Memory vectors for similarity search in high-dimensional spaces

Ahmet Iscen, Teddy Furon, Vincent Gripon, Michael Rabbat, and Hervé Jégou

Abstract—We study an indexing architecture to store and search in a set of high-dimensional vectors. This architecture is composed of several memory units, each of which summarizes a fraction of the database by a single representative vector. The potential similarity of the query to one of the vectors stored in the memory unit is gauged by a simple correlation with the memory unit’s representative vector. This representative optimizes the test of the following hypothesis: the query is independent from any vector in the memory unit vs. the query is a simple perturbation of one of the stored vector.

Compared to exhaustive search, our approach finds the most similar database vectors significantly faster without a noticeable reduction in search quality. Interestingly, the reduction of complexity is provably better in high-dimensional spaces. We empirically demonstrate its practical interest in a large-scale image search scenario. Our evaluation is carried out on standard benchmarks for large-scale image search and uses off-the-shelf state-of-the-art descriptors.

Index Terms—High-dimensional indexing, image indexing, image retrieval.

1 INTRODUCTION

We consider the problem of similarity search for large databases. In this context, many papers report how the curse of dimensionality makes indexing ineffective [18], [25]. The recent paper [18] describing and analyzing the popular FLANN method experimentally observes that even this state-of-the-art method performs poorly on synthetic high-dimensional vectors, and the authors conclude that “random datasets are one of the most difficult problems for nearest neighbor search.”

Some strategies have been proposed to (partly) overcome this problem. For instance, the vector approximation file [25] first relies on exhaustive search with approximate measurements and then computes the exact similarities only for a subset of vectors deemed of interest. The cosine sketch [6] approximates cosine similarity with faster Hamming distance. Other works like spectral hashing [26], Euclidean sketches [10], product quantization [15] and inverted multi-index [2] also rely on compact codes to speed up neighbor search while compressing the data. An interesting strategy is the Set Compression Tree [1], which uses a structure similar to a k-d tree to compress a set of vectors into an extremely compact representation. Instead of trying to explicitly return the vector, the algorithm uses the structure to determine if a vector similar to a given query has been stored. Again, this method is dedicated to small dimensional vectors (its authors recommend the dimension be smaller than \(\log_2(N)\) where \(N\) is the size of the database) so that it is used in conjunction with a drastic dimension reduction PCA to cope with classical computer vision descriptors.

This paper proposes a similarity search approach specifically adapted to high-dimensional vectors such as those recently introduced in computer vision to represent images [16], [21]. The proposed indexing architecture consists of memory units, each of which is associated with several database vectors. A representative, called memory vector, is produced for each memory unit and defined such that, by performing a single inner product with a new query, one can quickly and reliably determine whether or not at least one similar vector is stored in this unit.

Our problem is similar to the descriptor pooling problem in computer vision, but at a higher level. Many successful descriptors, such as BOV [7], [24], VLAD [16], FV [21], and EMK [5], encode and aggregate a set of local features into global representations. Although max-pooling and sum-pooling methods have been used traditionally, recent works [17], [19] try to optimize the aggregation step to yield a constant similarity between aggregated local features and the global representation. We solve a similar problem at a higher-level; instead of aggregating local features for a global image representation, we aggregate global representations into group representations to perform efficient image search.

We make the following contributions.

1) First, we focus on the design of a single memory vector. We formalize the similarity of a query with the vectors of one memory unit as a hypothesis test. We derive the optimal representative vector under some design constraints and show how to compute it in an online manner.

2) Second, we propose and analyze different ways to assign vectors to memory units, such as random assignment and weakly supervised assignment.

3) Third, we provide a theoretical and experimental analysis of different design and assignment strategies.

4) Finally, we discuss strategies to search vectors with such an architecture in image search.

We apply this approach to real datasets: descriptors (vectors describing images) extracted with the most recent state-of-the-art algorithm in computer vision [17].

This paper is organized as follows. Section 2 introduces notation, the detection framework considered in the paper, and
derives the optimal solution under some assumptions. Section 3 extends the problem to memory vectors, optimizing the design of the indexing system. The experimental section 4 evaluates our approach on standard benchmarks for image search, namely the Oxford building [22], Inria Holidays [14] and UKB [20] datasets. Our results show the potential of our approach for this application.

2 MEMORY VECTORS

A memory unit $X$ is defined as a set of $n$ vectors $\{x_1, \ldots, x_n\}$ of dimension $d$. Our objective is to produce a representative, so-called memory vector, such that, given a query vector $Y$, we can efficiently perform a “similarity” test: is $Y$ a quasi-copy of, or similar to, at least one of the vectors of the memory unit? For the sake of analysis, this section assumes that all vectors follow a uniform distribution on the $d$-dimensional unit hypersphere. The similarity between two vectors $x$ and $y$ is the inner product $x^\top y$. Note that, under our distribution hypothesis, this inner product is one for a perfect match and centered on zero for two independently generated vectors. We model the query as a random vector $Y$ distributed according to one of the two laws:

- Hypothesis $H_0$: $Y$ is not related to any vector in $X$. $Y$ is then uniformly distributed on the unit hypersphere.
- Hypothesis $H_1$: $Y$ is related to one vector in $X$, say $x_1$ without loss of generality. We write this relationship as $Y = \alpha x_1 + \beta Z$, where $Z$ is a random vector orthogonal to $x_1$ and $\|Z\| = 1$. This means that $Y$ is more similar to $x_1$ as $\alpha$ gets closer to 1. We have $\alpha^2 + \beta^2 = 1$ because $\|Y\| = 1$.

We consider a representative vector satisfying the following design constraints. First, the set of vectors $X$ is summarized by a unique $d$-dimensional vector $m(X) \in \mathbb{R}^d$, called the memory vector and denoted by $m$ when not ambiguous. Then, membership of $Y$ to $X$ is tested by thresholding the inner product $m^\top Y$.

2.1 Sum-memory vector: analysis

A very simple way to define the memory vector is

$$m(X) = \sum_{x \in X} x, \quad (1)$$

where we assume that $X$ is composed of $n$ different vectors. Albeit naive, this strategy offers some insights when considering high-dimensional spaces.

Appendix A derives the pdf of the score $m^\top Y$ under $H_0$ when $m$ is a known vector. This score has expectation 0 and variance $\|m\|^2/d$, and it is asymptotically distributed as $\mathcal{N}(0, \|m\|^2/d)$ as $d \to \infty$. This gives an approximate pdf of $m^\top Y$ under $H_0$. In contrast, under $H_1$, the inner product equals

$$m^\top Y = \alpha + \alpha m(X')^\top x_1 + \beta m(X')^\top Z, \quad (2)$$

with $X' = X - \{x_1\}$. This shows two sources of randomness: the interference of $x_1$ with the other vectors of $X$, and with the noise vector $Z$. Assuming that $Y$ is statistically independent of the vectors in $X'$ (this implies that the vectors of $X'$ are mutually independent), we have

$$\mathbb{E}_Y[m^\top Y|H_1] = \alpha,$$

$$\sqrt{\mathbb{V}_Y[m^\top Y|H_1]} = \|m(X')\|^2/d. \quad (3)$$

Assuming that $X$ is composed of $n < d$ statistically independent vectors on the unit hypersphere also gives $\mathbb{E}_X[\|m(X')\|^2] = n$ and $\mathbb{E}_Y[\|m(X')\|^2] = n - 1$. To summarize, for large $d$, we expect the following distributions:

$$H_0: \quad m^\top Y \sim \mathcal{N}(0, n/d), \quad (4)$$

$$H_1: \quad m^\top Y \sim \mathcal{N}(\alpha, (n-1)/d). \quad (5)$$

Making a hard decision by comparing the inner product to a threshold $\tau$, the error probabilities (false positive and false negative rates) are given by

$$P_{fp} \approx 1 - \Phi\left(\frac{\tau \sqrt{n/d}}{\alpha}\right), \quad (6)$$

$$P_{fn} \approx \Phi\left(\frac{\tau - \alpha}{\sqrt{n/(n-1)}}\right), \quad (7)$$

where $\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^x e^{-t^2/2} dt$.

The number of elements one can store in a sum-memory vector is linear with the dimension of the space when vectors are drawn uniformly on the unit hypersphere. This construction is therefore useful for high-dimensional vectors only, as opposed to traditional indexing techniques that work best in low-dimensional spaces. Figure 1 depicts Receiver Operating Characteristic (ROC) curves for two ratios $n/d$ and two values of $\alpha$. As expected the test performs better when $\alpha$ is closer to 1 and when $n \ll d$.

If the vectors are pair-wise orthogonal, the dominant source of randomness (the interference between $x_1$ and vectors of $X'$) is cancelled in (2). The variance under $H_1$ reduces to $\beta^2(n-1)/d$. This prevents any false negatives if $\beta \to 0$. We further exploit this intuition that orthogonality helps in the next section.

2.2 Optimization of the hypothesis test per unit

We next consider optimizing the construction of the memory vector when the set $X'$ consists of a batch of $n$ vectors known at encoding time. Denote the $d \times n$ matrix $X = [x_1, \ldots, x_n]$. We impose that, for all $i$, $x_i^\top m(X') = 1$ exactly and not only in expectation, as assumed above. In other words, $X^\top m = 1_n$ where $1_n$ is the length-$n$ vector with all entries equal to 1. Achieving this, when $Y = x_1$, we eliminate the interference with the remaining vectors in $X'$ which was previously the dominant source of noise. In other words, under $H_1$, Eq. (2) becomes

$$m^\top Y = \alpha + \beta m^\top Z. \quad (8)$$

Under $H_0$, the variance of the score remains $\|m\|^2/d$. Therefore, the norm of the memory vector is the key quantity determining the false positive probability.

We thus seek the representation $m$ minimizing the energy $\|m\|^2$ subject to the constraint that $X^\top m = 1_n$. If multiple solutions exist, the minimal norm solution is given by the Moore-Penrose pseudo-inverse [23]:

$$m^* = (X^+)^\top 1_n. \quad (9)$$

Since $n < d$, $m^* = X(X^\top X)^{-1} 1_n$.

If no solution exists, $m^*$ is a minimizer of $\|X^\top m - 1_n\|$. This formulation amounts to treating the task of image search as a linear regression problem [4] with the objective of minimizing

$$\frac{1}{2} \|X^\top m - 1_n\|^2 \quad (10)$$
over \( m \). Taking the gradient, setting it equal to zero, and solving for \( m \) gives back \( m^\star \). When possible, and for large \( d \), this new construction leads to the distributions:

\[
H_0 : \quad m^\top Y \sim \mathcal{N}(0, \|m^\star\|^2/d) \\
H_1 : \quad m^\top Y \sim \mathcal{N}(\alpha, \beta^2\|m^\star\|^2/d).
\]

The major improvement comes from the reduction of the variance under \( H_0 \) for small values of \( \beta^2 \), i.e., \( \alpha \leq 1 \). Appendix B shows that, if the vectors of \( \mathcal{X} \) are uniformly distributed, then in expectation \( \|m^\star\|^2 \) is larger than the square norm of the naive sum representation from Section 2.1. The reduction of the variance under \( H_1 \) comes at the price of an increase of the variance under \( H_0 \). However, this increase is small if \( n/d \) remains small. For large \( d \), we have

\[
\mathbb{P}_{fp} \approx 1 - \Phi \left( \tau \sqrt{\frac{1}{n} - 1} \right),
\]

\[
\mathbb{P}_{fn} \approx \Phi \left( \tau - \alpha \sqrt{\frac{1}{n} - 1} \right).
\]

Note that \( \beta = \sqrt{1 - \alpha^2} \) is a decreasing function of \( \alpha \). Therefore, if \( \tau < \alpha \), \( \mathbb{P}_{fn} \) is a decreasing function of \( \alpha \). In particular \( \mathbb{P}_{fn} \to 0 \) when \( \alpha \to 1 \) as claimed above. In contrast to the naive sum approach from Section 2.1, there is no longer false negative when the query \( Y \) is exactly one of the vectors in \( \mathcal{X} \). This holds for any value of \( \tau < 1 \) when \( \alpha = 1 \), so that the false positive rate can be as low as \( 1 - \Phi(\sqrt{d/n - 1}) \).

**Remark.** This solution is identical (up to a regularization) to the “generalized max-pooling” method introduced to aggregate local image descriptors [19]. However in our case the aggregation is performed on database side only. Our solution is moreover theoretically grounded by a hypothesis test interpretation.

### 2.3 Online processing

Section 2.2 resorts to a linear regression problem assuming that all the vectors are stored in the dataset. In practice, the vectors to be included in the memory unit are not necessarily known in advance. It then becomes meaningful to propose an online strategy.

We assume that the number of vectors to store is smaller than the dimensionality of the vectors. Intuitively, when storing a new vector we should (a) not change the test scores of previously stored vectors and (b) ensure that the test score of the newly added vector is one. Algorithm 1 guarantees these properties by construction.

To prove this, we consider a sum-memory vector \( m \) obtained from a set of vectors \( \mathcal{X} \). Denote by \( \mathcal{X}^\prime \subseteq \mathcal{X} \) a previously stored vector and \( x \) a new vector to be added. Algorithm 1 keeps a set \( R \) of the residuals of vectors already stored. The Gram-Schmidt step guarantees that \( \mathcal{X}^\prime \) is in the span of \( R \), since \( \text{span}(\mathcal{X}) = \text{span}(R) \).

**Algorithm 1** Online optimization of a sum-memory vector. At initialization, \( m = 0 \) and \( R = \emptyset \).

At reception of a new vector \( x \) with unit norm

\[
r \leftarrow x - \sum_{r \in R} \frac{r^\top r}{r^\top x} r \\
R \leftarrow R \cup \{r\}
\]

If \( r^\top x \neq 0 \), then

\[
m \leftarrow m + \frac{1 - m^\top x}{r^\top x} r
\]

Thus by construction we have \( x^\top r = 0 \). It follows that the test score of \( x^\prime \) is

\[
x^\top m + \frac{1 - m^\top x}{r^\top x} r^\top x = x^\top x = 1,
\]

which shows that the test score of \( x^\prime \) is not affected when adding a new vector. The second property (b) is verified if we assume that the new vector \( x \) is not in the span of \( \mathcal{X} \), which is satisfied almost surely if \( d > n \). In this case, the obtained residual vector \( r \) is nonzero, and we have

\[
m^\top x + \frac{1 - m^\top x}{r^\top x} r^\top x = 1,
\]

which proves the second property (b). This shows that the memory vector \( m \) produced by Algorithm 1 satisfies \( X^\top m = 1_n \). Among the solutions satisfying this linear equation, the one from (9) is the only one such that \( m \in \text{span}(\mathcal{X}) \). Since the memory vector produced by the online algorithm also shares this property, the constructions of \( m \) with (9) and Algorithm 1 are equivalent.

Including the same vector several times does not change the value of \( m \), since \( r \) is \( O_d \) in such a case. The resulting memory vector produced by the online algorithm does not depend on the order in which the data vectors are processed. The complexity of Algorithm 1 is \( O(dn) \) operations. The global complexity after storing all elements of \( \mathcal{X} \) is therefore \( O(dn^2) \).

### 2.4 Weakly supervised assignment

We now analyze a scenario where the vectors packed in the same unit are random but no longer uniformly distributed over the hypersphere: There is some correlation among them. We assume that the vectors in a memory unit are uniformly distributed over a spherical cap (see Appendices D and E). This models an assignment done using k-means to process batches of database vectors (see Sec 4.2).

We derive the same analysis as in the previous subsection with expectations and variances which now depend on the angle of the spherical cap. These expressions are complex and their derivation is detailed in the above mentioned appendices. In summary, this shows that the Kullback-Leibler distance between both distributions increases as the correlation between the vectors stored in...
the same unit increases. In other words, identifying the positive memory units becomes easier when we assign correlated vectors to the same memory unit. Interestingly, this mechanism helps the sum construction more than pinv, so that when the vectors are very correlated, both construction perform equivalently.

3 MEMORY VECTORS AND APPLICATION

We consider now an application scenario where we need to store a large number $N$ of vectors and perform similarity search. One memory vector is not sufficient to achieve a reliable test. We therefore consider an architecture that consists of several memory vectors. The search strategy is as follows. A given query vector is compared with all the memory vectors. Then we compare the query with the vectors stored in the memory units associated with the high responses, i.e., those likely to contain a similar vector.

The main question that we consider is: how many vectors per memory unit should we store to achieve the best complexity/accuracy trade-off?

3.1 Random assignment

We suppose that the $N$ vectors in the database are randomly grouped into $M$ units of $n$ vectors; $N = nM$. We aim at finding the best value for $n$. When the query is related to the database (i.e., under $H_1$), we make the following assumption: $\alpha_0 < \alpha < 1$, and we fix the following requirement: $P_{\text{fp}} < \epsilon < 1/2$. Since $P_{\text{in}}$ is an increasing function of $\alpha$, we need to ensure that $P_{\text{in}}(\alpha_0) = \epsilon$. This gives us the following threshold $\tau$:

$$\tau = \mu_{H_1} + \sigma_{H_1} \Phi^{-1}(\epsilon),$$  \hspace{1cm} (17)

with $\mu_{H_1}$ and $\sigma_{H_1}$ being the expectation and the standard deviation of $m_j^\top Y$ under $H_1$. Note that $\Phi^{-1}(\epsilon) < 0$ because $\epsilon < 1/2$ so that $\tau < \alpha$. The probability of false positive equals $1 - \Phi(\tau/\sigma_{H_0})$ which depends on $n$, denoted by $P_{\text{fp}}(n)$. This is indeed an increasing function for both memory vector constructions. Now, we decide to minimize the total computational cost $C_{H_0}$ when the query is not related. We need to compute one inner product $m_j^\top y$ per unit, and then to compute $n$ inner products $x_i^\top y$ for the units giving a positive detection. In expectation, there are $M \cdot P_{\text{fp}}(n)$ such units, and so

$$C_{H_0} = M + M \cdot P_{\text{fp}}(n) \cdot n = N(n^{-1} + P_{\text{fp}}(n)).$$  \hspace{1cm} (18)

The total cost is the sum of a decreasing function ($n^{-1}$) and an increasing function ($P_{\text{fp}}(n)$). For the random assignment strategy, there is a tradeoff between having a few big units ($n$ large) and many small units ($n$ small). Fig. 2 illustrates this tradeoff for different values of $\alpha_0$. It is not possible to find a closed form expression for the cost minimizer $n^*$. When $\alpha_0$ is close to 1, the threshold is set to a high value, producing reliable tests, and we can pack many vectors into each unit; $n^*$ is large allowing a huge reduction in complexity. Even when $\alpha_0$ is as small as 0.5, $n^*$ is small but the improvement remains significant. In the setup of Fig. 2, the proposed approach has a complexity that is less than one tenth of that of searching through all database vectors (equivalent to $n = 1$). This is confirmed by our experiments on real data presented in Section 4. The pinv construction performs better than the sum all the more so as $\alpha_0$ is bigger as explained in Sect. 2.2. However, in order to increase the efficiency, we introduce an additional $O(Md) = O(dN/n)$ memory overhead using memory vectors.

3.2 Weakly supervised assignment

We generalize this analysis to the case where the vectors in a memory unit are correlated by using the expression of expectations and variances given in Appendices D and E. However, this does not exactly reflect reality: When $k$-means is used to assign vectors to memory units, the number $n$ of vectors per memory unit is no longer constant.

Up to this approximation, Fig. 2 shows that both constructions sum and pinv perform better as the inner correlation increases, but more surprisingly, they perform equivalently.

4 EXPERIMENTS

This section shows that memory vectors perform extremely well with real data. We consider the two constructions for memory vectors introduced in Section 2: naive sum and the version optimized using the pseudo-inverse, pinv.

4.1 Evaluation Protocol

Datasets. We use the Inria Holidays [14], Oxford5k [22], and UKB [20] image datasets in our experiments. Additionally, we conduct a large scale experiment by adding images from the Flickr1M [22] dataset to the Holidays dataset.

Descriptors. We use the state-of-the-art triangular embedding descriptor [17], denoted by $\phi_D$. We use the off-the-shelf reference implementation provided by the authors, which can be found online. Each image is represented by a feature vector. The only difference is that we do not apply the “powerlaw normalization” to better illustrate the benefit of the pinv technique for the memory vector construction compared to the sum (when applying the powerlaw normalization, both designs perform equally well since the vectors are nearly orthogonal). Ultimately, we have $d = 8064$ (or $d = 1920$) dimensional feature vectors for each image, obtained by using a vocabulary of size 64 (resp. 16).

We also experiment using deep learning features provided by Babenko et al. [3]. As explained in their paper, the performance for the UKB dataset drops with adapted features trained on the Landmarks dataset. Therefore, we use the original neural codes trained on ILSVRC for the UKB dataset, and the adapted features for Holidays and Oxford5k ($d = 4096$ dimensions).

Performance. Unless stated otherwise, our experimental analysis follows the standard image retrieval protocol where each image is represented by a feature vector and the ground truth is based on the visual similarity. The goal is to return visually similar images for a given query image. The similarity of two images is measured by the cosine of their descriptor vectors, and the images are ordered accordingly. We adopt the performance measure defined for each benchmark (mAP or 4-recall@4, the higher the better).

Complexity. Following Section 3, we measure the similarity between the query and memory vectors, and re-rank all the images in positive memory units with similarity above a threshold $\tau$. We characterize the complexity of the search per database vector by:

$$C_{H_1}(\tau) = M + n \cdot M_p(\tau),$$  \hspace{1cm} (19)

where $M_p(\tau)$ is the number of positive memory units per query that are above the threshold $\tau$. We measure the complexity ratio $C_{H_1}(\tau)/N$ and the retrieval performance for different values

1. http://www.tinyurl.com/democratic-kernel
of the threshold \( \tau \). For large \( \tau \), no memory vector gives a positive output, resulting in \( C_{H_1}(\tau)/N = M \) and no candidate is returned. As \( \tau \) decreases, more memory vectors trigger reranking.

### 4.2 Comparison of \textit{pinv} vs \textit{sum}

**Cosine similarity**-based ground truth is one way to measure the performance since it is directly related with the model we considered previously. For each query vector we deem a database vector as relevant if their cosine similarity is greater than \( \alpha_0 \), regardless of their visual information. To have enough ground-truth vectors, we perform this experiment on the Holidays+Flickr1M dataset using various \( \alpha_0 \) values, and report the ROC curve in Fig. 3. The performance gain of \textit{pinv} over \textit{sum} for memory vector design is significant when \( \alpha_0 \) is higher, but it decreases with \( \alpha_0 \).

**Assignment strategies.** In Section 2, we proposed random and weakly supervised assignment strategies. For the latter, we adopt the spherical \( k \)-means clustering [8] with \( k = M \) instead of random assignment. For the \textit{pinv} aggregation case, we modify the update stage of spherical \( k \)-means to represent clusters using \textit{pinv} instead of \textit{sum}. However, when we use a such an assignment method, the number of vectors per memory vector is not evenly distributed. To have comparable similarity scores, we normalize the similarity score for each memory vector by dividing it by the norm of the memory vector. This mapping is done every time we have a dot product between a memory vector and a query.

Fig. 4 shows that this strategy brings a significant improvement compared to the random assignment. By grouping similar vectors in a single memory unit, 1) the separation between the distributions of positive and negative unit scores increases, yielding a more reliable test, 2) having more than a single match in a memory unit is more likely. Hence, when we rerank vectors in a single positive memory unit, we are likely to identify more than one match. We have explained the effect of the first point by deriving a simple statistical model used in Sect. 2.4 and 3.2. We now show both effects experimentally in Figures 6 and 7.

The performance of both assignment methods for different \( M \) is shown in Fig 5. Observe that spherical \( k \)-means with \textit{sum} performs well even for small \( M \), which is not the case with random assignment. This confirms that clustering similar vectors in the same memory unit helps \textit{sum} more than \textit{pinv} as already theoretically shown for simple assumptions. The difference between the two constructions is more visible for small \( M \). We use \( M = N/10 \) for the rest of our experiments.

Unless stated otherwise, in the remainder of the paper we adopt spherical \( k \)-means for the assignment of vectors to memory vectors. Although it brings improvement in performance, using a clustering approach has some drawbacks. This strategy might be impossible, such as in the case of streaming data, where one would need to use random assignment. It may not be tractable for very large databases, such as the Holidays+Flickr1M dataset. Section 4.3 discusses another assignment strategy for large datasets. At last, memory units no longer have the same number of vectors.

### 4.3 Versatility

This subsection demonstrates that indexing memory vectors is compliant with usual mechanisms in the image search literature.

The **dimensionality** of the descriptor linearly impacts the efficiency of any system. Dimensionality reduction with PCA is one way to improve this point. Our method is compatible with dimensionality reduction as shown in Fig. 8, where we reduce the vectors to \( d' = 1024 \) components. It achieves performance comparable to the baseline with lower computational complexity.

We also apply our method to features learned with deep learning (\( d = 4096 \)). Fig. 9 shows that the reduction in complexity also applies when using high performance deep learning features, Note that, under weak supervision (with spherical \( k \)-means), \textit{sum} provides better performance than \textit{pinv} in Oxford5k.

Finally, we use low-dimensional SIFT features (\( d = 128 \)) from the SIFT10k dataset [15] in Figure 10. The performance is measured by recall@10. We introduce the \textit{pinvreg} construction for \( k \)-means assignment for this experiment; this refers to the regularized pseudo-inverse or ridge regression. We use this due to the fact that there may be more items in a memory unit than the dimensionality of the feature vectors. In that case, \textit{pinv} does not have a unique solution. Therefore, we add a regularization term \( \lambda \| \mathbf{m} \|_2^2 \) in (10) with \( \lambda = 0.1 \). We obtain similar scores for random assignment, but \textit{sum} and \textit{pinvreg} slightly perform better than \textit{pinv} with clustering assignment.

**Compact Codes** are another way to increase efficiency. We reduce the dimensionality of the descriptor vectors to \( d' = 1024 \) and binarize them by taking the \textit{sign} of each component, in the spirit of cosine sketches [6]. In the asymmetric case [11], [13], only
Comparison between \textit{sum} and \textit{pinv} performs better than \textit{sum} binary codes. For the symmetric case, the hypothesis test is better due to grouping similar vectors together. However, by clustering the data, this is improved especially for higher-ranked units. This shows that the expected number of matches per positive unit is bigger due to grouping similar vectors together.

Figures 11 and 12 show the performance when using compact spherical k-means instead of exhaustive search; see Fig. 13.

Fig. 4. Image retrieval performance using different assignment methods and visual similarity. K-means variants bring significant improvement. \textit{reg. kmeans} is the regular k-means algorithm, i.e., mean of vectors.

Fig. 5. \textbf{Left:} Retrieval performance using random assignment and \textit{pinv}. \textbf{Middle:} Retrieval performance using spherical k-means and \textit{sum}. \textbf{Right:} Comparison between \textit{sum} and \textit{pinv} using spherical k-means for $M = N/200$.

Fig. 6. Probability of memory units containing at least one match with respect to their rank. For both random and k-means, there is a high probability of retrieving a match in most similar memory units, but it decreases faster with k-means as we verify other memory units. This shows that the hypothesis test is better due to grouping similar vectors together.

Fig. 7. The average number of matches given that the unit is positive. Using random assignment, we have about 1 matching vector per memory unit. However, by clustering the data, this is improved especially for higher-ranked units. This shows that the expected number of matches per positive unit is bigger due to grouping similar vectors together.

memory vectors and dataset vectors are binarized, whereas in the symmetric case query vectors are also binarized.

Figures 11 and 12 show the performance when using compact binary codes. For the symmetric case, the \textit{sum} method seems to perform better than \textit{pinv} on the Holidays and Oxford5k datasets. In the asymmetric case, both methods perform similarly. In all cases, we achieve convergence to the baseline with a complexity ratio well below 1. Implementation efficiency is further improved in the symmetric case by using the Hamming distance calculation instead of dot product.

Large scale experiments. We conduct large scale experiments on Holidays+Flickr1M, this time using image similarity as a performance measure. When using random assignment of images to groups, the performance is close to the baseline while performing roughly three times fewer vector operations than exhaustive search; see Fig. 13.

\textbf{Batch assignment.} We further improve the performance by performing memory unit assignments using \textit{batch spherical k-means}. This is the same as the regular spherical k-means approach discussed in previous sections, except we do not cluster the whole dataset at once, as this would not be compatible with large collections. Instead, we randomly divide the dataset into batches of the same size and run the clustering algorithm separately for each batch. Fig 14 shows that this strategy improves the
Fig. 8. The performance of memory vectors after PCA dimensionality reduction with $d' = 1024$.

Fig. 9. Image retrieval performance with deep learning features ($d = 4096$), as trained by Babenko et al. [3]. Features for Holidays and Oxford5k are retrained on the Landmarks dataset, whereas the ones for UKB are trained on ILSVRC.

Fig. 10. Comparison of sum, pinv and pinvreg constructions in SIFT10k dataset. pinvreg is ridge regression with regularization factor $\lambda = 0.1$. The performance is measured by recall@10.

Fig. 11. Experiments with binary codes after PCA reduction to $d' = 1024$. The quantization is symmetric: real query vectors are binarized and then compared to binary memory vectors.

Fig. 12. Experiments with binary codes after PCA reduction to $d' = 1024$. The quantization is asymmetric: real query vectors are compared to binary memory vectors.

Fig. 13. Image retrieval experiments for Holidays+Flickr1M. Random assignment was used with $d = 1,920$ features for this experiment.
performance while keeping the batch size compatible and reducing the complexity of the clustering algorithm.

4.4 Complexity Gain

Comparison with FLANN [18]. FLANN algorithm on the Holidays+Flickr1M dataset reveals that the convergence to the baseline is achieved with a speedup of 1.25, which translates to a complexity ratio of around 0.8. We can achieve similar performance with a complexity ratio of only 0.3. This confirms that FLANN is not effective for high-dimensional vectors. In this experiment, we chose the autotuned mode of FLANN library, and set target_precision = 0.95, build_weight = 0.01, and memory_weight = 0.

Execution time. We have shown that we get close to baseline performance while executing significantly fewer operations. We now measure the difference in execution time under a simple setup: \( d = 1024 \) and \( N = 1M \) dataset vectors. An average dot product calculation between the query and dataset is 0.2728s. With \( N/10 \) memory vectors and \( nM_p \approx 100k \) vectors in positive memory vectors, the execution time decreases to 0.0544s. We improve the efficiency even further by using memory vectors with symmetric compact codes and Hamming Distance computation: the execution time becomes 0.0026s. Our method can be parallelized for even more improvement.

5 Conclusion

We have presented and analyzed two strategies for designing memory vectors, enabling efficient membership tests for real-valued vectors. We have also showed two possible assignment strategies and analyzed their performance theoretically and experimentally. For random assignment, a simple strategy based on sum aggregation achieves remarkably good results for high-dimensional vectors. Yet it is possible to go beyond this simple strategy by optimizing the hypothesis test under a few assumptions. The resulting approach turns out to be a weighted summation of the stored vectors that can be implemented in an online manner. On the other hand, when the vectors to be stored in the same memory unit are assumed to have some correlation with each other, sum is on par with the optimized pmv. Our analysis therefore confirms the interest of existing methods based on clustering. Our theoretical analysis is validated by experiments done on standard benchmarks for image retrieval, with datasets comprising up to 1 million images. Our conclusion is the following. When indexing an image collection represented by high-dimensional vectors, there are two strategies to create a search architecture based on memory vectors. If the images are numerous and pre-clustering is not an option, one can use the pmv construction to represent memory units with random assignment. Alternatively, for more scalable problems, it is possible to use a clustering algorithm to group similar vectors in the same memory unit. This strategy improves overall performance for low complexity ratio, but introduces additional offline training time. Under this strategy, sum and pmv perform similarly, and sum is more preferable due to its simpler calculation.

References

[1] R. Arandjelović and A. Zisserman. Extremely low bit-rate nearest neighbor search using a set compression tree. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2014.
[2] A. Babenko and V. Lempitsky. The inverted multi-index. In CVPR, June 2012.
[3] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky. Neural codes for image retrieval. In ECCV, 2014.
[4] C. M. Bishop et al. Pattern recognition and machine learning, volume 1. Springer New York, 2006.
[5] L. Bo and C. Sminchisescu. Efficient match kernels between sets of features for visual recognition. In NIPS, 2009.
[6] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In STOC, pages 380–388, May 2002.
[7] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In ECCV Workshop Statistical Learning in Computer Vision, 2004.
[8] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. Machine learning, 42(1-2):143–175, 2001.
[9] A. R. DiDonato and M. P. Jarnagin. A method for computing the incomplete beta function ratio. Technical Report No. 1949, Computation and Analysis laboratory, U.S. naval weapons Laboratory, Dahlgren, VI, USA, Oct. 1966.
[10] W. Dong, M. Charikar, and K. Li. Asymmetric distance estimation with sketches for similarity search in high-dimensional spaces. In SIGIR, pages 123–130, July 2008.
[11] A. Gordo and F. Perronnin. Asymmetric distances for binary embeddings. In CVPR, 2011.
[12] T. Groves and T. Rothenberg. A note on the expected value of an inverse matrix. Biometrika, 56(3):690–691, Dec. 1969.
[13] M. Jain, H. Jégou, and P. Gros. Asymmetric hamming embedding. In ACM Multimedia, October 2011.
[14] H. Jégou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In ECCV, October 2008.
[15] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. IEEE Trans. PAMI, 33(1):117–128, January 2011.
[16] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In CVPR, June 2010.
[17] H. Jégou and A. Zisserman. Triangulation embedding and democratic kernels for image search. In CVPR, June 2014.
[18] M. Muja and D. G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. IEEE Trans. PAMI, 36, 2014.
[19] N. Murray and F. Perronnin. Generalized max-pooling. In CVPR, June 2014.
[20] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In CVPR, pages 2161–2168, June 2006.
[21] F. Perronnin and C. R. Dance. Fisher kernels on visual vocabularies for image categorization. In CVPR, June 2007.
[22] J. Philbin, O. Cham, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In CVPR, June 2007.
[23] C. R. Rao and S. K. Mitra. Generalized inverse of matrices and its applications, volume 7. 1971.
[24] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In ICCV, pages 1470–1477, October 2003.
[25] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In Proceedings of the International Conference on Very Large Databases, pages 194–205, 1998.
[26] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In NIPS, December 2009.
**APPENDIX A**

**DISTRIBUTION OF A SCALAR PRODUCT**

Let $Y$ be a random vector uniformly distributed on the unit hypersphere in $\mathbb{R}^d$ ($\|Y\| = 1$), and $m$ a fixed vector. This section studies the distribution of $S = Y^\top m$. To generate $Y$, we can first generate a multivariate Gaussian vector $G = (G_1, \ldots, G_d)^\top \sim N(0, I_d)$, where $I_d$ is the $d \times d$ identity matrix, and set $Y = G/\|G\|$. This means that $G_i$ are i.i.d. Gaussian distributed: $G_i \sim N(0, 1)$. Without loss of generality, using symmetry of the Euclidean norm, assume that $m = (\|m\|, 0, \ldots, 0)$. This simplifies into

$$S = Y^\top m = \|m\| \frac{G_1}{\sqrt{\sum_{i=1}^d G_i^2}}.$$  

(20)

Obviously, $-\|m\| \leq S \leq \|m\|$ so that its cdf $F_S(s)$ equals 0 if $s \leq -\|m\|$, and 1 if $s \geq \|m\|$. For all $s \in [0, \|m\|]$, $P(S \geq s)$ is the probability that vector $G$ belongs to the convex cone pointing in the direction $m$ and with angle $\cos^{-1}(s/\|m\|)$ By a symmetry argument (replacing $m$ by $-m$), $P(S \leq -s) = P(S \geq s)$, giving

$$F_S(-s) = 1 - F_S(s), \quad \forall 0 \leq s \leq \|m\|. \quad (21)$$

It also implies that

$$P(S^2 \geq s^2) = P(S \geq s) + P(S \leq -s) = 2(1 - F_S(s)). \quad (22)$$

Going back to $G$, we can write

$$P(S^2 \geq s^2) = P\left( \frac{G_1^2}{\sum_{i=1}^d G_i^2} \geq \tau^2 \right)$$

$$= P\left( \frac{G_1^2}{\sum_{i=2}^d G_i^2} \geq \frac{1 - \tau^2}{1 - \tau^2} \right), \quad (23)$$

with $\tau = s/\|m\|$. By definition, the random variable $U = (d - 1)\sum_{i=2}^d G_i^2$ has an F-distribution $F(1, d - 1)$. It follows that

$$P(S^2 \geq \tau^2 \|m\|^2) = 1 - I_{\tau^2}(1/2, (d - 1)/2), \quad (24)$$

where $I_{\tau^2}(a, b)$ is the regularized incomplete beta function. In the end, $\forall s \in [0, \|m\|]$, we have

$$F_S(s) = \left(1 + I_{\tau^2}(1/2, (d - 1)/2) \right)/2. \quad (25)$$

By taking the derivative of this expression and accounting for symmetry, the pdf is found to be:

$$f_S(s) = \frac{(1 - s^2/\|m\|^2)^{d-2}}{\|m\|^2 B(1/2, (d - 1)/2)}, \forall s, -\|m\| \leq s \leq \|m\| \quad \text{where } B(a, b) \text{ is the beta function. This implies that } E[S] = 0 \text{ and}$$

$$\forall [S] = \frac{\|m\|^2}{B \left(\frac{1}{2}, \frac{d-1}{2}\right)} \int_0^1 a^2(1 - a^2)^{d-3} da$$

$$= \|m\|^2 \frac{B \left(\frac{1}{2}, \frac{d-1}{2}\right)}{B \left(\frac{1}{2}, \frac{d-3}{2}\right)} = \frac{\|m\|^2}{d}. \quad (26)$$

When $d \to \infty$, using the expansion [9, Eq. (26)] of the regularized incomplete beta function in (25) yields

$$F_S(s) \approx \Phi \left( \frac{\sqrt{d} \|m\|^2 s}{\|m\|^2 + \sqrt{1 - s^2/\|m\|^2}} \right), \quad (26)$$

which is approximately $\Phi(\sqrt{d/\|m\|^2}s)$ for small $s$, i.e., the cdf of a centered Gaussian r.v. with variance $\|m\|^2/d$.

**APPENDIX B**

**EXPECTED VALUE OF THE MEMORY UNIT**

We assume that the vectors $x_i$ are not related to each other so that $X < d, X$ has $n$ linearly independent columns. Then $X^T = (X^T X)^{-1} X^T$ and $\|m\|^2 = 1_n^T (X^T X)^{-1} 1_n$. The $n \times n$ Gram matrix $X^T X$ is real, symmetric, and positive semidefinite, and so it has an eigendecomposition $U \Lambda U^\top$, where $\Lambda$ is a diagonal matrix with non-negative coefficients $\{\lambda_i\}_{i=1}^n$ and $U \Lambda U^\top = U^\top U = 1_n$. Moreover, $tr(X^T X) = \sum_{i=1}^n \lambda_i = n$ because $x_i^\top x_i = 1$, for all $i$.

The first construction $m = X 1_n$ has a square norm $\|m\|^2 = 1_n^T X^T X 1_n = 1_n^T U \Lambda U^\top 1_n$. The second construction $m^* = X (X^T X)^{-1} 1_n$ has a norm $\|m^*\|^2 = 1_n^T (X^T X)^{-1} 1_n = 1_n^T U \Lambda^{-1} U^\top 1_n$. It is difficult to say anything more for a given $X$. However, if we consider $X^T X$ as a random matrix, then $E[(X^T X)^{-1}] - (E[X^T X])^{-1} = E[(X^T X)^{-1}] - I_n$ is positive semidefinite [12]. This shows that $E[\|m^*\|^2] \geq E[\|m\|^2] = n$. This means that the second construction increases the variance of the score under $H_0$.

To make further progress, we resort to the asymptotic theory of random matrices, and especially to the Marcenko-Pastur distribution. Suppose that both $d$ and $n$ tend to infinity while remaining proportional: $n = cd$ with $c < 1$. The matrix $X^T X$ can be thought of as an empirical covariance matrix of the $d$ vectors which are the rows of $X$. These vectors are i.i.d. with bounded support. Therefore, the marginal distribution of the $n$ eigenvalues $\{\lambda_i\}_{i=1}^n$ of $X^T X$ asymptotically follows the Marcenko-Pastur distribution: for all $\lambda$ such that $(1 - \sqrt{c})^2 \leq \lambda \leq (1 + \sqrt{c})^2$,

$$f_{MP}(\lambda) = \frac{\sqrt{\lambda(1 - \sqrt{c})^2(1 + \sqrt{c})^2 - \lambda}}{2\pi \sqrt{c}}. \quad (27)$$

Moreover, for any function $\psi$ bounded over the interval $[(1 - \sqrt{c})^2, (1 + \sqrt{c})^2]$:

$$\frac{1}{n} \sum_{i=1}^n \psi(\lambda_i) - \int \psi(\lambda) f_{MP}(\lambda) d\lambda \to 0. \quad (28)$$

Using the eigendecomposition, $X^T X = U \Lambda U^\top$, we get $X (X^T X)^{-1} = U \Lambda^{-1} U^\top$, where the columns of $U$ are the random eigenvectors. Then, asymptotically as $d \to \infty$:

$$n^{-1} E[1_n^T (X^T X)^{-1} 1_n] - \int \lambda^{-1} f_{MP}(\lambda) d\lambda \to 0. \quad (29)$$

This shows that $E[\|m^*\|^2]/n$ converges to $\frac{1}{n}$ asymptotically (the above integral is the Stieljes transform of the Marcenko-Pastur distribution evaluated at $z = 0$), whereas $E[\|m\|^2]/n$ converges to $E[\lambda] = 1$. In expectation, $M^*$ has a higher variance, but only by a factor of $1/(1 - c)$ which remains acceptable if $c = n/d$ is small.

**APPENDIX C**

**Y UNIFORMLY DRAWN OVER A SPHERICAL CAP**

Assume that $Y$ is uniformly distributed over the spherical cap $C_{u, \gamma}$, which is the intersection of the unit hypersphere and the single hypercone of axis $u$, $\|u\| = 1$ and angle $\gamma$. In other words, $\|Y\| = 1$ and $S' = Y^\top u > \cos(\gamma)$. Denote $\eta = \cos(\gamma)$ and $\bar{\eta} = \sin(\eta)$. The probability distribution function of $S'$ is

$$f_{S'}(s') = \frac{f_{S}(s') \mathbb{1}_{[\bar{\eta} > \eta]}(s')}{1 - F_{S}(s')}, \quad (30)$$
This stems into
\[
E[S'] = \frac{2(1 - \eta^2)^{\frac{d}{4}}}{(d - 1)B(\frac{1}{2}, \frac{d-1}{2})} \left(1 - \tilde{\eta}I_{d^2} \left(\frac{1}{2}, \frac{d-1}{2}\right)\right),
\] (31)

Note that:
- \( \eta = -1: E[S'] = 0 \). The cap is the full hypersphere.
- \( \eta \to 1: E[S'] \to 1 \), thanks to De l’Hospital’s rule. The cap reduces to \( \{u\} \).

In the same way:
\[
E[S'^2] = \frac{1 + \frac{d}{2} \tilde{\eta}I_{d^2} \left(\frac{3}{2}, \frac{d-1}{2}\right)}{d - \frac{3}{2} \tilde{\eta}I_{d^2} \left(\frac{3}{2}, \frac{d-1}{2}\right)},
\] (32)

from which we can deduce \( \mathbb{V}(S') \) with the König-Huygens formula. Note that:
- \( \eta = -1: \mathbb{V}[S'] = 1/d \).
- \( \eta \to 1: \mathbb{V}[S'] \to 0 \), thanks to De l’Hospital’s rule.

From now on, we define
\[
\mu_n(\eta, d) = E[(S')^n],
\] (33)
with \( S' = Y^\top u \) and \( Y \sim \mathcal{U}_{\text{cor, } \gamma} \).

**APPENDIX D**

**MODELING K-MEANS FOR THE SUM CONSTRUCTION**

We assume that k-means has packed together in a memory unit independent vectors uniformly distributed over the spherical cap: \( x_i \sim \mathcal{U}_{\text{cor, } \gamma} \). Vector \( x_i \) can be modeled as \( x_i = S'_i u + N_i \), where \( \{S'_i\}_{i=1}^n \) are i.i.d. according to the pdf described in Appendix C, \( N_i^\top u = 0 \) and \( ||N_i||^2 = 1 - (S'_i)^2 \). Their memory vector is the sum \( m = \sum_{i=1}^n x_i \).

**D.1 Hypothesis \( \mathcal{H}_0 \)**

\( Y \) is independent from \( m \). It follows that:
\[
E[Y^\top m] = 0, \quad \mathbb{V}[Y^\top m] = \mathbb{V}[\|m\|^2]/d
\] (34)

with
\[
\mathbb{E}[\|m\|^2] = \mathbb{E}\left[\sum_{i=1}^n \|x_i\|^2 + 2 \sum_{i,j} S'_i S'_j + N_i^\top N_j\right] = n + n(n-1)\mu_1(\eta, d)^2,
\] (35)

where \( \mu_1(\eta, d) \) is given in (31).

To sum up: while \( E[Y^\top m] = 0 \), the variance is increasing with \( \eta \).
- \( \eta = -1: \) It equals \( n/d \) as already shown by (4).
- \( \eta \to 1: \) It converges to \( n^2/d \).

**D.2 Hypothesis \( \mathcal{H}_1 \)**

We now single out the role of the matching vector \( X_1: m = x_1 + m' \). The query is modeled as \( Y = \alpha x_1 + \beta Z \) with \( \|Z\| = 1 \) and \( Z^\top x_1 = 0 \). With the notation as above:
\[
Y^\top m = \alpha + \alpha \sum_{i=2}^n S_i S_i + \alpha N_i^\top m' + \beta Z^\top m'.
\] (37)

The expectation easily comes as
\[
E[Y^\top m] = \alpha (1 + (n - 1)\mu_1(\eta, d)^2).
\] (38)

The variance of the second summand is given by the law of total variance:
\[
\mathbb{V}[\alpha \sum_{i=2}^n S_i S_i] = \alpha^2 \mu_2(\eta, d)^2 - \mu_1(\eta, d)^4.
\] (39)

The variance of the third term is
\[
\mathbb{V}[\alpha \sum_{i=2}^n N_i^\top N_i] = \alpha^2 \mu_1(\eta, d)^2.
\] (40)

The variance of the last summand is more complex to analyze. We need to decompose \( Z \) into its projection on \( u \) and on the complementary space.
\[
\mathbb{V}[\beta (Z^\top u) m'] = (1 - \alpha^2)^{n-1} \mathbb{V}[\beta (Z^\top u) m'] = (1 - \alpha^2)^{n-1} (1 - \mu_2(\eta, d))
\]
\[
\times (1 + (n - 2)\mu_1(\eta, d)^2)
\]

To sum up: \( E[Y^\top m] \) increases while \( \mathbb{V}[Y^\top m] \) decreases with \( \eta \).
- \( \eta = -1: \) \( E[Y^\top m] = \alpha \) while \( \mathbb{V}[Y^\top m] = (n - 1)/d \) as already shown by (5).
- \( \eta \to 1: \) \( E[Y^\top m] \to \alpha n \) whereas \( \mathbb{V}[Y^\top m] \to 0 \).

In the end, under the Gaussian assumption, the Kullback-Leibler distance between both distributions increases which proves that identifying the positive memory units becomes easier as \( \eta \) increases (see Fig. 15).

**APPENDIX E**

**MODELING K-MEANS FOR THE PINV CONSTRUCTION**

**E.1 Hypothesis \( \mathcal{H}_0 \)**

We make the same assumption as in Appendix D. We write \( X = uS^\top + N \) with \( S^\top \) a \( n \times 1 \) vector storing the correlations \( u^\top x_i, 1 \leq i \leq n \) and \( N \) a \( d \times n \) matrix whose columns are
random vectors orthogonal to $u$ and of norm $\sqrt{1 - S_i^2}$. The memory unit is now given by (9): $m^* = X(X^T X)^{-1}1_n$. Eq. (34) holds but with a new expression for the norm of $m^*$:

$$
E[||m^*||^2] = E[1_n^T (X^T X)^{-1} 1_n] \\
\geq 1_n^T (E[X^T X])^{-1} 1_n.
$$

(41)  \quad (42)

We write matrix $X^TX = X^T X = SS^T + NN^T$ s.t.: 

$$
E[X^T X] = (1 - E[S^T]1_n + E[S^T]1_n^T 1_n),
$$

(43) whose inverse is given by the Sherman-Morrison formula:

$$
E[X^T X]^{-1} = 
\frac{1}{1 - E[S^T]^2} \left( 1_n + \frac{E[S^T]^2}{1 + (n - 1)E[S^T]^2} 1_n^T 1_n \right),
$$

leading to

$$
E[||m^*||^2] \geq \frac{n}{1 + (n - 1)E[S^T]^2}.
$$

(44)

To sum up: While $E[Y^T m^*]$ remains constant, the lower bound of the variance is decreasing with $\eta$.

- $\eta = -1$: The lower bound equals $\frac{1}{d}$ which is tight w.r.t. to the previous result in Sect. B (i.e. $\frac{n}{d} \approx (1 + \frac{n}{d})$ for $n$ much smaller than $d$.
- $\eta \to 1$: The lower bounds converges to $1/d$ and is also tight since $m^* \to x_1$ as the vectors are converging to $x_1$.

### E.2 Hypothesis $H_1$

We single out the matching vector $x_1$ by writing $X = (x_1, \bar{X})$ with $\bar{X} = uS^T + N$. $S^T$ being now a $(n - 1)\times 1$ vector storing the correlations $u_x$, $2 \leq i \leq n$ and $N$ a $d \times (n - 1)$ matrix whose columns are random vectors orthogonal to $u$ and of norm $\sqrt{1 - S_i^2}$. This makes

$$
X^T X = \left( \begin{array}{c}
1 \\
X^T x_1 \\
\bar{X}^T \bar{X}
\end{array} \right),
$$

(45)

whose inverse is

$$
(X^T X)^{-1} = \left( \begin{array}{ccc}
1 + x_1^T XD & x_1^T x_1 & -x_1^T XD \\
X^T x_1 & -DX^T x_1 & D
\end{array} \right). \quad (46)
$$

with $D = (\bar{X}^T (I - X_1 X_1^T) \bar{X})^{-1}$. This makes the following memory vector:

$$
\begin{align*}
\bar{m}^* &= x_1 + m_1^* \\
\bar{m}_1^* &= (I - x_1 x_1^T) XD (1_{n-1} - X x_1)
\end{align*}
$$

(47)  \quad (48)

Note that $||\bar{m}^*||^2 = 1 + ||\bar{m}_1^*||^2$ because $x_1 \bar{m}_1^* = 0$.

The query vector being defined as in Sect. D.2, its correlation with the memory vector is

$$
Y^T m^* = \alpha + \sqrt{1 - \alpha^2} Z^T \bar{m}^*,
$$

(49)

whose expectation and variance are given by

$$
E[Y^T m^*] = \alpha, \quad V[Y^T m^*] = (1 - \alpha^2) E[||\bar{m}^*||^2] \frac{d}{d - 1}
$$

(50) with

$$
E[||\bar{m}^*||^2] = \frac{E[||m^*||^2] - 1}{1 + (n - 1)E[S^T]^2}.
$$

(51)  \quad (52)

To sum up: While $E[Y^T m^*]$ remains constant, the lower bound of the variance is decreasing with $\eta$.

Fig. 16. Expectations and Variances under both hypothesis and Kullback-Leibler distance as functions of $\eta$ for the pinv construction. $\alpha = 0.5$, $d = 512$.

- $\eta = -1$: The lower bound equals $(1 - \alpha^2) \frac{n-1}{\alpha^2-1}$ which is tight w.r.t. to the previous result in Sect. B (i.e. $(1 - \alpha^2) \frac{n}{\alpha^2-1}$ for $n$ much smaller than $d$.
- $\eta \to 1$: The lower bounds converges to 0 and is also tight since $m^* \to x_1$ as the vectors are converging to $x_1$.

In the end, under the Gaussian assumption, the Kullback-Leibler distance between both distributions increases which proves that identifying the positive memory units becomes easier as $\eta$ increases (see Fig. 16).