Product Quantization for Nearest Neighbor Search
A parallel aproach

Marcelo de Araújo [1], **André Fernandes** [1]

[1]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.*

1. 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:

$$NN(x) = \arg\min d(x, y)\ ,\ y \in Y \tag{1}$$

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

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, $I = \{0, \cdots, k-1\}$ e c_i são centróídes.

$$q(x) = \arg\min d(x, c_i)\ ,\ c_i \in C \tag{3}$$

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.

$$\begin{gathered} y = \{y_1, y_1, \cdots, y_n\},\ \text{seus respectivos subvetores } u_i \\ u_1 = \{y_1, y_2, \cdots, y_d\},\ u_2 = \{y_{d+1}, y_{d+2}, \cdots, y_{2d}\} \\ u_m = \{y_{n-d}, y_{n-d+1}, \cdots, y_n\},\ u_i \in R^d \end{gathered} \tag{4}$$

E seus respectivos centróídes de seus subespaços:

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

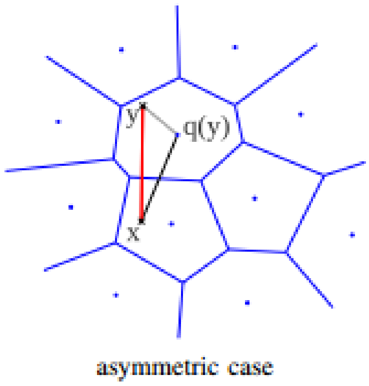

Figure 1. Centroides e Vetores

1.1. 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 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.

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

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:

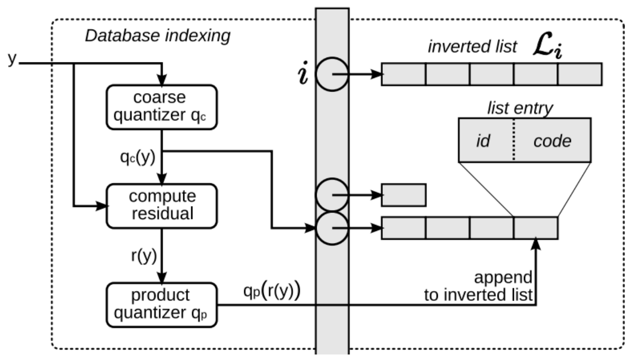

Figure 2. Processo de indexação

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

2. Algoritmo

2.1. Aprendizagem

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

2.2. 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 qual $c_j \in C_2$ guardamos na entrada correspondente junto com o identificador do vetor na base.

3. Solução Paralela

4. Sections and Paragraphs

4.1. Subsections

5. Figures and Captions

References

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

[Herve Jegou]