Um Projeto de Pesquisa para Acelerar o Algoritmo *k*-NN

Gabriel Bernardo da Silva[1], Osvaldo Luiz de Oliveira[1]

[1]Centro Universitário Campo Limpo Paulista (UNIFACCAMP)
Rua Guatemala, 167, Jardim América – 13.231-230 – Campo Limpo Paulista – SP – Brasil

gabriel7798@gmail.com, osvaldo@faccamp.br

Abstract. *The k-nearest neighbor (k-NN) algorithm has been widely used in pattern recognition, case-based reasoning, data mining, and many other Machine Learning applications, to search in a dataset of $n \geq 1$ instances of $m \geq 1$ dimensions, $k \geq 1$ instances more similar to an instance given as the algorithm input. The k-NN search is time consuming in datasets composed with large number of instances or with high dimension instances. Usually the similarity function used in the k-NN algorithm is a metric and this allows the triangular inequality concept to be used to develop more efficient k-NN algorithm variants. This article reports a work in progress in which an algorithm is proposed to accelerate k-NN using the triangular inequality concept.*

Resumo. *O algoritmo k-NN – k-nearest neighbor – tem sido amplamente utilizado em reconhecimento de padrões, raciocínio baseado em casos, mineração de dados e muitas outras aplicações de Aprendizado de Máquina, para buscar em um conjunto de $n \geq 1$ instâncias de $m \geq 1$ dimensões, $k \geq 1$ instâncias mais similares a um instância dada como entrada do algoritmo. Tal busca consome muito tempo em conjunto de instâncias com grande número de instâncias e/ou dimensionalidade. Normalmente a função de similaridade utilizada no algoritmo k-NN é uma métrica e, sendo uma métrica, o conceito de desigualdade triangular pode ser utilizado para desenvolver variantes mais eficientes do algoritmo k-NN. Este artigo relata um trabalho em andamento que propõe um algoritmo para acelerar o k-NN utilizando o conceito de desigualdade triangular.*

1. Introdução

O algoritmo *k*-NN (Cover & Hart, 1967) – do inglês *k-nearest neighbor* – é frequentemente listado como um dos 10 algoritmos mais famosos da área de Aprendizado de Máquina e tem sido amplamente utilizado em reconhecimento de padrões, raciocínio baseado em casos, mineração de dados e em muitos outros domínios. O *k*-NN pertence à classe dos algoritmos baseados em instâncias (Mitchell, 1997) e, como um exemplar de algoritmo desta classe, não demanda nenhum procedimento para treinamento afim de desenvolver um modelo genérico sobre um conjunto de instâncias (*CI*). O próprio *CI* pode ser considerado como o conjunto de treinamento do algoritmo *k*-NN de tal modo que a inclusão de uma nova instância no *CI* não requer nada além da própria inclusão da instância.

Dada uma instância x e um inteiro $k \geq 1$, *o* k-NN busca, em um *CI* com $n \geq 1$ instâncias descritas em um espaço de $m \geq 1$ dimensões[1], k instâncias mais similares à instância x, tendo por referência uma função de similaridade entre as instâncias. Frequentemente a função de similaridade é definida como uma métrica sobre o espaço m-dimensional e, entre possíveis métricas, a distância euclidiana (Anton, 1994) é comumente utilizada como função de similaridade.

O mecanismo de busca do k-NN é simples e consiste apenas em computar o valor da função de similaridade entre a instância x e cada uma das instâncias do *CI* para, posteriormente, selecionar as k instâncias mais similares a x e, com base em tal seleção, computar uma classe tendo como referência as k classes das instâncias selecionadas. Em contrapartida a esta simplicidade está o alto consumo de tempo da busca, especialmente em aplicações *Big Data* nas quais o conjunto de instâncias é massivamente populado e frequentemente possui alta dimensionalidade (número de atributos). Diversos métodos têm sido propostos para diminuir o tempo de busca do k-NN, entre eles estão os métodos aproximados (e.g, Wang, Pan & Li, 2019; Andoni & Indyk, 2008) que abrem mão da busca exata das k instâncias mais similares a x em troca da diminuição do tempo de busca, e outros métodos que objetivam a busca exata e exigem o preenchimento de estruturas de dados especiais em uma fase anterior à busca (e.g, Fukunaga & Narendra, 1975; Liu, Moore & Gray, 2003).

Este trabalho objetiva a busca exata em um *CI* das k instâncias mais similares a uma dada instância x. Quando a função de similaridade é uma métrica sobre o espaço m-dimensional do *CI*, o conceito de desigualdade triangular pode ser empregado para desenvolver variantes do algoritmo k-NN, via estratégias que visam economizar cálculos da função de similaridade. O foco deste trabalho está na diminuição do tempo de busca do k-NN via a utilização do conceito de desigualdade triangular (Lima, 2013). Especificamente, uma estratégia de uso do conceito de desigualdade triangular é proposta para desenvolver uma variante do algoritmo k-NN denominada k-NN-I. Dois algoritmos recentemente propostos, que também realizam busca exata e implementam o uso de desigualdade triangular, KMC-OTI-FS (Pan *et al*., 2020) e kMkNN (Xueyi, 2011) são estudados com o propósito de reuni-los futuramente em um experimento de medida da economia da quantidade de cálculos de função de similaridade e de tempo de busca.

O restante deste artigo está organizado da seguinte forma. A Seção 2 relata o principal fundamento que motiva e orienta este trabalho, a desigualdade triangular, juntamente com outros fundamentos relacionados a este conceito. A Seção 3 descreve as ideias que fundamentam os algoritmos KMC-OTI-FS e kMkNN. A Seção 4 apresenta o algoritmo k-NN-I, proposto neste trabalho. Os experimentos planejados para serem realizados futuramente são descritos na Seção 5. Por fim, a Seção 6 apresenta algumas considerações finais sobre o estágio atual deste trabalho em desenvolvimento.

2. Fundamentos: espaços métricos, métrica, desigualdade triangular e dis(similaridade)

Em Matemática, o conceito de espaço métrico reúne o conceito de conjunto e o conceito de métrica sobre os elementos do conjunto. Uma métrica é uma função que estabelece um valor real entre dois quaisquer elementos de um conjunto. O conceito de espaço métrico

[1] O k-NN é classificado como um algoritmo supervisionado e, como tal, opera sobre conjunto de instâncias que apresentam, na dimensão $m + 1$, a descrição da classe a que a instância pertence.

e de métrica são úteis em Aprendizado de Máquina para estabelecer uma noção de similaridade ou de dissimilaridade entre instâncias pertencentes a um conjunto de instâncias. Os conceitos descritos a seguir são uma adaptação, para a terminologia comumente utilizada em Aprendizado de Máquina, do conceito de espaço métrico descrito em Lima (2013).

Um **espaço métrico** é um par ordenado $< CI, d >$, onde *CI* é um conjunto de instâncias e *d* é uma **métrica** sobre os elementos de *CI*, i.e., *d* é uma função $d: CI \times CI \rightarrow \mathbb{R}$, tal que, para quaisquer três instâncias $x_1, x_2, x_3 \in CI$, os seguintes axiomas são válidos:

1. Reflexividade ou identidade: $d(x_1, x_2) = 0 \leftrightarrow x_1 = x_2$.
2. Simetria: $d(x_1, x_2) = d(x_2, x_1)$.
3. Desigualdade triangular: $d(x_1, x_2) \leq d(x_1, x_3) + d(x_2, x_3)$.

A métrica *d* é comumente chamada de "função distância" ou, simplesmente de "distância". Um exemplo de métrica, muito utilizada por algoritmos de Aprendizado de Máquina, é a distância euclidiana (Lima, 2013; Theodoridis, & Koutroumbas, 2009) *d* entre duas instâncias pertencentes a um conjunto de instâncias *CI* do espaço métrico descrito por $< CI, d >$.

O Axioma 1 estabelece que é zero a distância de uma instância a ela mesma e, inversamente, duas instâncias iguais distam zero uma da outra. O Axioma 2 estabelece que a distância de uma instância x_1 até uma instância x_2 é igual a distância de uma instância x_2 até uma instância x_1. De especial importância para este trabalho é o Axioma 3. Conhecido como **desigualdade triangular**, este axioma estabelece que a distância entre duas instâncias x_1 e x_2 é sempre menor ou igual à soma da distância delas para uma terceira instância x_3. Informalmente, pode-se dizer que o axioma da desigualdade triangular estabelece que x_3 não é um atalho para o caminho que conecta x_1 e x_2 ou, ainda que o comprimento de um lado de um triângulo é menor ou igual a soma do comprimento dos outros dois lados.

A função de **dissimilaridade** entre duas instâncias é frequentemente associada a uma função distância *d*, sendo frequentemente definida como

$$dsim(x_1, x_2) = d(x_1, x_2). \qquad \text{(Eq. 1)}$$

Ou seja, a distância entre duas instâncias x_1 e x_2 é uma medida do quanto elas são dissimilares entre si. A noção de **similaridade** é definida como o oposto da dissimilaridade, ou seja, quanto mais dissimilar são duas instâncias x_1 e x_2 menos similar elas são.

3. Trabalhos Relacionados

Cálculos da função de dissimilaridade (ou similaridade) demandam muito tempo, especialmente em *CI*s massivamente populados ou com instâncias de alta dimensão. O conceito de desigualdade triangular tem sido utilizado para evitar cálculos da função de dissimilaridade entre instâncias em variantes do algoritmo *k*-NN e, com isto, acelerar o mecanismo de busca. Dois recentes algoritmos, kMkNN e KMC-OTI-FS utilizam diferentes estratégias de aplicação da desigualdade triangular visando acelerar o mecanismo de busca do *k*-NN. As ideias que fundamentam estes algoritmos são descritas nesta seção.

O algoritmo kMkNN (Xueyi, 2011) – do Inglês, *k-means for k-nearest neighbors* – utiliza o algoritmo de agrupamento *k*-Means (MacQueen, 1967) para criar $q = \sqrt{n}$ grupos de instâncias do *CI* em uma fase anterior à fase de busca, onde *n* é o número de instâncias do *CI*. Esta fase, denominada em Xueyi por fase *offline*, é uma fase de preparo para a fase de busca propriamente, esta última denominada de fase *online*. A fase de busca do algoritmo utiliza dados obtidos na fase *offline*: os grupos, os centroides dos grupos, os raios dos grupos e, para cada grupo, a distância de cada instância dentro do grupo ao centroide do grupo. Tendo como entrada uma instância *x* e um valor *k* para o número de instâncias mais similares a *x*, a serem buscadas, o algoritmo calcula as distâncias euclidianas – a métrica usada em Xueyi (2011) é a distância euclidiana – da instância *x* para cada um dos centroides dos grupos. O objetivo desta busca é localizar o grupo que tem a possibilidade de conter as *k* instâncias mais similares a *x*, tendo como métrica a distância euclidiana. Posteriormente, o algoritmo itera sobre os $q - 1$ outros grupos utilizando uma estratégia de desigualdade triangular para economizar cálculos de distância. Ao final, o algoritmo entrega como saída o conjunto das *k* instâncias mais similares a *x*. Em um experimento com 20 *CI*s de diferentes tamanhos e dimensões Xueyi afirma que o algoritmo kMkNN tem bom desempenho de tempo, especialmente em *CI*s com alta dimensão, quando comparado com duas variantes tradicionais do algoritmo *k*-NN: *kd*-Tree (Fukunaga & Narendra, 1975) – usa árvores KD como estrutura de dados para orientar a busca – e Ball-Tree (Liu, Moore & Gray, 2003) – usa árvores Ball para orientar a busca.

O algoritmo KMC-OTI-FS (Pan *et al*., 2020) estende o algoritmo kMkNN usando outra estratégia de emprego da desigualdade triangular que, segundo seus autores, otimiza o kMkNN, reduzindo o número de cálculos de distância euclidiana no processo de busca, às custas de um aumento de uso de memória em *q* vezes, onde *q* é quantidade de grupos formado na fase *offline* do algoritmo. Em um experimento com *CI*s, de diferentes dimensões, os autores afirmam que o algoritmo KMC-OTI-FS supera em desempenho de tempo outras propostas para acelerar *k*-NN: Ball-Tree (Liu, Moore & Gray, 2003), kdTree (Fukunaga & Narendra, 1975), VP-Tree (Yianilos, 1993), kMkNN (Xueyi, 2011) e kNNVWC (Almalawi *et al*., 2016).

4. *k*-NN-I: o algoritmo proposto

O conceito de desigualdade triangular pode ser aplicado também para a função dissimilaridade, dado que a dissimilaridade é definida como sendo igual à função distância (Equação 1). Assim, existe uma desigualdade triangular entre os valores das dissimilaridades de três instâncias *x*, *a* e *j* que pode ser escrita como:

$$dsim(a,j) \leq dsim(x,a) + dsim(x,j). \qquad \text{(Eq. 2)}$$

Tendo em vista que a função distância é sempre positiva ou nula, assim também é a função de dissimilaridade. Desse fato e da Equação 2 decorre o seguinte limite inferior para a dissimilaridade entre uma instância *x* e uma instância *j*:

$$dsim(x,j) \geq |\, dsim(a,j) - dsim(x,a) \,|. \qquad \text{(Eq. 3)}$$

O limite inferior dado pela Equação 3 é usado pelo algoritmo *k*-NN-I, para economizar cálculos da função de dissimilaridade, conforme está descrito nesta seção.

O pseudocódigo do algoritmo *k*-NN-I é apresentado na Figura 1. O algoritmo possui uma fase *offline* onde cálculos, que auxiliarão na aceleração da fase de busca, são

realizados e armazenados, e uma fase *online* que representa a fase de busca propriamente. A fase *offline* produz a matriz *DSIM* de dissimilaridade, de dimensão *n* x *n*, preenchida com as dissimilaridades entre todas as *n* instâncias do *CI*, uma a uma. Assim, *DSIM* [x_1, x_2] contém o valor da dissimilaridade entre as instâncias x_1 e x_2. Um vetor *S*, de dimensão igual a *n*, também é produzido na fase *offline*. O vetor *S* referencia, para cada elemento, a instância mais similar ao elemento. Assim, por exemplo, *S* [x_1] referencia a instância mais similar à instância x_1.

Algoritmo *k*-NN-I

Fase *offline*

Entrada: CI: Conjunto indexado de instancias; n: número de instâncias em CI.

Saída: DSIM, matriz de dimensão n x n, em que DSIM[v, w] indica o valor da dissimilaridade entre as instâncias de índice v e w; S: vetor de dimensão n, em que S[v] possui uma referência à instância mais similar à instância de índice v.

Fase online (busca propriamente)

Entrada: CI: conjunto indexado de instancias; n: número de instâncias em CI; x: instância que se quer buscar instâncias mais similares em CI; k: quantidade de instâncias similares que se quer buscar.

Usa: DSIM e S calculados na fase offline.

Saída: SK: fila, de dimensão igual a k, ordenada da instância mais similar a x (frente da fila) até a k-ésima instância mais similar a x (fim da fila).

```
1   {
2     para i := 1 até k faça {
3         // Escolhe aleatoriamente em CI uma instância para que seja considerada,
4         // inicialmente, a instância mais similar à instância x. A partir de i = 2,
5         // escolhe como a instância a ser considerada, inicialmente, a i-ésima
6         // instância mais similar a x, a instância mais similar à instância que foi o
7         // resultado da busca da instância mais similar a x, calculada na iteração
8         // anterior. Seja a o índice desta instância em qualquer dos dois casos.
9         se ( i = 1 ) a  := random (n);
10        senão {s := instancia referenciada no fim da fila SK; a := S[s];}
11        // Calcula a dissimilaridade entre x e a instância de índice a.
12        dxa := dsim(x, CI[a]);
13
14        para cada uma das outras instâncias de índice j em CI faça {
15          // Usa a desigualdade triangular dada pela Equação 3, para verificar se é
16          // possível economizar cálculo da função de dissimilaridade. Isto é feito
17          // calculando-se um limite inferior para a dissimilaridade entre a instância x e
18          // a instância de índice j do CI.
19          Linf := | DSIM[ CI[a], CI[j] ] – dxa |;
20          se ( Linf ≥ dxa )
21            // Não é necessário calcular a dissimilaridade entre x e a instância de
22            // índice j, ou seja, economiza cálculo da dissimilaridade.
23          senão {
24          // Calcula a dissimilaridade entre x e a instância de índice j.
25            dxj := dsim(x, CI[j]);
26            se (dxj < dxa) {
27              // A instância de índice j é considerada, agora, a instância corrente mais
28              // similar a x.
29              a := j;  dxa := dxj;
30            }
31
32      } // Fim do "para cada uma das outras ... "
33
```

```
34      Insere o valor da variável a na fila SK para sinalizar que este é
35      o índice da i-ésima instância mais similar a x;
36     } // Fim do "para i := ... "
37   }
```

Figura 1. Pseudocódigo do algoritmo *k*-NN-I.

Na fase *on-line*, fase de busca propriamente, o algoritmo recebe como entrada o conjunto *CI* de instâncias, o número *n* de instâncias do *CI*, a instância *x* que se quer buscar instâncias mais similares no *CI* e o número *k* de instâncias similares que se quer buscar. Essa fase usa a matriz *DSIM* e o vetor *S*, calculados na fase *offline* e retorna uma fila *SK* ordenada da instância mais similar a *x* (frente da fila) até a *k*-ésima instância mais similar a *x* (fim da fila).

Para a busca da primeira instância similar a *x*, o algoritmo sorteia uma instância do *CI* para ser considerada, inicialmente, a mais similar (linha 9 da Figura 1). O algoritmo itera entre as demais instâncias do *CI* e usa o limite inferior, dado pela Equação 3, para economizar cálculos da função de dissimilaridade (linhas 15 a 22 da Figura 1). Quando não é possível tal economia, os cálculos são realizados (linhas 24 a 29 da Figura 1). A economia de cálculos da função de dissimilaridade depende do sorteio realizado e da ordem com que as demais instâncias são tomadas. A economia é máxima caso a instância sorteada como a mais similar a *x* seja a mais similar a *x*. Neste caso, não há necessidade de cálculos adicionais; o teste de desigualdade triangular (linhas 15 a 22 da Figura 1) descarta a necessidade de cálculos da função de dissimilaridade.

A busca das *k* instâncias mais similares a *x* para $i \geq 2$ tem o potencial de economizar muito mais cálculo da função de dissimilaridade do que para a primeira, sendo que esta é a principal ideia deste algoritmo. É que, em vez de sortear uma instância para ser considerada inicialmente mais similar a *x*, o algoritmo usa a instância, digamos *s*, buscada na iteração anterior e elege para ser a instância mais similar a *x*, inicialmente, a instância mais similar a *s* (linha 10 da Figura 1).

5. Experimentos Planejados

Planeja-se comparar o desempenho dos algoritmos *k*-NN-I, *k*-NN, kMkNN e KMC-OTI-FS usando a coleção de 9 *CI*s descritos a seguir por meio da sintaxe "nome (número de instâncias, número de dimensões): Arcene(900, 10000), dorothea(1950, 770), DUWWTP(527, 38), GasSensors(13910, 128), Multiplefeatures(2000, 649), Shuttle(58000, 9), Slice(53500, 384), Spambase(4601, 57) e Waveform(5000, 40). Todos estes *CI*s estão disponíveis no repositório UCI (Dua & Graff, 2019) e serão testados para *k* igual a 10, 50, 100 e 200. Estes *CI*s têm diferentes tamanhos e possuem dimensões pequena, média e alta. A comparação será realizada tendo como referência os índices:

- RRDC (Pan *et al*., 2020), taxa de redução de cálculos de distância, dada em termos percentuais por $RRDC = 100\ (1 - c)/(b\ n)$, onde *c* é o número de cálculos de distância realizados pelo algoritmo, *b* é a quantidade de buscas realizadas e *n* é o número de instâncias de treinamento do *CI*.
- RRST (Pan *et al*., 2020), taxa de redução do tempo de busca, dada em termos percentuais por $RRST = 100\ (1 - t)/\ t_{k-NN}$, onde *t* é o tempo de busca gasto

pelo algoritmo no experimento para b buscas e $t_{k\text{-NN}}$ é o tempo gasto pelo algoritmo *k*-NN para realizar *b* buscas.

Todos os experimentos usarão uma validação cruzada de 10 dobras pelo método *s-fold* (Bishop, 2011) de modo que a quantidade de buscas a ser realizada em cada experimento, para cada valor de *k*, é igual a $b = (n - n/10)\, n/10$.

6. Considerações Finais

A letra "I" no final do nome do algoritmo proposto, *k*-NN-I, vem de "inflamar" e foi proposta com o objetivo de chamar a atenção para a dinâmica da economia de cálculos de função de dissimilaridade que este algoritmo proporciona. O processo de acendimento de uma fogueira de lenha é lento no início, especialmente quando não há disponível pedaços pequenos de lenha. Por outro, o processo de acendimento é mais fácil, inicialmente, com pequenos gravetos. Ao inflamarem-se, os pequenos gravetos vão gradativamente levando o fogo aos pedaços de lenha maiores até que, em um certo momento, o fogo toma de forma intensa todas as lenhas da fogueira. A inflamação é uma metáfora para a dinâmica do processo de economia de cálculos da função de similaridade proporcionada pelo algoritmo proposto. A economia é lenta quando o processo de busca de *k* instâncias similares a uma instância *x* está lidando com instâncias muito dissimilares a *x* (distâncias grandes). No entanto, o processo de economia se intensifica quando atinge ou se inicia lidando com instâncias mais similares a *x* (distâncias pequenas). A partir da busca da segunda instância mais similar a *x* o processo de economia de cálculos de função de similaridade proporcionada pelo *k*-NN-I torna-se ainda mais vigoroso, pois se inicia por instâncias que tendem a ser mais similares a *x*.

Algoritmos como o *k*-NN-I melhoram o desempenho de tempo da busca graças a uma fase de preparo, neste artigo denominada de fase *offline*. A fase de preparo normalmente envolve cálculos custosos em termos de tempo e aumentam a complexidade de espaço. A inserção, remoção ou alteração de uma instância no *CI* demanda um recálculo parcial ou total. Assim, uma importante questão é: existem aplicações reais para estes algoritmos? Felizmente, a resposta é sim. Não é raro encontrar aplicações em que o número de buscas é muito maior do que eventuais inserções, remoções ou alterações no *CI*. Por exemplo, em sistemas para Raciocínio Baseado em Casos frequentemente o número de recuperação de casos na base de casos tem uma ordem de grandeza absolutamente maior do que a retenção de um novo caso na base de casos.

Este artigo descreveu um trabalho em desenvolvimento que planeja investigar o desempenho do algoritmo *k*-NN-I em relação a recentes algoritmos correlatos. Resultados preliminares são promissores e motivadores, mas independentemente do que vier a ser concluído, algo já é certo: o *k*-NN-I é muito mais simples do que seus principais correlatos kMkNN e KMC-OTI-FS.

Referencias

Almalawi, A., Fahad, A., Tari, Z., Cheema, M. & Khalil, I. (2016). kNNVWC: An Efficient k-Nearest Neighbors Approach Based on Various-Widths Clustering. *IEEE Trans. Knowl. Data Eng.*, 28 (1), 68–81.

Andoni, A. & Indyk, P. (2008). Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51, 117–122.

Anton, H. (1994). *Elementary Linear Algebra* (7th ed.). Edited by Nadia Magnenat-Thalmann and Daniel Thalmann, John Wiley & Sons ltd.

Bishop, C. M. (2011). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag.

Cover, T. M. & Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*. 13 (1), 21–27.

Dua, D. & Graff, C. (2019). *UCI Machine Learning Repository* [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science

Fukunaga K. & Narendra, P. M. (1975). A branch and bound algorithm for computing. *IEEE Trans. Comput.*, 100 (7), 750–753.

Lima, E. L. (2013). *Espaços métricos*. 5ª ed. IMPA.

Liu T., Moore, A. W. & Gray, A, (2003). Efficient exact k-NN and nonparametric c classification in high dimensions, In *Proceedings of Neural Information Processing Systems*. Vancouver and Whistler, British Columbia, Canada.

MacQueen, J. B. (1967). Some Methods for classification and Analysis of Multivariate Observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*. Berkeley, USA. 281–297.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.

Pan, Y., Pan, Z., Wang, Y. & Wang, W. (2020). A new fast search algorithm for exact k-nearest neighbors based on optimal triangle-inequality-based check strategy. *Knowledge-Based Systems*. 189. Article 105088.

Wang, Y., Pan, Z. & Li, R. (2019). A New Cell-Level Search Based Non-Exhaustive Approximate Nearest Neighbor (ANN) Search Algorithm in the Framework of Product Quantization. *IEEE Access*, 7, 37059–37070.

Xueyi, W. (2011). A fast exact k-nearest neighbors algorithm for high dimensional search using k-means clustering and triangle inequality. In *Proceedings Int. Joint Conf. Neural Netw*. 1293–1299.

Yianilos, P. N. (1993). Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the fourth annual ACM-SIAM symposium on Discrete algorithm*. 311–321.

Theodoridis, S. & Koutroumbas, K. (2009). *Pattern recognition*. Academic Press.