% **Errados**
Euclideana	5%	v1	293.2 (9.4)	52.3	0	0
		v2	295.0 (9.9)	52.7	0	0
	10%	v1	325.6 (4.8)	58.1	0	0
		v2	329.4 (5.0)	58.8	0	0
	20%	v1	348.4 (7.1)	62.2	2.0 (0.7)	0.6
		v2	352.6 (6.9)	63.0	3.6 (1.1)	1.0
Chebychev	5%	v1	317.0 (4.9)	56.6	0.8 (0.8)	0.2
		v2	317.0 (4.9)	56.6	0.8 (0.8)	0.2
	10%	v1	387.2 (7.3)	69.1	7.4 (1.5)	1.9
		v2	392.8 (6.4)	70.1	8.0 (1.7)	2.0
	20%	v1	387.2 (7.3)	69.1	7.4 (1.5)	1.9
		v2	392.8 (6.4)	70.1	8.0 (1.7)	2.0

Tabela 5: *Breast–Cancer* — Resultados com 139 ($\sim 20\%$) exemplos rotulados

Erro						
Pós-Rotulação					Pré-Rotulação	
Distância	*Threshold*	**Versão**	$\mathcal{C}$4.5rules	$\mathcal{CN}2$	$\mathcal{C}$4.5rules	$\mathcal{CN}2$
Euclideana	5%	v1	11.0 (0.9)	13.9 (1.5)	11.2 (1.1)	14.7 (1.2)
		v2	11.4 (1.2)	14.5 (1.2)	11.2 (1.1)	14.7 (1.2)
	10%	v1	12.4 (0.8)	18.1 (1.8)	11.8 (0.9)	16.7 (1.5)
		v2	13.7 (1.1)	18.1 (2.1)	11.7 (0.8)	17.0 (1.5)
	20%	v1	15.1 (1.2)	19.3 (2.2)	11.5 (0.9)	17.7 (1.7)
		v2	12.3 (1.3)	20.7 (2.0)	11.1 (0.9)	17.4 (1.8)
Chebychev	5%	v1	11.1 (0.4)	15.4 (1.4)	11.4 (0.9)	15.8 (1.3)
		v2	11.1 (0.4)	15.4 (1.4)	11.4 (0.9)	15.8 (1.3)
	10%	v1	12.8 (0.6)	15.6 (0.8)	12.0 (0.9)	16.4 (1.0)
		v2	12.8 (0.6)	16.0 (1.3)	12.3 (0.9)	16.8 (1.0)
	20%	v1	12.8 (0.6)	15.6 (0.8)	12.0 (0.9)	16.4 (1.0)
		v2	12.8 (0.6)	16.0 (1.3)	12.3 (0.9)	16.8 (1.0)

Tabela 6: *Breast–Cancer* — Performance dos classificadores antes (com 139 exemplos rotulados) e depois da rotulação

siderando, para cada caso, o total de exemplos não rotulados.

Por exemplo, utilizando a distância Euclideana com 69 exemplos inicialmente rotulados, o algoritmo rotulou 389 exemplos, os quais representam 61.7% dos 630 exemplos (*i.e.* $699 - 69$) não rotulados. Como pode ser observado, em três dos quatro casos nenhum exemplo foi erroneamente rotulado pelo $k\text{-}means_{ki}$.

Na Tabela 7 pode-se verificar que o erro dos classificadores induzidos na fase **Pós-Rotulação** é bem menor que os daqueles induzidos na fase **Pré-**

Erro							
Parâmetros				Pós-Rotulação		Pré-Rotulação	
Distância	**Rotulados**	*Threshold*	**Versão**	$\mathcal{C}4.5$rules	$\mathcal{CN}2$	$\mathcal{C}4.5$rules	$\mathcal{CN}2$
Euclideana	69	20%	v2	1.5 (0.5)	2.4 (0.7)	7.1 (3.2)	8.6 (3.8)
Chebychev	69	20%	v2	1.0 (0.5)	1.8 (0.6)		
Euclideana	139	5%	v1	3.3 (0.8)	3.3 (0.9)	10.8 (2.5)	8.6 (3.0)
Chebychev	139	5%	v1	3.5 (0.7)	3.2 (0.5)		

Tabela 7: *Breast-Cancer 2* — Performance dos classificadores antes e depois da rotulação, utilizando 10–*fold cross–validation*

Erro						
Parâmetros						
Distância	**Rotulados**	*Threshold*	**Versão**	**Total Rotulados**	**% Total**	**Errados**
Euclideana	69	20%	v2	389	61.7	0
Chebychev	69	20%	v2	419	66.5	8
Euclideana	139	5%	v1	309	55.2	0
Chebychev	139	5%	v1	325	58.0	0

Tabela 8: *Breast-Cancer 2* — Número de exemplos rotulados pelo k-$means_{ki}$

Rotulação, para ambos os indutores utilizados no experimento. Assim, o k-$means_{ki}$ estaria apto a ser executado novamente, uma vez que os exemplos por ele rotulados foram capazes de aumentar a precisão do classificador. Nesse caso, o conjunto de exemplos rotulados pelo k-$means_{ki}$ adicionado dos exemplos inicialmente rotulados seria, nessa nova iteração do algoritmo, fornecido ao k-$means_{ki}$ como sendo o conjunto inicial de exemplos rotulados, e apenas os exemplos não rotulados pelo algoritmo na iteração anterior pertenceriam ao conjunto dos exemplos não rotulados.

Conclusões

A qualidade dos exemplos inicialmente rotulados é de suma importância para o bom funcionamento do algoritmo proposto. O algoritmo k-$means_{ki}$ apóia-se em duas premissas, ambas relacionadas com o conjunto de dados utilizado. Uma delas diz que os exemplos rotulados devem estar localizados perto do centro do agrupamento ao qual eles pertencem.

A medida de distância utilizada, principalmente quando da utilização de conjuntos de dados heterogêneos (com atributos discretos e contínuos), influi, drasticamente no processo de rotulação. Em alguns experimentos preliminares com conjuntos de dados heterogêneos utilizando a distância Chebychev, o algoritmo k-$means_{ki}$ não rotulou nenhum exemplo, visto que essa distância não se mostrou apropriada para esse caso [7]. Assim, ao se escolher a medida de similaridade a ser utilizada no processo de rotulação deve-se levar em conta os dados utilizados.

Já o valor do *threshold* deve ser utilizado com cautela. A utilização de um alto *threshold* acarretará na formação de grandes clusters. Por conseguinte, aumenta-se a possibilidade de rotular um exemplo erroneamente, visto que quanto maior o *threshold*, mais longe o exemplo não rotulado pode estar do centro do cluster para ser rotulado. O k-$means_{ki}$ utiliza restrições para rotular um exemplo, o que não necessariamente diminuirá a percentagem de exemplos rotulados erroneamente. Além disso, tais restrições somente são válidas quando há uma intersecção de clusters originados por exemplos de classes diferentes. Caso contrário, o aumento do *threshold*, certamente acarretará no aumento do número de exemplos rotulados erroneamente.

Em relação às duas versões propostas do algoritmo k-$means_{ki}$, pode-se notar que elas comportam-se de forma semelhante. Pode ser comprovado pelas experiências realizadas que, na grande maioria dos casos, a versão 2 do k-$means_{ki}$ rotulou mais exemplos que a versão 1. Pode ser visto também que o erro no processo de rotulação foi maior para a versão 2. Conclui-se então que ao se escolher a versão a ser utilizada deve-se ponderar entre um número menor, mas com maior precisão, de exemplos rotulados — versão 1 — ou mais exemplos rotulados, com a possibilidade de aumentar o erro — versão 2.

Como mencionado anteriormente, a medida de distância tem uma forte influência, especialmente para conjuntos de dados heterogêneos. Uma forma de lidar com esses conjuntos de dados é usar uma função de distância heterogênea que utiliza funções de distância diferente para diferentes tipos de atributos [2]. Como trabalhos futuros, pretende-se implementar a métrica

Value Difference Metric – VDM [8], é a *Heterogeneous Distance Function* – HVDM [10], as quais provem uma função de distância apropriada para atributos discretos. Também, após implementadas essas medidas, pretendemos comparar o $k\text{-}means_{ki}$ com outros algoritmos semelhantes de aprendizado de *clustering* semi-supervisionado propostos.

Agradecimentos O desenvolvimento deste trabalho contou com o auxílio financeiro da CAPES – Coordenação de Aperfeiçoamento de Pessoal de Nível Superior.

Referências

[1] Sugato Basu, Arindam Banerjee, and Raymond Mooney. Semi-supervised clustering by seeding. In *Proceedings of the Nineteenth International Conference on Machine Learning — ICML-2002*, pages 19–26, Sidney, Australia, 2002.

[2] Gustavo E. A. P. A. Batista. Pré-processamento de Dados em Aprendizado de Máquina Supervisionado, 2003. Tese de Doutorado, ICMC-USP, `http://www.icmc.usp.br/~gbatista`.

[3] Avrim Blum and Tom M. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory — COLT-98*, pages 92–100, New York, USA, 1998. ACM Press.

[4] Rebecca Bruce. A bayesian approach to semi-supervised learning. In Mitsuru Ishizuka and Abdul Satter, editors, *Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium — NLPRS-2001*, pages 57–64, Tokyo, Japan, 2001. Springer Verlag.

[5] J. Macqueen. Some methods for classification and analysis of multivariate observation. In *Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, Berkeley, 1967. University of California Press.

[6] C. J. Merz and P. M. Murphy. UCI repository of machine learning datasets, 1998. University of California, Irvine, CA, USA. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[7] Marcelo Kaminski Sanches. Aprendizado de máquina semi-supervisionado: proposta de um algoritmo para rotular exemplos a partir de poucos exemplos rotulados, 2003. Dissertação de Mestrado, ICMC-USP, `http://www.teses.usp.br/teses/disponiveis/55/55134/tde-12102003-140536`.

[8] C. Stanfill and D. Waltz. Instance-based learning algorithms. *Communications of the ACM*, 12:1213–1228, 1986.

[9] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schroedl. Constrained *k-means* clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning — ICML-2001*, pages 577–584, Williamstown, MA, USA, 2001.

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