Aprendizado Semi-Supervisionado Baseado em Grafos

Renato Fabbri
Universidade de São Paulo, São Carlos, São Paulo, Brazil
renato.fabbri@gmail.com
Junho de 2010

Resumo

Neste trabalho abordamos em especial dois métodos de aprendizado semi-supervisionado que se baseiam em grafos. Primeiramente apresentamos brevemente o assunto de aprendizado semi-supervisionado. Depois expomos, também brevemente, o que vem a ser grafos e redes complexas. Na sequência explicitamos os dois métodos que julgamos representativos do aprendizado semi-supervisionado com grafos, a saber o método de propagação de rótulos e o método de mincut. Finalizamos com algumas palavras sobre os dois métodos e suas utilidades e pontos fracos.

1 Introdução

1.1 Aprendizado Semi-Supervisionado

O *aprendizado semi-supervisionado* (ASS) visa o aprendizado a partir da utilização tanto dos objetos rotulados quanto dos não rotulados. São exemplos de ASS a *classificação semi-supervisionada* (CSS) e a *clusterização forçada*. Este trabalho é um apanhado sobre CSS em grafos que é como geralmente é entendido o ASS com grafos.

O nome "semi-supervisionado"vem do fato de que os dados utilizados para o aprendizado está entre os dados utilizados no aprendizado supervisionado e não supervisionado[1].

Podemos comparar a CSS ao processo de uma criança que entende o que é um animal através de alguns exemplos dados pelos adultos e de outros vários que ela mesma observa. Ou seja, para o aprendizado do que é (e o que não é) um animal, ela utiliza tanto o conhecimento sobre os animais que contaram para ela que é um animal quanto o conhecimento sobre outros seres que não falaram para ela o que é.

Um grande atrativo dos métodos de ASS é que ele pode ajudar a contornar o fato de que os objetos rotulados podem ser custosos. Comumente necessitam de um humano especializado para rotulá-los, outras vezes necessitam de equipamentos, condições ou processamentos especiais. Muitas vezes é simplesmente impossível obter novos objetos rotulados. Já os objetos não rotulados são geralmente imediatos e abundantes, podendo ser utilizados pelo ASS de imediato.

Uma distinção importante que se faz é quanto ao ASS *indutivo* e o ASS *transdutivo*. O ASS indutivo se assemelha ao aprendizado transdutivo. A diferença entre ambos é que o aprendizado transdutivo é otimizado para tratar somente os objetos utilizados para o treinamento, enquanto o ASS indutivo trata de objetos não presentes na fase de treinamento.

Dados não rotulados podem ajudar de diversas formas, em especial quando as diferentes classes se mostram clusterizadas. Na Figura 1 é explicitado esquematicamente a forma com que os dados não rotulados podem auxiliar a determinar uma fronteira de decisão mais pertinente. Embora este seja o comportamento natural mais comum, devemos notar que o contrário também ocorre e que neste caso o princípio da clusterização das classes semelhantes pode nos levar a resulta-

[1] No aprendizado *supervisionado* são utilizados somente dados rotulados. No aprendizado *não supervisionado* são utilizados somente dados não rotulados.

dos enganosos. A Figura 2 demonstra este caso no qual o critério de clusterização levaria a uma pior classificação.

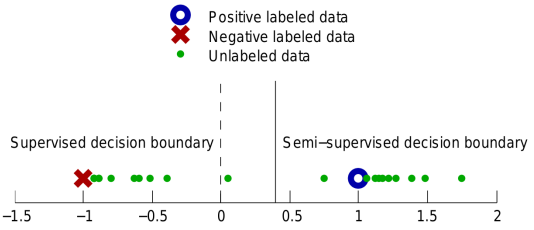

Figura 1: Dados não-rotulados auxiliando a criar uma melhor fronteira de decisão. Fonte: (Zhu, X. and Lafferty, J. and Rosenfeld, R., 2005).

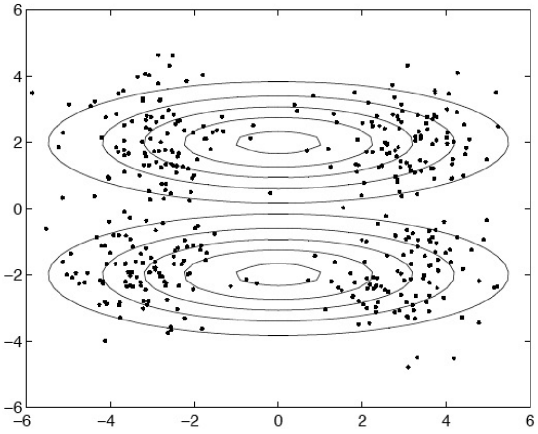

Figura 2: Dados não-rotulados atrapalhariam a correta separação das classes. Fonte: (Müller, K. R. and Zien, A., 2008).

1.1.1 Breve Histórico do Aprendizado Semi-Supervisionado

Os primeiros trabalhos em CSS assumiam duas classes, cada uma com uma distribuição normal. Desta forma, os dados como um todo resultam em um *modelo de mistura* (*mixture model*). São necessárias somente de duas amostras rotuladas, uma de cada classe, para determinar completamente o modelo de mistura.

Uma variante é o *self-training*: o classificador é treinado com os dados rotulados e depois aplicado aos não rotulados. Os pontos não rotulados que foram rotulados de forma mais confiante, são então adicionados ao conjunto de dados rotulados e então o classificador é treinado novamente. Este procedimento é chamado também de *self-teaching* e *bootstrapping*.

Ambos estes métodos são utilizados há muito tempo e continuam comuns devido à sua simplicidade conceitual e algorítmica.

Métodos mais recentes incluem o *co-training* (em que dois classificadores utilizam partes disjuntas dos atributos de cada objeto e utiliza os novos objetos rotulados um do outro) e *SVM transdutivo*.

Na última década, o aprendizado semi-supervisionado baseado em grafos atraiu grande atenção. Estes métodos iniciam-se com grafos cujos nós são os objetos rotulados e não rotulados e cujas arestas tem pesos que refletem a similaridade entre os objetos. Assume-se que objetos conectados por arestas de peso alto tendem a ter o mesmo rótulo e que os rótulos podem propagar-se pelo grafo. A seguir daremos uma definição e introdução mais formal sobre os grafos e sobre redes complexas.

1.2 Grafos e Redes Complexas

Está fora do escopo deste trabalho prolongar-se com a definição de grafos e redes complexas. Nesta parte é dado uma breve introdução e histórico sobre o tema. O estudo dos grafos é matéria da *teoria dos grafos*.

Dito de forma direta, um grafo é uma representação abstrata de um conjunto de objetos no qual alguns destes objetos são ligados. Tais objetos são geralmente chamados de *nós* ou *vértices*. As ligações entre vértices são comumente chamadas de *arestas*. Tipicamente, um grafo é visualmente representado por um conjunto de pontos ou bolas (os vértices) ligados por linhas (arestas).

Embora uma grande parte da teoria dos grafos seja elaborada apenas sobre esta estrutura básica descrita acima, existem diversos possíveis acréscimos estruturais. Provavelmente o mais importante é o de especificar uma direção para as arestas, resultando em *arestas direcionadas*. A segunda carac-

terística diferencial de alguns grafos é a existência grandezas numéricas associadas às arestas, que resultam nas chamadas *arestas com pesos*.

Outras características dos grafos são: mais de um tipo de aresta (possibilitando mais de uma aresta entre cada par de vértices); pesos associados aos nós; atributos diversos associados às arestas e aos nós (p.ex. cor); arestas incompletas; qualquer mistura de todas as características mencionadas acima.

Alguns grafos são clássicos, citamos alguns deles. Grafo regular é aquele no qual cada vértice tem o mesmo número de arestas. O grafo completo possui aresta entre qualquer par de vértices e também é chamado de *click*. Um grafo conectado é aquele que apresenta um caminho de sucessão de arestas entre qualquer par de vértices. Grafo randômico é um grafo construído com uma probabilidade de se ter uma aresta entre dois vértices. Similarmente, um grafo randômico é aquele que exibe *as mesmas características* que um grafo construído desta forma.

Formalmente falando, um grafo é um par ordenado $G = (V, E)$ composto de um conjunto de vértices V e um conjunto E de arestas, que são subconjuntos de V com dois elementos. Estes são os grafos simples (com somente uma aresta possível entre cada par de vértices e sem aresta entre um vértice e ele mesmo) e não direcionado. Caso o conjunto E seja composto de pares ordenados de elementos de V, o grafo é direcionado.

A medida mais comum de um grafo é o *grau*, que é a quantidade de arestas ligadas a um vértice. Caso o grafo seja direcionado, podemos dividir o grau em duas quantidades: grau de entrada e grau de saída de um vértice. Caso o grafo tenha arestas com peso, pode-se falar em *força* de um vértice que é a soma dos pesos das arestas ligadas ao vértice. Similarmente, podemos num grafo direcionado com peso, falar em força de entrada e força de saída de um vértice.

Antes de continuar com as medidas de um grafo, para o nosso caso é pertinente citar o que é uma *rede complexa*. Dito de forma direta, uma rede complexa é um grafo com uma grande quantidade de vértices e arestas. Muitos pequisadores definem as redes complexas como sendo grafos de grandes dimensões que exibem características não triviais, isto é, que diferem de um grafo regular ou randômico. Esta definição é invalidada pelo fato de que se fala em rede complexa randômica e em rede complexa regular. Pode-se sim entender que a área de estudo das redes complexas se interessa pelos casos de redes complexas (comumente) encontrados na natureza que exibem características não triviais.

Outra diferença entre as áreas de grafo e redes complexas é que enquanto a área de grafos tradicionalmente é vinculada à extração de propriedades matemáticas de grafos em geral ou com construções particulares, a área de redes complexas é marcada mais acentuadamente pela observação das redes complexas que representam fenômenos humanos ou naturais e suas divergências se comparadas às redes mais triviais (regulares e randômicas). Exemplos destas características peculiares são as conhecidas *lei livre de escala* e *lei de pequeno mundo*.

1.2.1 Breve Histórico da Teoria dos Grafos e das Redes Complexas

A estrutura de pontos e retas é tão simples que é difícil precisar quando foi utilizada pela primeira vez, se é que isso é possível. O artigo escrito por Leonhard Euler chamado "Konigsberg Bridge problem"é geralmente descrito como o primeiro trabalho tanto de topologia quanto de teoria dos grafos.

Mais de um século depois, Cayley estudou *árvores* através de formas analíticas que surgiam do cálculo diferencial. Tais estudos tiveram grandes consequências para a química teórica e esta fusão entre química e grafos está no centro da nomenclatura comum utilizada para grafos.

Em particular, o nome *grafo* foi introduzido por Sylvester em 1878 através de um artigo publicado na Nature.

No século XX a teoria dos grafos teve focos diferentes, passando pelo problema de

coloração de grafos, topologia (em particular em conexão com a álgebra, p.ex. *Lei de Kirchoff* para circuitos), e, por fim, por teorias probabilísticas.

Já adentrando o terreno das chamadas redes complexas, os artigos que demonstravam as características *livre de escala* e de *pequeno mundo* para grafos grandes representando estruturas da natureza (Ferrer i Cancho and Sole, 2001; Albert and Barabasi, 2002) marcaram este novo grande interesse pela área.

Atualmente, as redes complexas são aplicadas com sucesso a diversas áreas (Costa et al., 2008), o que inclui neurociências (Sporns, 2002), física (Gfeller, 2007), linguística (Dorogovtsev and Mendes, 2001) e ciência da computação (Moura et al., 2003), para citar simplesmente alguns casos.

2 Aprendizado Semi-Supervisionado em Grafos

Diversas técnicas de CSS baseada em grafos tem sido desenvolvidas e utilizadas. Citamos os seguintes métodos, de forma ilustrativa e sem a pretensão de obtermos uma lista completa: algorítmo de regularização (Belkin, M. and Matveeva, I. and Niyogi, P., 2004), mincut (Blum, A. and Chawla, S., 2001), propagação de rótulo (Zhu, X. and Ghahramani, Z., 2003), modelos de mistura e maximização de entropia (Zhu, X. and Lafferty, J. and Rosenfeld, R., 2005).

De especial importância é a forma de construção do grafo. Uma discussão detalhada sobre o assunto é encontrada no terceiro capítulo de (Zhu, X. and Lafferty, J. and Rosenfeld, R., 2005).

O resto desta seção é dedicado a explorar dois métodos e CSS com grafos, a saber a propagação de rótulo e o mincut.

2.1 Propagação de Rótulo

Estudamos aqui a propagação de rótulo em um grafo, onde o rótulo de um nó propaga aos nós vizinhos de acordo com as proximidades. Os rótulos dos dados rotulados são fixos, desta forma funcionam como fontes que emanam rótulos pelos dados não rotulados.

2.1.1 Convenções e Exposição do Problema

Sejam $(x_1, y_1)...(x_l, y_l)$ os dados rotulados, $y \in 1...C$ com C o número de classes e $x_{l+1}...x_{l+u}$ os dados não rotulados. Geralmente $l \ll u$. Seja $n = l + u$.

Note que L e U denotam os dados rotulados e não rotulados respectivamente.

É importante que as classes C sejam conhecidas e que estejam presentes, cada uma, nos dados rotulados.

Este é um problema transdutivo, no qual devemos associar as classes dadas a cada um dos dados não rotulados.

Criamos um grafo em que cada ponto é um dado, seja ele rotulado ou não rotulado. Para a criação das arestas fixamos o fato de que queremos dados similares com o mesmo rótulo. Já foram pensadas algumas formas diferentes de se fazer isso, a mais difundida assume um grafo totalmente conectado (há aresta entre qualquer par de nós). Cada aresta tem um peso, que reflete a similaridade entre os dados. Este peso pode ser calculado assim:

$$w_{ij} = exp(-\frac{\|x_i - x_j\|^2}{\alpha^2}) \qquad (1)$$

Onde α é um *hiperparâmetro* regulador que deve ser testado em diferentes valores.

Os rótulos são propagados através das arestas. Pesos maiores permitem que os rótulos sejam propagados mais facilmente. Definimos a matriz probabilística de transição da seguinte forma:

$$P_{ij} = P(i \to j) = \frac{w_{ij}}{\sum_{k=1}^{n} w_{ik}} \qquad (2)$$

Onde P_{ij} é a probabilidade de transição do nó i para o nó j (passagem do rótulo). Também definimos a matriz de rótulos Y_L, $l \times C$, cuja *iésima* linha possui 1 na coluna correspondente à classe do dado x_i.

A tarefa consiste em computar os rótulos dos nós todos (na verdade somente dos não rotulados). Seja f uma matriz $n \times C$ em que cada linha pode ser interpretada como uma distribuição de probabilidade sobre os rótulos. Agora podemos seguir para o algoritmo (a inicialização de f não é importante).

2.1.2 O Algoritmo de Propagação de Rótulo

O algoritmo de propagação de rótulo é o seguinte:

1. Propague $f \leftarrow Pf$.
2. Mantenha os dados rotulados iniciais $f_L = Y_L$.
3. Repita do passo 1 até que o algoritmo convirja.

No passo primeiro, todos os nós propagam o seu rótulo para os seus vizinhos. O segundo passo trata de assegurar que os rótulos dos nós inicialmente rotulados não sejam sobrescritos.

2.1.3 Convergência

O resultado interessante deste método é que ele não necessita do processo iterativo. Satisfeitas as condições de que todas as classes estejam presentes e de que cada componente conectado possua ao menos um dado rotulado, a convergência é segura.

Seja $f = \binom{f_L}{f_U}$. Como f_L está fixo em Y_L, estamos somente interessados em f_U. Nós partimos P em sub-matrizes rotuladas e não rotuladas:

$$\begin{bmatrix} P_{LL} & P_{LU} \\ P_{UL} & P_{UU} \end{bmatrix} \tag{3}$$

Pode-se entender intuitivamente (e pode-se mostrar matematicamente) que este algorítmo é:

$$f_U \leftarrow P_{UU} f_U + P_{UL} Y_L \tag{4}$$

O que nos leva a:

$$f_u = \lim_{n\to\infty} [(P_{UU})^n f_U^0 + (\sum_{i=1}^{n} (P_{UU})^{(i-1)}) P_{UL} Y_L] \tag{5}$$

onde f_U^0 é o valor inicial de f_U.

Mostraremos agora que, no primeiro termo, $(P_{UU})^n f_U^0 \to 0$. Lembrando que as linhas de P estão normalizadas e P_{UU} é uma sub-matriz de P, temos:

$$\exists\, \gamma < 1 : \sum_{j=1}^{u} (P_{UU})_{ij} \leq \gamma, \forall i = 1...u \tag{6}$$

Portanto:

$$\sum_j (P_{UU})_{ij}^n = \sum_j \sum_k (P_{UU})_{ik}^{(n-1)} (P_{UU})_{kj} \tag{7}$$

$$= \sum_k (P_{UU})_{ik}^{(n-1)} \sum_j (P_{UU})_{kj} \tag{8}$$

$$\leq \sum_k (P_{UU})_{ik}^{(n-1)} \gamma \tag{9}$$

$$\leq \gamma^n \tag{10}$$

Podemos observar que a soma das linhas de $(P_{UU})^n$ converge para zero, o que significa que $(P_{UU})^n f_U^0 \to 0$. Note que o valor inicial f_U^0 não faz diferença.

Voltando à nossa condição inicial explicitada em (4) percebemos que temos o ponto fixo:

$$f_u = (I - P_{UU})^{-1} P_{UL} Y_L \tag{11}$$

Esta equação é, portanto, o único ponto fixo e a solução dada pelo nosso algoritmo iterativo. Isso nos permite resolver a questão de propagação de rótulo diretamente sem a propagação iterativa.

Note que a solução é valida somente quando $I - P_{UU}$ é inversível. Esta condição é satisfeita intuitivamente quando cada componente conectado no grafo possui ao menos um ponto rotulado.

2.1.4 Exemplos Ilustrativos

Aqui demonstramos as propriedades do algorítmo de propagação de rótulos com dois bancos de dados artificiais. A Figura 1 mostra um banco de dados sintético com três bandas horizontais. Os pontos são uniformemente distribuidos das bandas. Existem 3 pontos rotulados (um para cada banda) e 178 dados não rotulados. Os dados foram rotulados utilizando-se *k-nearest-neighbors* (com

$k = 1$) e o algoritmo de propagação de rótulos em grafos apresentado nesta monografia.

Importante aqui é observar que o *KNN* ignora os dados não rotulados e portanto a estrutura em bandas dos mesmos. Isso pode ser visto na Figura 2 em que parte dos dados de bandas diferentes assumem o rótulo do dados rotulado mais próximo, mas não o rótulo do dado rotulado que se encontra na mesma banda.

Isso é completamente resolvido no caso do algorítmo de propagação de rótulo. Foi utilizado $\alpha = 0.22$.

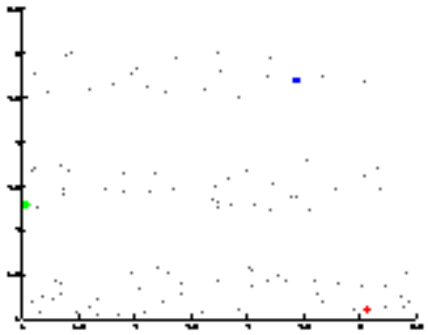

Figura 3: Dados originais, 3 rotulados, 178 não rotulados.

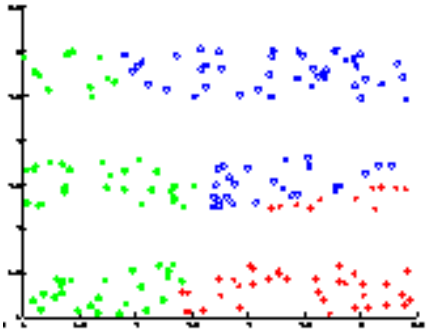

Figura 4: Dados rotulados utilizando-se knn.

Já o o nosso segundo exemplo utiliza um banco de dados, também sintético, com duas classes espiraladas. O banco original, os 184 dados não rotulados e os 2 dados rotulados podem ser vistos na Figura 4.

Nas Figuras 6, 7 e 8 temos o mesmo comportamento do exemplo anterior. O *KNN* não leva em consideração a distribuição dos dados, portanto as espirais acabam tendo elementos rotulados das duas maneiras. Já com a utilização da propagação de rótulos, cada espiral fica com sua própria classe.

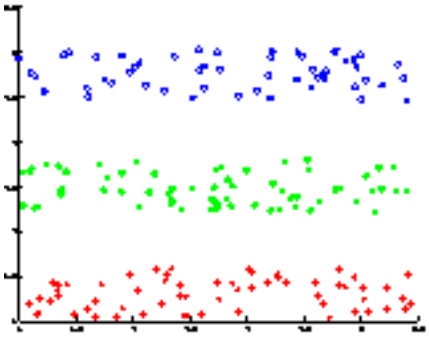

Figura 5: Dados rotulados utilizando propagação de rótulos.

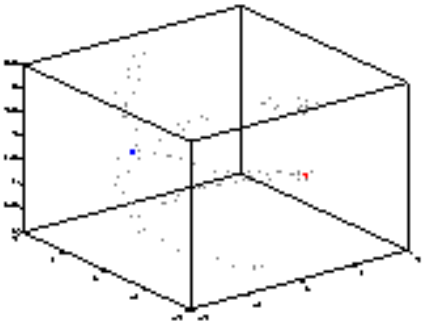

Figura 6: Dados originais, 2 rotulados, 184 não rotulados.

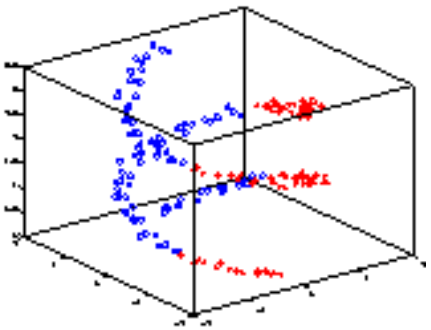

Figura 7: Dados espiralados rotulados utilizando-se knn.

2.2 Mincut

Como a maior parte dos métodos de aprendizado semi-supervisionado, a idéia geral do método *mincut* é associar rótulos aos dados

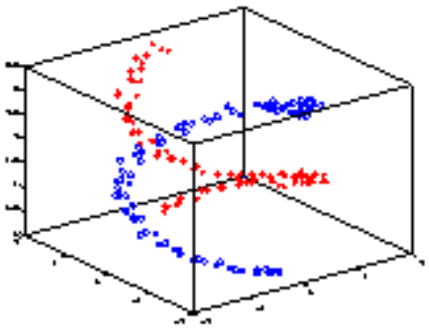

Figura 8: Dados espiralados rotulados utilizando propagação de rótulos.

não rotulados de forma a otimizar uma função objetivo. Para o caso do mincut, as funções objetivo são limitadas aos casos em que dependem somente das relações entre pares de nós.

O que torna este método especialmente atraente é a complexidade computacional que, para os casos em que o mincut pode ser aplicado, é polinomial e resulta no ótimo global. Portanto, mesmo com a perda da generalidade com relação ao EM (*Expectation Maximization*) ou com o hill-climbing (que podem ser aplicados praticamente em qualquer caso), nos interessa a vantagem de calcular de fato o ótimo.

Neste caso temos a seguinte questão: quão interessantes são os casos em que podemos aplicar o mincut?

Vale notar que na literatura existe a seguinte relação: o mincut se relaciona aos algorítmos de vizinhos mais próximos assim como o SVM transdutivo se relaciona ao SVM padrão.

2.2.1 Convenções e Exposição do Problema

São dados os conjuntos de dados rotulados L e não rotulados U. Sem perda de generalidade, assumimos que estamos interessados em uma classificação binária e denotamos por L_+ o conjunto de dados positivos em L e por L_- o conjunto de dados negativos em L. Os vértices em L_+ e L_- serão denotados respectivamente v_+ e v_- e serão chamados *vértices de classificação*. Todos os outrso vértices serão chamados *vértices de exemplo* ou simplesmente *exemplos*.

2.2.2 Algoritmo Mincut

O algorítmo segue os seguintes passos:

1. Construímos um grafo com pesos $G = (V, E)$, onde $V = L \cup U$ e $E \subseteq V \times V$. A cada aresta $e \in E$ é associado um peso $w(e)$.[2]

2. Os vértices de classificação são conectados aos vértices de mesma classe com peso infinito. Ou seja, $w(v_1, v_2) = \infty, \forall \ v_1, v_2 \in L_+$ e $w(v_1, v_2) = \infty, \forall \ v_1, v_2 \in L_-$.

3. Às arestas entre os vértices de exemplo são dados pesos com base em alguma relação entre os exemplos, tal qual a similaridade ou a distância entre eles. A função utilizada para atribuir peso às arestas dos exemplos é chamada (ao menos neste compêndio) de *função de atribuição de pesos* e será denotado por w.

4. Determinamos o conjunto de arestas com a menor soma de pesos que, se removidas, separam todos os v_+ dos v_-. Esta remoção é chamada de *corte mínimo* do grafo e dá o nome ao método. Este corte mínimo parte o grafo em duas partes, às quais denominamos V_+ e V_- nos quais todos os $v_+ \in V_+$ e todos os $v_- \in V_-$. Por completude, convencionamos que caso existam mais de um corte mínimo possível, escollhemos o que resulta em um V_+ menor.

5. Por fim, rotulamos como positivos todos os vértices em V_+ e como negativos todos os vértives em V_-.

A motivação principal para este algorítmo é que a dois vértices similares é associada

[2]Note que esta parte de montagem do problema não foi tida como parte do algoritmo no caso da propagação de rótulos, mas sim como parte das convenções e exposição do problema. No caso do mincut essa atribuição de pesos é bem menos resolvida.

uma aresta de peso alto e, portanto, estes provavelmente estarão no mesmo subconjunto de vértices. Isso é completamente concordante com a premissa básica da maioria dos algorítmimos de classificação: dados similares devem ter classes similares.

Isso nos leva à função de atribuição de pesos w. Uma primeira e natural maneira de obtermos uma função de atribuição de pesos é através da distância euclidiana entre os objetos. Vértices com pequena distância entre si possuiriam grande peso e vértices com grande distância teriam pequeno peso. No caso de não termos diretamente como calcular esta distância euclidiana (p.ex. no caso de atributos não numéricos), então uma possibilidade é utilizar um algoritmo de aprendizado auxiliar que nos retorne as distâncias. O crucial aqui é o fato de que a forma de atribuição de pesos inflencia diretamente e fortemente o resultado e do algorítmo, assim como a sua qualidade.

2.2.3 Calibragem

O crítico neste algorítmo é sem dúvida a função de atribuição de pesos. Pode-se, inclusive, atribuir peso zero para a maior parte das arestas, selecionando somente as arestas de maior interesse para cada dado.

Das inumeráveis maneiras de se atribuir pesos, listamos a seguir duas formas principais de se formar as arestas, cada uma com as suas vantagens:

1. **Mincut-3** conectamos cada vértice a seu vizinho rotulado mais próximo e ao seus dois vizinhos mais próximos que não são o vizinho antes escolhido. Esta conexão forçada com o rotulado mais próximo tem o intuito de evitar a formação de componentes isolados.

2. **Mincut-δ** utilizando-se da métrica do espaço das variáveis, se a distância entre dois dados for menos que um dado limiar denotado por δ, conectamos os vértices correspondentes por uma aresta. A melhor escolha de δ pode variar bastante dependendo das intensões e do banco de dados. Segue algumas alternativas:

   - No **Mincut-δ_0**, escolhemos o δ máximo para o qual o grafo tem um corte de valor zero.
   - No **Mincut-$\delta_{\frac{1}{2}}$**, utilizamos um valor para δ em que o tamanho do maior componente conectado no grafo possui metade dos dados.
   - No **Mincut-δ_{opt}**, escolhemos o valor de δ que apresenta o menor erro. Este δ é estabelecido depois de testes e serve como um valor de referência para o estudo do desempenho de outros valores de δ.

2.2.4 Discussão

Os resultados obtidos pela utilização do mincut são bastante relacionados aos obtidos com a utilização do *kNN*[3].

Este algorítmo também não está livre de problemas. Alguns podem ser notados com clareza. Pode-se ter problemas caso tenhamos pouquíssimos exemplos rotulados (ou muito localizados) e muitos exemplos não rotulados. Neste caso o algorítmo pode separar quase a totalidade dos dados para uma só classe. Por exemplo, se temos somente um dado positivo e outro negativo e conectamos cada vértice aos seus três vizinhos mais próximos com arestas de peso 1, o corte em que temos somente o vértice posivo na classe dos positivos tem custo 3.

Outro problema em potencial pode surgir no caso em que o grafo é esparço o suficiente para ter um certo número de componentes não conectados[4]. O algorítmo de mincut deverá então escolher arbitrariamente qual rótulo associar a todo o compontente. Desta forma podemos entender que a função de atribuição de peso deve ser escolhida cuidadosamente de forma a não permitir que isso aconteça.

3 Conclusão

A utilização de dados não rotulados para CSS é de grande utilidade quando os dados rotulados são poucos ou custosos. Também pode-se

[3] Resultado levantado pela utilização do LOOCV (*leave-one-out cross-validation*.

[4] Por exemplo, no caso de se utilizar kNN, uma parte dos dados pode criar uma componente isolada.

DATASET	$\|L\|\&\|U\|$	NUMBER OF FEATURES	MINCUT				ID3	3-NN
			MINCUT-3	MINCUT-δ_{opt}	MINCUT-δ_0	MINCUT-$\delta_{1/2}$		
MUSH	20+1000	22	82.1	**97.7**	**97.7**	97.0	93.3	91.1
MUSH*	20+1000	22	74.2	**88.7**	56.9	**87.0**	80.8	83.3
TAE	10+100	5	86.0	**99.0**	96.0	**97.0**	86.0	80.0
TAE*	10+100	5	76.0	**96.0**	86.0	**94.0**	76.0	62.0
VOTING	45+390	16	89.1	**91.3**	66.1	83.3	86.4	**89.6**
MUSK	40+200	166	73.0	**92.5**	91.0	**92.5**	83.5	87.0
PIMA	50+718	8	63.8	**72.3**	48.8	**72.3**	70.0	68.1
IONO	50+300	34	71.0	81.6	78.0	77.6	**88.6**	69.6
BUPA	45+300	6	53.3	**59.3**	48.0	41.7	**55.3**	52.7
MI	124+432	6	70.0	64.4	64.4	64.4	**98.6**	81.1
MII	169+432	6	**68.6**	67.2	57.2	67.2	67.9	63.6
MIII*	122+432	6	79.1	80.6	64.8	80.6	**94.4**	83.6

Figura 9: Mincut com alguns valores para δ e comparação com outros métodos.

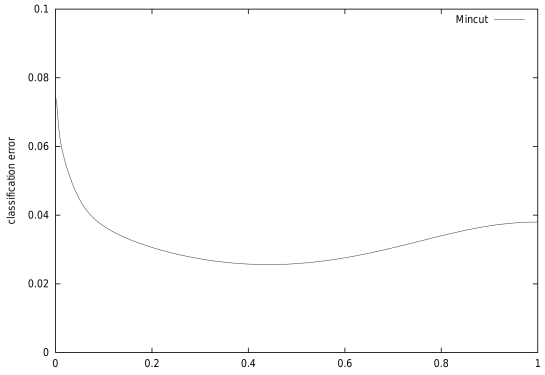

Figura 10: Variação do erro com o Delta.

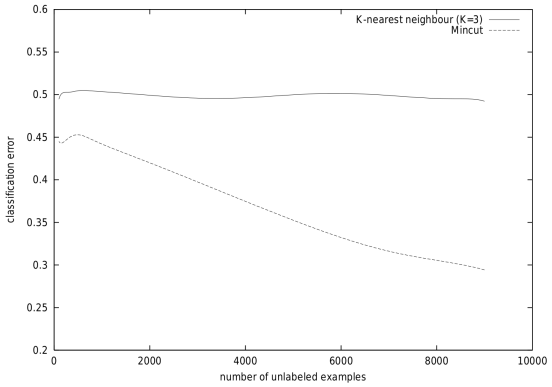

Figura 12: Comparação do erro entre mincut e kNN com a variação de exemplos não rotulados δ_{opt}.

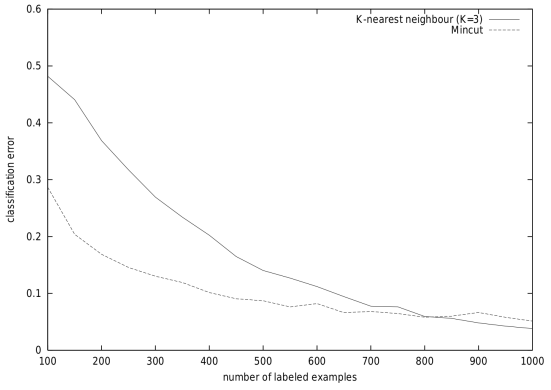

Figura 11: Comparação do erro entre mincut e kNN com a variação de exemplos rotulados δ_{opt}.

utilizar com segurança CSS quando as classes estão clusterizadas entre si.

O método de propagação de rótulo assegura uma classificação que segue a distribuição das amostras, não somente a proximidade de um dado a outro rotulado, como no caso do kNN. Já o método do mincut apresenta a dificuldade de seleção do método de interconexão dos dados, i.e. da distribuição das arestas. Mesmo assim, segundo os resultados que temos, ele apresenta melhor performance que o tradicional kNN para a maioria dos casos. Isso se deve também à utilização da estrutura dos dados para a classificação geral.

Referências

Zhu, X. and Lafferty, J. and Rosenfeld, R. 2005. Semi-supervised learning with graphs. *Unpublished doc-*

toral dissertation.

Zhu, X. 2006. Semi-supervised learning literature survey. *Citeseer*.

Zhu, X. and Ghahramani, Z. 2002. Learning from labeled and unlabeled data with label propagation. *Citeseer*.

Blum, A. and Chawla, S. 2001. Learning from labeled and unlabeled data using graph mincuts.

Belkin, M. and Matveeva, I. and Niyogi, P. 2004. Regularization and Semi-supervised Learning on Large Graphs.

Müller, K. R. and Zien, A. 2008. Semi-Supervised Learning.

Chapelle, O. and Sch
"olkopf, B. and Zien, A. 2006. Semi-supervised learning. *Citeseer*.

R. Ferrer i Cancho and R. V. Sole. 2001. The small world of human language. *Proceedings of the Royal Society of London B*, 268:2261.

M. E. J. Newman. 2003. The Structure and Function of Complex Networks. *SIAM Review*, 45:167–256.

R. Z. Albert and A.L. Barabasi. 2002. Statistical Mechanics of Complex Networks. *Rev. Modern Phys.*, 74:47–97.

L. F. da Costa, O. N. Oliveira Jr., G. Travieso, F. A. Rodrigues, P. R. Villas Boas, L. Antiqueira, M. P. Viana, L. E. C. da Rocha. 2008. Analyzing and Modeling Real-World Phenomena with Complex Networks: A Survey of Applications. *arXiv* 0711.3199.

O. Sporns. 2002. Network analysis, complexity, and brain function. *Complexity*, 8(1):56–60.

D. Gfeller, P. LosRios, A. Caflisch and F. Rao. 2007. Complex network analysis of free-energy landscapes. *Proceedings of the National Academy of Science USA*, 104 (6):1817–1822

S. N. Dorogovtsev and J. F. F.Mendes. 2001. Language as an evolving word web. *Proceedings of the Royal Society of London B*, 268:2603.

A. P. S. de Moura, Y. C. Lai and A. E. Motter. 2003. Signatures of small-world and scale-free properties in large computer programs. *Physical Review E*, 68(1):017102.

Belkin, M. and Matveeva, I. and Niyogi, P.. 2004. Regularization and semi-supervised learning on large graphs. *Learning theory, Springer*, 624–638.