Implementação Eficiente do *Shared Nearest Neighbour* em Dados Espaciais

Bruno Filipe Faustino[1], João Moura-Pires[1] e Maribel Yasmina Santos[2]

[1] CENTRIA, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa
b.faustino@campus.fct.unl.pt, jmp@fct.unl.pt
[2] Centro de Investigação Algoritmi, Universidade do Minho
maribel@dsi.uminho.pt

Resumo A taxa de colecta de dados espaciais está a aumentar e os algoritmos de agrupamento tornam-se cada vez mais populares, pois não necessitam de informação *a priori*. Contudo, estes algoritmos requerem um tempo de execução significativo e várias corridas para alcançar os melhores resultados. O *Shared Nearest Neighbour* (*SNN*) é um algoritmo de agrupamento cuja complexidade temporal no pior caso é $O(n^2)$, comprometendo a sua escalabilidade. Neste artigo, conjuga-se o *SNN* com estruturas de dados métricas que dão suporte à procura dos K vizinhos mais próximos, permitindo melhorar a sua complexidade temporal no caso esperado para $O(n \times log(n))$, com conjuntos de dados espaciais. Propomos, ainda, uma estratégia de reaproveitamento entre corridas do cálculo dos K vizinhos mais próximos, atingindo a complexidade de $O(n)$. Através dos resultados experimentais, que avaliam a escalabilidade desta solução e a comparam com uma versão original do *SNN*, são obtidos ganhos muito significativos.

Palavras-chave: dados espaciais, *kd-tree*, *shared nearest neighbour*

1 Introdução

Actualmente, a taxa de colecta de dados espaciais está a aumentar [7], tornando-se importante conseguir extrair informação dos mesmos. Os algoritmos de agrupamento representam uma técnica não supervisionada de extracção de informação, que não requer qualquer tipo de informação *a priori* sobre um conjunto de dados.

Neste artigo, é abordado o problema da escalabilidade de algoritmos de agrupamento que necessitam de obter os K vizinhos mais próximos, para que estes sejam mais eficientes, nomeadamente quando lidam com dados espaciais. Para tal, foi considerado o *Shared Nearest Neighbour* (*SNN*) [2], não só pelos resultados obtidos em [6] com dados espaciais, mas também por demonstrar melhores resultados [5] que o *DBSCAN*, já que este último não consegue identificar grupos de densidade superior, sem comprometer os de densidade inferior e vice-versa [2].

O *SNN*, introduzido na literatura por Jarvis e Patrick [4], usa o número de vizinhos partilhados entre objectos, como uma medida de semelhança. Mais tarde, é introduzida uma medida de densidade [2], com o intuito de resolver

problemas relacionados com o ruído nos conjuntos de dados e com a variação da densidade, forma e tamanho dos grupos identificados. Esta abordagem é aqui referenciada como *SNN* Original.

Ao longo do tempo surgiram novas abordagens ao *SNN* [5]. Independentemente da abordagem seguida, o *SNN* tem as suas limitações, sobretudo com conjuntos de dados de grande dimensão. Isto deve-se à complexidade do *SNN* ser $O(n^2)$ no pior caso [2], onde n representa o número de objectos de um conjunto de dados em análise. Esta complexidade é observada nos resultados experimentais apresentados com dados espaciais, demonstrando que a mesma se verifica no domínio da baixa dimensionalidade.

Com a motivação de resolver esta limitação, surge mais uma variante do *SNN*, o *Shared Nearest Neighbour Algorithm with Enclosures* (*SNNAE*) [1]. Os autores do *SNNAE* afirmam que este fica mais escalável, mais eficiente e que obtém melhores resultados de agrupamento. No entanto. a complexidade no pior caso permanece $O(n^2)$, não apresentam complexidade no caso esperado e os resultados experimentais não são convincentes, pois assentam num conjunto de dados com 209 objectos.

O objectivo desde trabalho é melhorar a escalabilidade do *SNN* sobre dados espaciais, conjugando-o com estruturas de dados métricas, para suportar a procura dos K vizinhos mais próximos e reaproveitar esta procura entre corridas consecutivas.

O resto do artigo está organizado da seguinte forma: na secção 2 é discutida a complexidade desta variante; na secção 3 são apresentados os resultados obtidos com um conjunto de dados do Twitter; e por fim, na secção 4 são apresentadas as conclusões e as direcções que vão ser tomadas no que toca ao trabalho futuro.

2 Uso da *kd-Tree* no *SNN*

A complexidade no pior caso do *SNN* Original é $O(n^2)$ e é igualmente observada experimentalmente, como mostraremos mais à frente, na implementação de referência do *SNN* Original [2]. O principal factor que determina esta complexidade é a consulta dos K vizinhos mais próximos. Em [2] é feita uma breve referência à utilização de estruturas de dados métricas, de forma a reduzir a complexidade.

Neste artigo, de forma a avaliar a eficiência obtida na procura pelos K vizinhos mais próximos de um objecto, foi escolhida a *kd-Tree*, por ser popular na literatura, por ter a sua complexidade avaliada e por ter uma implementação em Java. Esta abordagem será referenciada ao longo do artigo como *kd-SNN*.

Com a introdução das estruturas de dados métricas no *SNN*, é conveniente considerar três tipos de corridas do algoritmo: (i) a corrida inicial, em que é necessário construir a estrutura de dados, consultar e guardar os K vizinhos mais próximos de cada objecto; (ii) a corrida em que já se tem a estrutura de dados construída, mas é preciso consultar e guardar os K vizinhos mais próximos; (iii) a corrida em que já se tem a estrutura de dados construída e os K vizinhos

mais próximos já calculados, por exemplo, o K da nova corrida é menor ou igual que o da corrida anterior.

A primeira operação do *SNN* está encarregue de realizar consultas à estrutura de dados métrica, de forma a obter os K vizinhos de cada objecto. Estes são guardados e ordenados ascendentemente pela distância. A complexidade deste método depende do esforço requerido em obter os K vizinhos. Segundo [3], o custo de construir a árvore é $O(d \times n \times log(n))$, em que d denota a dimensionalidade de um conjunto de dados, e o custo esperado da pesquisa dos K vizinhos de um dado objecto é proporcional a $O(log(n))$. Assim, a complexidade no caso esperado de construir a árvore e de determinar os K vizinhos de cada objecto é proporcional a $O(n \times log(n))$.

As restantes operações do *SNN* têm uma complexidade proporcional a $O(n)$, sendo que a constante de proporcionalidade depende do valor de K. Assim, podemos inferir que o *kd-SNN* tem uma complexidade, no caso esperado, que depende da corrida (Tabela 1).

Tabela 1: Complexidades para as Várias Corridas do *SNN*

Corrida	Complexidade
(i) Com Construção e Com Obtenção dos K Vizinhos	$O(n \times log(n))$
(ii) Sem Construção, mas Com Obtenção dos K Vizinhos	$O(n \times log(n))$
(iii) Sem Construção e Sem Obtenção dos K Vizinhos	$O(n)$

3 Resultados Experimentais

O conjunto de dados Twitter utilizado é constituído por 512.000 objectos, onde cada um é definido por uma coordenada geográfica, que representa o local de onde foi realizado o *tweet*, e pela data/hora do *tweet*. Na procura pelos K vizinhos foi usada a função euclidiana.

O tempo de execução (figura 1, com $K = 6$) é decomposto pelas três corridas definidas na tabela 1. O impacto da construção da *kd-Tree* é observável, comparando as corridas (i) e (ii). O peso que a obtenção dos K vizinhos tem, é perceptível, comparando as corridas (ii) e (iii). Na figura 2a, evidencia-se uma clara melhoria do *kd-SNN*. Na figura 2b, apresenta-se a razão, $\frac{Tempo(s)\ do\ SNN}{Tempo(s)\ do\ kd-SNN}$, revelando que, o tempo de execução do *SNN* Original é cerca de 370 vezes superior ao do *kd-SNN* (com 128.000 objectos). Não foram realizados estudos comparativos com o *SNNAE*, devido ao código não ser público e às limitações já referidas.

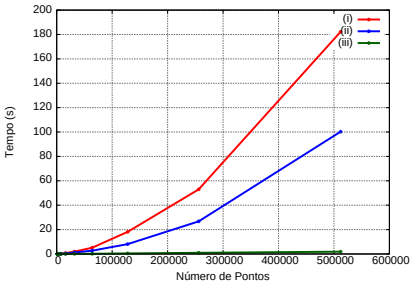
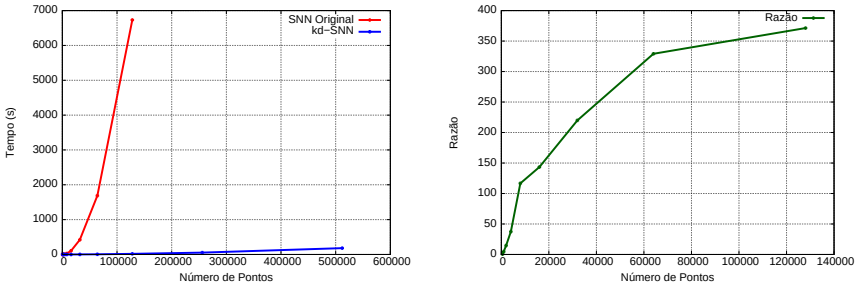


Figura 1: Tempo de Execução (Twitter)



(a) Tempo (b) Razão

Figura 2: Comparação dos Tempos de Execução do *SNN* Original e do *kd-SNN*

4 Conclusões

Este artigo propõe uma implementação do *SNN*, usando uma *kd-Tree*, o que permite melhorar a complexidade no caso esperado para $O(n \times log(n))$ ou $O(n)$, dependendo da corrida. Os resultados experimentais mostram que, com os K vizinhos previamente calculados, os tempos de execução são dramaticamente inferiores. Este facto induz uma estratégia sobre o espaço de parâmetros, começando com valores de K mais altos. Isto pode potenciar algum trabalho futuro: (1) soluções em que o *SNN* faz auto *tuning* de parâmetros, se estivermos na presença de critérios de avaliação do agrupamento resultante; (2) soluções interactivas, onde o utilizador pode alterar os parâmetros, obtendo em tempo útil o resultado de agrupamento. Adicionalmente, prevemos estender este trabalho para estruturas de dados métricas em memória secundária.

Referências

1. Bhavsar, H.B., Jivani, A.G.: The shared nearest neighbor algorithm with enclosures (SNNAE). In: 2009 WRI World Congress on Computer Science and Information Engineering. vol. 4, pp. 436–442. IEEE (Apr 2009)
2. Ertöz, L., Steinbach, M., Kumar, V.: Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In Proceedings Of Second SIAM International Conference On Data Mining (2003)
3. Friedman, J.H., Bentley, J.L., Finkel, R.A.: An algorithm for finding best matches in logarithmic expected time. ACM Trans. Math. Softw. 3(3), 209–226 (Sep 1977)
4. Jarvis, R.A., Patrick, E.A.: Clustering using a similarity measure based on shared near neighbors. IEEE Transactions on Computers C-22(11), 1025– 1034 (Nov 1973)
5. Moreira, A., Santos, M.Y., Carneiro, S.: Density-based clustering algorithms – DBSCAN and SNN (Jul 2005), `http://get.dsi.uminho.pt/local/`
6. Santos, M.Y., Silva, J.P., Moura-Pires, J., Wachowicz, M.: Automated traffic route identification through the shared nearest neighbour algorithm. In: Bridging the Geographic Information Sciences, pp. 231–248 (2012)
7. Yang, C., Goodchild, M., Huang, Q., Nebert, D., Raskin, R., Xu, Y., Bambacus, M., Fay, D.: Spatial cloud computing: how can the geospatial sciences use and help shape cloud computing? International Journal of Digital Earth 4(4), 305–329 (2011)