Uniform Variance Product Quantization

Qin-Zhen Guo, Zhi Zeng and Shuwu Zhang

95 Zhongguancun East Road, 100190, BEIJING, CHINA

{ qinzhen.guo, zhi.zeng, shuwu.zhang }@ia.ac.cn

Keywords: Approximate nearest neighbor search, product quantization, uniform variance projection, image retrieval.

Abstract.
Product quantization (PQ) is an efficient and effective vector quantization approach to fast approximate nearest neighbor (ANN) search especially for high-dimensional data. The basic idea of PQ is to decompose the original data space into the Cartesian product of some low-dimensional subspaces and then every subspace is quantized separately with the same number of codewords. However, the performance of PQ depends largely on the distribution of the original data. If the distributions of every subspace have larger difference, PQ will achieve bad results as shown in our experiments. In this paper, we propose a uniform variance product quantization (UVPQ) scheme to project the data by a uniform variance projection before decompose it, which can minimize the subspace distribution difference of the whole space. UVPQ can guarantee good results however the data rotate. Extensive experiments have verified the superiority of UVPQ over PQ for ANN search.

Introduction

Approximate nearest neighbor (ANN) search has been widely used to avoid excessive computational and memory cost of nearest neighbor (NN) search in many computer vision problems, such as image retrieval, scene classification and 3D reconstruction. The key idea of ANN is to find the nearest neighbors with high probability instead of probability one. It is a fundamental problem in computer vision community. Nowadays, several ANN techniques have been developed including tree-based methods, hashing-based methods and vector quantization.

The tree-based methods use spatial partitions and recursive hyperplane decomposition to provide an efficient approach for low-dimensional data search, such as KD-tree [1]. However, the tree-based methods can degenerate to exhaustive search in the worst case for high-dimensional data.

The hashing-based methods [2, 3] have been a popular approach to ANN search. These methods map the data into Hamming space and the similarity between two data points is approximated by the Hamming distance of their hashed codes. But because of the thick boundary of Hamming space, these methods usually cannot achieve ideal results [4].

Vector Quantization (VQ) [5] is an effective method for ANN search. These methods quantize the data into codewords in order to reduce the cardinality of the data space. The Euclidean distances of vectors can be approximated by the distances of codewords. Among the VQ methods, product quantization (PQ) [6] was designed to decompose the original data space into the Cartesian product of m low-dimensional subspaces and each subspace is quantized into k codewords by k-means. The distance between two data points is approximated by the sum of the distances between their codewords of the same subspace. PQ obtains more accurate results than various hashing methods, which is largely due to its lower quantization distortions and more precise distance computation.

However, PQ considers every subspace having the same importance and uses the same number of codewords to quantize every subspace, which is unreasonable since the distribution of every subspace of the data may be not identical, especially when the data are projected by principal component analysis (PCA). The subspace formed by dimensions with larger variance carry more information. If we use the same number of codewords in subspaces with more information to quantize the data in subspaces with less information, it will be redundant. And the distortion will be expanded if we use the same number of codewords in subspaces with less information to quantize the data in subspaces with more information. Hence, the performance of PQ depends largely on the distribution of the original data. Motivated by the above considerations, in this paper, we propose a uniform variance product quantization (UVPQ) approach for ANN search. The key idea is that before decomposing the data into subspaces, we use a uniform variance projection to map the data into a new space where every subspace have the same information. UVPQ can ensure good results even though the distributions of every subspace are very different. Extensive experiments demonstrate the superiority of our method.

Uniform Variance Product Quantization

In this section, we introduce the details of our uniform variance product quantization (UVPQ) scheme. Firstly, we will review the prior work PQ briefly. Then the UVP problem is stated.

First of all, let us introduce some notations. We have a training set of n data $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$, $\boldsymbol{x}_i \in \mathbb{R}^d$, that form the rows of the data matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$. $\boldsymbol{x}^{(i)}$ is the i-th subvector of $\boldsymbol{x}$. We use $\boldsymbol{X}^i$ to indicate the subspace that $\boldsymbol{x}^{(i)}$ forms and X_j^i means the j-th dimension of subspace $\boldsymbol{X}^i$.

Product Quantization. The basic idea of PQ is that the original data are decomposed into m subspaces of dimension $q = d/m$ (see Eq. 1).

$$\overbrace{x_1, \cdots, x_q}^{x^{(1)}}, \cdots, \overbrace{x_{d-q+1}, \cdots x_d}^{x^{(m)}} \tag{1}$$

Then find k codewords to represent the data effectively in every subspace. The most intuitive method is to cluster the data by k-means algorithm and use the cluster centers as codewords to represent the data based on nearest neighbor (NN) rule. The original data can be represented by the Cartesian product of all the subspace. And the Euclidean distance between two points can be computed by:

$$D(x, y) = \sum_{i=1}^{m} D(x^{(i)}, y^{(i)}) \tag{2}$$

D($\boldsymbol{x}$, $\boldsymbol{y}$) means the Euclidean distance between $\boldsymbol{x}$ and $\boldsymbol{y}$.

In every subspace, D($\boldsymbol{x}^{(i)}$, $\boldsymbol{y}^{(i)}$) can be approximated by D(c($\boldsymbol{x}^{(i)}$), c($\boldsymbol{y}^{(i)}$)) for symmetric distance computation or D($\boldsymbol{x}^{(i)}$, c($\boldsymbol{y}^{(i)}$)) for asymmetric distance computation if $\boldsymbol{x}$ is the query and $\boldsymbol{y}$ is an item in the database where c($\boldsymbol{x}^{(i)}$) and c($\boldsymbol{y}^{(i)}$) are codewords that $\boldsymbol{x}^{(i)}$ and $\boldsymbol{y}^{(i)}$ are quantized to in the i-th subspace respectively. D($\boldsymbol{x}$, $\boldsymbol{y}$) can be computed fast by looking up table if we precompute D(c($\boldsymbol{x}^{(i)}$), c($\boldsymbol{y}^{(i)}$)) or D($\boldsymbol{x}^{(i)}$, c($\boldsymbol{y}^{(i)}$)). Moreover, the items in the database can be indexed by the codewords' indices with m*log$k$ bits, which can largely reduce the storage.

Uniform Variance Projection.

Definition 1 Average Distribution Variance (ADV)

$$ADV(X^i) = \frac{1}{q} \sum_{j=1}^{q} \mathrm{var}(X_j^i) \tag{3}$$

where $\mathrm{var}(X_j^i)$ denotes the variance of the j-th dimension of the subspace $\boldsymbol{X}^i$.

Definition 2 Subspace Distribution Difference (SDD)

$$SDD(X) = \frac{1}{m} \sum_{i=1}^{m} (ADV(X^i) - \frac{1}{m} \sum_{j=1}^{m} ADV(X^j))^2 \tag{4}$$

where ADV($\boldsymbol{X}^i$) denotes the ADV of the i-th subspace of the $\boldsymbol{X}$, which is the whole space.

We use ADV, which can reflect the information of a subspace, to describe the variance of a subspace and SDD to describe the variance of the whole space. For the original data, the ADV of every subspace is not the same and the SDD of the whole space may be large. It is unreasonable to use the same number of codewords to quantize all the subspaces.

In order to minimize the SDD of the whole space, we use an orthogonal uniform variance projection ***P*** to project (or rotate) the data, which does not change the neighbor structure of the original data. We formulate our objective function as:

$$\arg\min_{P^T P=I} SDD(XP), \tag{5}$$

It is easy to understand that the optimal solution of Eq. 5 is ADV($(\boldsymbol{XP})^i$) = ADV($(\boldsymbol{XP})^j$) for $i \neq j$. And in order to achieve this goal, we compel ***P*** to satisfy $\mathrm{var}((XP)_u^i) = \mathrm{var}((XP)_v^j)$ for $1 \le i,j \le m$, $1 \le u,v \le q$. This means that for $\boldsymbol{Y}=\boldsymbol{XP}$, every dimension of ***Y*** has the same variance. Directly solve the problem is very difficult because there are much degrees of freedom. Since ***P*** is an orthogonal matrix, by some simple mathematical transformations, we have Eq. 6

$$\Lambda = Y^T Y = P^T X^T XP, \tag{6}$$

Λ is a diagonal matrix and $\Lambda(i, i) = \Lambda(j, j)$ for $1 \le i,j \le d$. Eq. 6 has the similar formula as [7]. And we can respectively use lift and project algorithm and gradient flow algorithm to obtain the uniform variance projection matrix ***P***. The corresponding approaches are termed UVPQ-LP and UVPQ-GF.

Experiments

In this section we evaluate our methods for approximate nearest neighbor (ANN) on GIST1M [6] and SIFT1M [6]. We randomly choose 100K vectors as training set, 5K of the rest as queries, and the remaining as database. The first 1000 nearest neighbors computed by the exhaustive, linear scan based on the Euclidean distance are defined as the ground truth. The performance is measured by mean Average Precision (mAP) and recall. The compared methods are:

- PCA-PQ: For comparison, the original data are rotated by PCA projection.
- PQ [6]: Do PQ procedure directly on the original data.
- rPQ: The original data are rotated by a random orthogonal projection.
- UVPQ-LP: The uniform variance projection is calculated based on lift and project algorithm.
- UVPQ-GF: The uniform variance projection is calculated based on gradient flow algorithm.

Table 1. Experiment on GIST1M dataset in term of 1000-NN mAP and Recall@1000.

	1000-NN mAP						Recall@1000					
# bits	16	32	64	128	256	512	16	32	64	128	256	512
PCA-PQ	0.0063	0.0082	0.0114	0.0181	0.0305	0.0608	0.0550	0.0615	0.0713	0.0900	0.1212	0.1825
PQ	**0.0153**	0.0224	0.0448	0.0804	0.1598	0.2851	**0.0793**	0.1154	0.1635	0.2217	0.3184	0.4212
rPQ	0.0110	0.0202	0.0415	0.0820	0.1564	0.2884	0.0777	0.1123	0.1562	0.2232	0.3168	0.4235
UVPQ-LP	0.0111	0.0228	**0.0450**	0.0862	0.1600	0.2898	0.0778	**0.1167**	**0.1651**	0.2304	0.3203	**0.4477**
UVPQ-GF	0.0077	**0.0231**	0.0411	**0.0868**	**0.1626**	**0.2909**	0.0742	0.1162	0.1632	**0.2314**	**0.3226**	0.4473

Table 2. Experiment on SIFT1M dataset in term of 1000-NN mAP and Recall@1000.

	1000-NN mAP						Recall@1000					
# bits	16	32	64	128	256	512	16	32	64	128	256	512
PCA-PQ	0.0385	0.0807	0.1637	0.3540	0.6527	0.9000	0.1502	0.2132	0.3130	0.4907	0.7167	0.9087
PQ	**0.0513**	0.1219	0.2617	0.4631	0.7205	0.9098	**0.1782**	**0.2903**	0.4235	0.5805	0.7694	0.9071
rPQ	0.0420	0.1149	0.2542	0.4577	0.7179	0.9126	0.1706	0.2789	0.4155	0.5753	0.7660	0.9182
UVPQ-LP	0.0426	**0.1225**	**0.2653**	**0.4835**	**0.7276**	**0.9221**	0.1727	0.2876	**0.4246**	**0.5949**	**0.7730**	**0.9278**
UVPQ-GF	0.0407	0.1154	0.2595	0.4650	0.7243	0.9211	0.1684	0.2798	0.4214	0.5836	0.7702	0.9269

For all the experiments, we use the asymmetric distance computation (ADC) where only vectors in the database are quantized and we set k=256 in every subspace.

Table 1 and Table 2 show the 1000-NN mAP and Recall@1000 on GIST1M and SIFT1M. The code length means m*log$k$. Since the size of the codewords k is fixed to 256, we experiment with m set to 2, 4, 8, 16, 32, and 64. The corresponding code lengths are 16, 32, 64, 128, 256, and 512 bits. Our UVPQ method performs best across the tested bit lengths ranging from 32 bits to 512 bits on both datasets. Not surprisingly, PCA-PQ always achieves the worst results since its largest subspace distribution difference (SDD). The rPQ method gets good results, which may be due to the random projection balance the average distribution variance of the subspaces and SDD of the whole space to some extent. But rPQ is not better than our UVPQ method since UVPQ can guarantee the uniform variance in every subspace and this will reduce the quantization distortion in the subspaces.

The advantage of UVPQ on GIST1M and SIFT1M is not obvious. The main reason is that the SDD of the original space is small on both datasets, which will reduce the superiority of UVPQ over PQ. The performance improvement of our methods can be greater when the SDD of the original space is larger, like the case of PCA-PQ where PQ will achieve bad results on PCA-projected data. Since UVPQ gets the same results however the data rotate, the comparison between UVPQ and rPQ, PCA-PQ can be seen as the comparison between UVPQ and PQ on the random rotated and PCA-projected GIST1M and SIFT1M dataset where UVPQ performs significantly better than PQ.

Conclusion

In this work, we propose a new approximate nearest neighbor (ANN) search approach which we call UVPQ. It is simple and effective. Extensive experiments have demonstrated the superiority of our method. But to solve the objective function Eq. 5, we compel the uniform variance projection $\boldsymbol{P}$ to meet the requirement that every dimension of all the subspaces after projection has the same variance, which is not necessary. Actually, we only demand of the ADV of every subspace after projection to be the same. In the future, we will tackle this issue.

Acknowledgment

This work has been supported by the National Key Technology R&D Program of China under Grant No. 2012BAH04F02, 2012BAH88F02, 2013BAH61F01 and 2013BAH63F01 and the International S&T Cooperation Program of China under Grant No. 2013DFG12980.

References

[1] J. Freidman, J. Bentley, and A. Finkel, *ACM Transactions on Mathematical Software*, Vol. 3 (1977), p. 209.

[2] A. B. Torralba, R. Fergus, and Y. Weiss, in: *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2008, p. 1.

[3] Q.-Z. Guo, Z. Zeng, S. Zhang, Y. Zhang, and F. Wang, in: *Proc. of IEEE Conference on Multimedia and Expo*, 2013, p. 1.

[4] T. Trzcinski, V. Lepetit and P. Fua, *P. R. Letters*, Vol. 33 (2012), p. 2173.

[5] R. Gray, *ASSP Magazine*, *IEEE*, Vol. 1 (1984), p. 4.

[6] H. Jegou, M. Douze, and C. Schmid, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 33 (2011), p. 117.

[7] W. Kong, and W.-J. Li, in: *Advances in Neural Information Processing Systems*, 2012, p. 1646.