Tree Quantization for Large-Scale Similarity Search and Classification

Artem Babenko[1], Victor Lempitsky[2]
[1]Yandex. [2]Skolkovo Institute of Science and Technology (Skoltech).

As very large datasets of high-dimensional vectors proliferate, machine learning, computer vision, and information retrieval systems that work with such datasets increasingly rely on lossy vector compression or hashing schemes. A crucial requirement for these schemes is the ability to evaluate distances and scalar products between compressed and uncompressed vectors efficiently and without explicit decompression. At the moment, systems that are based on the *product quantization*[3] are often preferred.

Given a dataset of vectors in $\mathcal{R}^D$, product quantization starts by splitting the vector dimensions into M groups. Each dimension group is then quantized separately and independently from others using codebooks of small size (most often 256 codewords), whereas codewords in the codebooks have the dimension D/M. In the PQ compression scheme, an input vector is approximated as a concatenation of M codewords (one codeword from each codebook). Product quantization implicitly relies on the limited amount of correlation between the dimension groups, since each codebook is learned independently from others. The encoding process within PQ is very simple and fast, and the computation of scalar product and distances between a large number of PQ-compressed vectors and an uncompressed vector can be implemented very efficiently using look-up tables.

Recently, [1] have proposed an alternative compression scheme called *additive quantization* (AQ) that pushes the coding accuracy of PQ-based methods even further. Similarly to PQ, AQ maintains a set of M codebooks. However, the codewords within the codebooks are full-length, i.e. D-dimensional. During the compression stage AQ represents a vector as a sum of M codewords (one codeword from each codebook). The vector code is thus the same as within PQ, i.e. M codeword numbers. The additive nature of the compression means that the evaluation of scalar products between AQ-compressed and uncompressed vectors can use the same look-up table trick and is thus very fast. Evaluation of Euclidean distances takes slightly more time or an extra byte of memory but is still efficient. In general, AQ achieves a significant boost in coding accuracy over PQ, which can be explained by the lack of low-correlation assumption between dimension groups. Furthermore, AQ codebooks possess an increased number of parameters that can be adjusted at the codebook learning stage in order to fit the data distribution.

The main limitation of the AQ-compression is the inefficiency of the encoding step. As shown in [1], finding the optimal combination of the codebook vectors is equivalent to the MAP-inference in the fully-connected Markov random field with unstructured and highly non-submodular pairwise potentials. As reported in [1], none of the standard MRF optimization methods work well and therefore a special kind of Beam Search is used, which is able to find approximate codings resulting in lower coding error than PQ-compression. Still, this approximate inference takes orders of magnitude more time than PQ encoding, and can be prohibitively slow for many practical applications, especially when online encoding of new vectors is needed.

Here, we propose a new coding scheme called Tree Quantization (TQ) that belongs to the same family as PQ and AQ. Similarly to PQ and AQ, TQ maintains a set of M codebooks and, similarly to AQ, it encodes a vector as a sum of M codewords from different codebooks. The TQ-code for a vector is thus, once again, a set of M codeword numbers. The difference from AQ lies in the special structure that TQ imposes onto its codebooks. The encoding is based on a tree graph (the *coding tree*), where vertices correspond to codebooks, while each of the D dimensions is assigned to an edge. Each codebook then encodes only the dimensions that are assigned to edges that are incident to the vertex corresponding to this codebook (1). All other dimensions are then fixed to zero for all codewords in a given codebook.

The encoding process within the tree quantization is performed via the MAP-inference in a tree-shaped model, and is therefore exact and efficient.

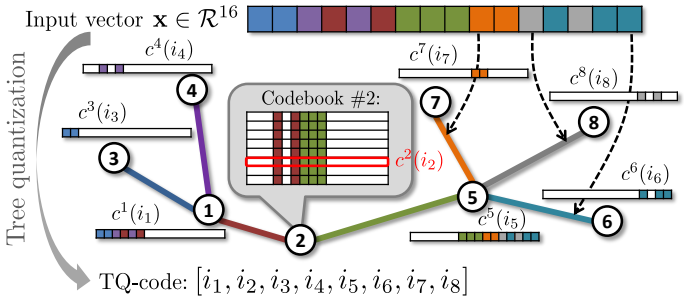

Figure 1: **Tree quantization** encoding for D-dimensional vectors (here D=16). Each dimension is assigned to an edge of the **coding tree** (the assignment is color-coded). Each of the M=8 vertices of the coding tree contains a codebook (shown for vertex #2). Each codebook encodes dimensions from the incident edges (also color-coded). An input vector $\mathbf{x}$ is then represented as a sum of M codewords $c^t(i_t)$ from vertex codebooks (shown as rectangles with active dimensions color-coded; the position of the codeword $c^2(i_2)$ within the second codebook is highlighted in red).

The most interesting part in the TQ framework is the codebook learning stage. Standard quantization, product quantization, and additive quantization all use k-means like processes to learn their codebooks, which alternate the encoding steps with the codebook-reestimation steps, during which the codebook assignments of the training vectors are kept fixed. Crucially, we demonstrate that TQ can follow the same scheme, and that given the codebook assignments of the training vectors, it is possible to estimate (i) the tree structure, (ii) the dimensions to edges assignments, and (iii) the codewords, all jointly and in a globally optimal way. Such global estimation requires solving an integer linear program (ILP), which in our experiments was always solvable to optimality using a modern ILP solver for D and M that were typically used in previous works.

While the AQ scheme is theoretically more powerful than TQ as it has more parameters (MKD) that can fit the distribution and potentially achieve lower reconstruction error, it is hindered by the slowness and inexactness of the encoding. As was shown in the [1] the problem of AQ encoding is equivalent to the problem of an inference in a fully-connected MRF in probabilistic modeling. The usage of TQ is then an analogy of Chow-Liu tree approximation for this MRF. Similarly to Chow-Liu tree, TQ can capture second-order correlations while remaining tractable for inference.

We evaluate TQ in terms of coding errors and within the contexts of the nearest neighbor search and classification. We compare TQ with the recent PQ-based methods: Optimized Product Quantization (OPQ) [2, 4], Additive Quantization (AQ) [1] and Composite Quantization (CQ) method [5]. Our experiments show that the accuracy of Tree Quantization, and in particular its "optimized" variant (OTQ) exceeds that of the OPQ. This advantage is more pronounced for descriptors with easily identifiable parts (such as spatial bins within SIFT, or separate GMM components within Fisher vectors).

Overall, TQ provides a combination of high compression accuracy and fast encoding that is attractive for retrieval and classification systems.

[1] Artem Babenko and Victor Lempitsky. Additive quantization for extreme vector compression. In *CVPR*, 2014.

[2] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. Optimized product quantization for approximate nearest neighbor search. In *CVPR*, 2013.

[3] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *TPAMI*, 33(1), 2011.

[4] Mohammad Norouzi and David J. Fleet. Cartesian k-means. In *CVPR*, 2013.

[5] Ting Zhang, Chao Du, and Jingdong Wang. Composite quantization for approximate nearest neighbor search. In *ICML*, 2014.

This is an extended abstract. The full paper is available at the Computer Vision Foundation webpage.