

# Product Quantization for Nearest Neighbor Search

## A parallel approach

Marcelo de Araújo<sup>1</sup>, André Fernandes<sup>1</sup>

<sup>1</sup>Departamento de Ciência da Computação - Universidade de Brasília(UNB)

**Resumo.** O artigo baseia-se na ideia proposta por [Herve Jegou ], onde o espaço é decomposto em vários subespaços de um produto cartesiano, produzindo vetores menores, que serão aproximados separadamente, e usados para a criação de uma lista invertida junto com uma base de dados contendo os códigos referentes a cada vetor da base, onde toda busca será feita por meio da lista invertida. Também será apresentada uma proposta de paralelização no ambiente distribuído, com o foco na parte de busca.

### Introdução

Dados um vetor  $x$ , e um conjunto de vetores  $Y \subset R^n$ , queremos achar o vetor  $y$  do conjunto  $Y$  que mais se aproxima de  $x$ , chamando de  $NN(x)$  o vizinho mais próximo e definido como:

$$NN(x) = \arg \min d(x, y) , y \in Y \quad (1)$$

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (2)$$

Onde  $d(x, y)$  é a distância euclidiana entre  $x$  e  $y$ . Porém para conjuntos  $Y$  grandes seria muito custoso a busca exaustiva. Por isso a estratégia adotada em [1], tenta aproximar os vetores da base  $Y$  em outro conjunto de vetores, chamados centróides( $c_i \in C$ ) aproximados com o algoritmo K-means a partir de um conjunto de treino.

Com o centróides conhecidos podemos definir formalmente como  $q(.)$  a função que mapeia um vetor arbitrário  $x \in R^n$  em  $q(x) \in C = \{c_i ; i \in I\}$ , onde  $I$  é um intervalo finito,  $I = \{0, \dots, k-1\}$  e  $c_i$  são centróides.

$$q(x) = \arg \min d(x, c_i) , c_i \in C \quad (3)$$

Além de aproximar os vetores  $y$  da base em seus centróides mais próximos, centróides são criados a partir de subvetores, e assim vetores  $y$  são divididos em partes de dimensão  $d = \frac{n}{m}$  e assinalada a cada subdimensão do centróide.

$$\begin{aligned} y &= \{y_1, y_2, \dots, y_n\}, \text{ seus respectivos subvetores } u_i \\ u_1 &= \{y_1, y_2, \dots, y_d\}, u_2 = \{y_{d+1}, y_{d+2}, \dots, y_{2d}\} \\ u_m &= \{y_{n-d}, y_{n-d+1}, \dots, y_n\}, u_i \in R^d \end{aligned} \quad (4)$$

E seus respectivos centróides de seus subespaços:

$$q(y) = \{q(u_1), q(u_2), \dots, q(u_m)\}, q(u_i) \in C \quad (5)$$

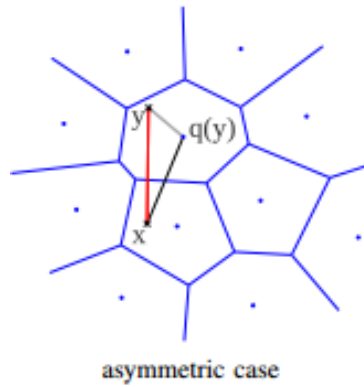


Figure 1. Centroides e Vetores

### Lista Invertida

Com a finalidade de tornar a busca mais eficiente uma estrutura de lista invertida foi utilizada por [Herve Jegou ].

Para montar a lista são usados dois conjuntos de centróides  $C_1$  e  $C_2$ , onde  $C_1$  representa os centroides assinalados a base de treino  $T$ , chamados em [Herve Jegou ] por *coarse centroids*, e após conhecidos,  $C_2$  é calculado e são os centróides assinalados ao resto ,  $r(t)$ , dos vetores de treino com cada um de seus centróides.

$$\begin{aligned} q(t) &\in C_1 \\ r(t) &= y - q(t), y \in T \\ q(r(t)) &\in C_2 \end{aligned} \quad (6)$$

Com os conjuntos  $C_1$  e  $C_2$  conhecidos, podemos montar a estrutura da lista em si, indexando os vetores de uma base  $Y$  na lista, da seguinte forma:

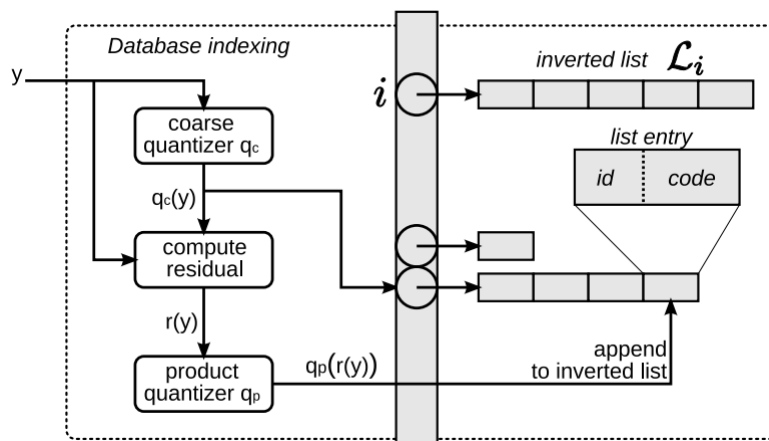


Figure 2. Processo de indexação

Cada entrada da lista representa um centróide de  $C_1$  e cada entrada da lista contida representa o centróide de  $C_2$  possuindo os identificadores dos vetores  $y$  da base que possuem aquele centróide como o mais próximo.

## Algoritmo

### Aprendizagem

Primeiramente o algoritmo necessita aprender os centróides  $c_i$  dos dois conjuntos  $C_1$  e  $C_2$ , para sabermos a função  $q(\cdot)$ , e realiza isto na parte de aprendizagem, onde a partir de uma base de treino  $T$  os conjuntos são aprendidos com o algoritmo K-means.

### Indexação

A figura 2 representa o processo de indexação de uma base de dados  $Y$ , onde e feito da seguinte forma:

- Para cada vetor  $y_i \in Y$  calculamos seu centroide mais próximo  $c_i \in C_1$ , assim sabemos a entrada da lista principal.
- Calculamos  $r(y_i)$  conforme (6) e calculamos o centroide mais próximo  $q(r(y_i)) = c_j \in C_2$ , para cada subdimensão
- Agora que temos o código para cada  $c_j \in C_2$ , guardamos na entrada correspondente junto com o identificador do vetor.

### Busca

Durante a busca, como dito na secção 1, queremos buscar o vizinho mais próximo de um determinado vetor  $x$ , ou  $k$  vizinhos mais próximos dele.

- Procuramos o centroide  $c_i \in C_1$  mais próximo de  $x$ , agora sabemos qual entrada da lista possui vetores associados ao mesmo centroide.
- Calculamos o  $r(x)$  e usamos para calcular a  $d(r(x), c_j)$ ,  $c_j \in C_2$ , para cada subdimensão.
- Somamos as distâncias das subdimensões de interesse, aquelas cujos  $c_j$  se encontram na entrada da listad descoberta no primeiro passo.
- Com as distâncias podemos procurar as  $k$  distância mínimas, gerando uma lista  $L$  a de possíveis candidatos da base  $Y$  próximos a  $x$ , que são encontrados pelos seus identificadores presentes nas entradas de cada lista.

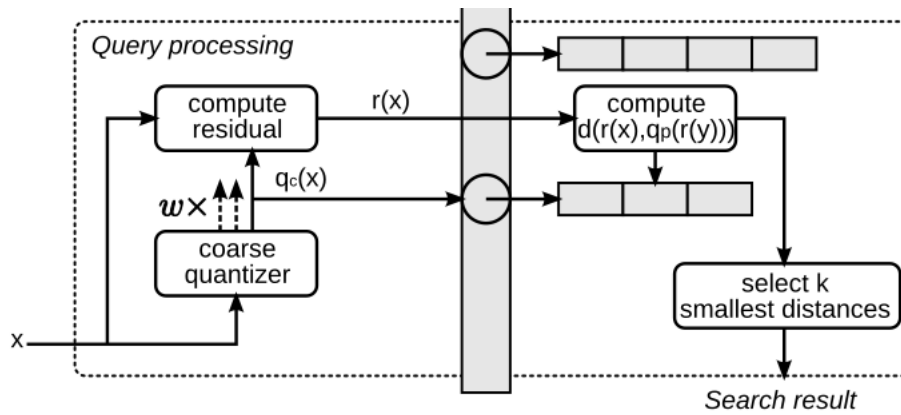


Figure 3. Processo de Busca

## **Solução Paralela**

## **Resultados**

## **Conclusão**

## **References**

Herve Jegou, Matthijs Douze, C. S. Product quantization for nearest neighbor search.  
33(1):117–128.

[Herve Jegou ]