ROBUST SPARSE HASHING

Anoop Cherian Vassilios Morellas Nikolaos Papanikolopoulos

Dept. of Computer Science and Engineering
University of Minnesota, Minneapolis, MN-55455

ABSTRACT

We study Nearest Neighbors (NN) retrieval by introducing a new approach: *Robust Sparse Hashing* (RSH). Our approach is inspired by the success of dictionary learning for sparse coding; the key innovation is to use learned sparse codes as hashcodes for speeding up NN. But sparse coding suffers from a major drawback: when data are noisy or uncertain, for a query point, an exact match of the hashcode seldom happens, breaking the NN retrieval. We tackle this difficulty via our novel dictionary learning and sparse coding framework called RSH by learning dictionaries on the robustified counterparts of uncertain data points. The algorithm is applied to NN retrieval for Scale Invariant Feature Transform (SIFT) descriptors. The results demonstrate that RSH is noise tolerant, and at the same time shows promising NN performance over the state-of-the-art.

Index Terms— Sparse coding, Robust optimization, Nearest neighbors

1. INTRODUCTION

A key task in many applications of computer vision and image processing is to retrieve data points from a large dataset that are similar to a query input. Generally, real-world data are high dimensional (SIFT, GIST, etc.) and thus retrieving Nearest Neighbors (NN) is computationally expensive [1]. Many approaches have been advocated to improve the speed and accuracy of the NN problem in large dimensions. One effective method being Locality Sensitive Hashing (LSH), which takes advantage of some specific properties of the data to generate compact hashcodes. The underlying idea being similar data will generate the same code leading to faster lookup using a hash table. In this paper, we deal with such a hashing algorithm based on Dictionary Learning (DL), which facilitates learning a hashing dictionary from the data space itself under sparsity constraints.

Traditionally, for each input vector, DL algorithms select a set of dictionary atoms that minimize the reconstruction error. This approach can cause two vectors close to each other in the input space, to have different dictionary atoms be involved in the reconstruction, making these vectors incomparable in the dictionary space. This mismatch can arise due to: (a) the vectors being noise corrupted versions of each other, whereby some of the dictionary atoms might reconstruct the noise; or (b) the true vectors themselves being perturbed or uncertain. Existing DL methods follow the first approach and *impose* a noise model on the data. We study the second scenario instead, and explicitly model data uncertainty from a robust optimization perspective.

Before we describe the details of our approach, we would like to enlist the main contributions of this paper. First, we extend the traditional dictionary learning setup towards learning robust sparse codes. Next, we propose an efficient algorithm to solve the resultant optimization problem. Finally, we illustrate the utility of RSH by applying it to the task of NN retrieval for SIFT descriptors.

To put our contributions into perspective, let us review the related work. Due to lack of space, we restrict ourselves to a few works that are directly related to ours. The DL problem has been viewed from certain robust perspectives previously. For example, in [2], dictionary learning in the presence of Gaussian noise is suggested. In [3], a robust formulation of the dictionary learning problem is suggested in the context of outlier detection by using the Lorentian norm instead of the L1 norm. But their method leads to a complex nonconvex optimization problem. Another method using LSH via dictionary learning method is [4], where the noise is modeled as a convolution of Gaussian and Laplacian distributions, followed by a denoising approach to generate robust hashcodes. The density of this convolution is difficult to characterize analytically leading to approximate inference schemes. Against these approaches, in this paper we propose a novel sparse coding algorithm based on robust optimization principles that does not make any distributional assumptions on the data, leading to tractable and efficient solutions.

2. ROBUST SPARSE HASHING

Before describing our formalism, we recall some standard background on DL. Learning an overcomplete basis dictionary in which the input data vectors have sparse representations is roughly the goal of DL.

A commonly used and effective formulation of DL is the following. Let $x_1, \dots, x_m \in \mathbb{R}^d$ be input (training) vectors.

Email ids:{cherian, morellas, npapas}@cs.umn.edu

We compute a dictionary $D \in \mathbb{R}^{d\times n}$ that is overcomplete (i.e., $n \gg d$), so that each $x_i \approx Dc_i$ for some *sparse* vector c_i. A natural optimization problem for learning D and c_i is then

$$\min_{D,c_1,\ldots,c_m} \sum\nolimits_{i=1}^{m} \left(\tfrac{1}{2}\|x_i - Dc_i\|^2 + \beta\|c_i\|_1\right) \text{ s. t. } \|d_j\| \leq 1, \tag{1}$$

for $j = 1, 2, \cdots, d$, where $\beta > 0$ is a sparsity tuning parameter while the constraint $\|d_j\| \leq 1$ normalizes each column d_j of D to prevent degeneracy. Problem (1) is natural but unfortunately also difficult due to its nonconvexity, so we can at best hope for locally optimal solutions. Once D is learned, a LASSO formulation as follows can used to sparse code a given data point x' to obtain the sparse code c':

$$c' = \operatorname{argmin}_c \quad \tfrac{1}{2}\|x' - Dc\|^2 + \beta\|c\|_1. \tag{2}$$

The sparse code generated for a given data point x' depends on the unique correlations that x' produces with the elements in D, along with the sparsity constraints. Thus, each dissimilar data point can be expected to produce a unique active set; a suitable encoding of this active set then used as a hashcode.

Mathematically, suppose $\mathcal{X} = \{x_1, \ldots, x_m\}$ is the set of input vectors for which we have learned a dictionary D and sparse vectors $c_1, \ldots, c_m$ by solving (1). For an arbitrary $x \in \mathcal{X}$, let $J(x) := \{j_1, \ldots, j_k\}$ be the corresponding set of indices such that $c_{j_l} \neq 0$ for $1 \leq l \leq k$ and $x \approx Dc$. The set $J(x)$ may be viewed as a hash-code for x. Indeed, we can build a data structure that stores D along with the vectors $\mathcal{X}$ hashed according to $\{J(x_1), \ldots, J(x_m)\}$. Though such an NN scheme seems attractive, it could be brittle in the presence of noise. Neither while building D, nor while testing a query point, do we really enforce any constraints that $J(x)$ should be similar to $J(\bar{x})$ whenever x and $\bar{x}$ are similar (or strongly correlated) vectors in the original input space. Traditional dictionary learning looks at this problem from a statistical perspective, by assuming that the signal noise belongs to a well-known distribution (like Gaussian [2], or Gaussian and sparse Laplacian [4]) leading to sophisticated optimization problems. In symbols, suppose $x \approx Dc$ and $\bar{x} = (1+\gamma)x$, for some small perturbation $\gamma \neq 0$. For facilitating NN retrieval, we desire $J(x) = J(\bar{x})$; but since D is overcomplete there is *no guarantee* that $c = (1+\gamma)\bar{c}$.

2.1. RSH Formulation

Let the input (training set) be a set of nominal vectors $\bar{\mathcal{X}} = \{\bar{x}_1, \ldots, \bar{x}_m\}$. Let $\mathcal{U}(\bar{x})$ model *uncertainty* for point $\bar{x}$, i.e., it is some set that models perturbations in $\bar{x}$. To be robust against the perturbations, RSH seeks a dictionary so that all points $x \in \mathcal{U}(\bar{x})$ have the same hash codes: $J(x) = J(\bar{x})$. A common approach [5] is to seek an optimal solution that is immune to the worst-case uncertainty. Thus, instead of (1) we consider its *robust* counterpart:

$$\begin{aligned} \min_{D,c_1,\ldots,c_m} \quad & \sum\nolimits_{i=1}^{m} \left(\tfrac{1}{2}\|x_i - Dc_i\|^2 + \beta\|c_i\|_1\right) \\ \text{s.t.} \quad & \|d_j\| \leq 1, \text{ for } j = 1, 2, \ldots, d \\ & x_i \in \mathcal{U}(\bar{x}_i), \text{ for } i = 1, 2, \ldots, m. \end{aligned} \tag{3}$$

Problem (1) is already difficult enough, and (3) is even harder because of the robustness constraints. But for a suitable choice of $\mathcal{U}$ we can solve (3) without making it much harder than the ordinary DL problem (1). Going by the robust principles [5], the uncertainty set can be written as:

$$\mathcal{U}(\bar{x}) := \{x : x = \bar{x} + Pu, \ \|u\|_p \leq 1\},$$

where P is an invertible matrix, and p defines the shape of the uncertainty set. Balancing between the tractability of the formulation and performance on our datasets, we decided to use *elliposidal uncertainty* for our data perturbations; this corresponds to P being a positive definite matrix and $p = 2$. This leads to the following two stage min-max type version of (3):

$$\begin{aligned} & \min_{D,C} \sum\nolimits_{i=1}^{m} \left(\tfrac{1}{2}\|\bar{x}_i + Pu_i^* - Dc_i\|^2 + \beta\|c_i\|_1\right) \\ \text{where} \quad & u_i^* := \max_{\|u\|\leq 1} \tfrac{1}{2}\|\bar{x}_i + Pu\|^2 \\ \text{and} \quad & \|d_j\| \leq 1, \text{ for } j = 1, 2, \ldots, d, \end{aligned} \tag{4}$$

where $C = [c_1, \ldots, c_m]$. The formulation (4) is the core RSH problem of this paper; so let us move onto potential algorithms for solving it.

2.2. Algorithms for RSH

The basic difference between the traditional dictionary learning problem (1) and the robust formulation (4) is in computing u^* which is the direction of worst case perturbation. Since u is dependent only on the nominal data points $\bar{x}$, we can solve for it independently of D and C. Once u is computed for each $\bar{x}$, (4) boils down to (1) where the data points are now the robustified vectors $x' = x + Pu^*$. To this end, the primary optimization problem that we need to tackle is in efficiently solving for u; avoiding the subscripts for simplicity, the formulation in (4) on u^* can be rewritten as:

$$f(u) := \min_{\|u\|\leq 1} \quad -\tfrac{1}{2}u^T P^T P u - u^T P x - \tfrac{1}{2}x^T x, \tag{5}$$

At first sight (5) might look bad, because the quadratic term is $-P^T P$, which is negative-definite instead of positive-definite. That is, (5) is a nonconvex quadratic program. Fortunately, because there is just one constraint, (5) still remains tractable. In fact, as is well-known, for this particular type of nonconvex problems, strong-duality holds thanks to the so-called S-lemma [see e.g.[6]]. Problem (5) is even more special: it is a trust-region subproblem, although a nontrivial one because of the negative-definite matrix $-P^T P$.

Though, we can use a generic trust-region solver like the LSTRS method [7] for this problem, we show below that this problem can be solved more efficiently.

2.3. Efficient Implementation via Newton Descent

Using a multiplier λ, the Lagrangian of (5) can be written as:

$$L(\lambda, u) := -\tfrac{1}{2}u^T P^T P u - u^T P x - \tfrac{1}{2}x^T x + \lambda\left(u^T u - 1\right)$$

Setting $\frac{\partial L}{\partial u} = 0$, we have: $u = \left(P^T P - 2\lambda\mathcal{I}\right)^{-1} P x$, where $\mathcal{I}$ is the $d \times d$ identity matrix. Let $P^T P = U \Sigma U^T$ for a unitary U and diagonal Σ, and substituting for u in the constraint, we have the following root finding problem in the scalar λ: $x^T U^T \Sigma\left(\Sigma - 2\lambda\mathcal{I}\right)^{-2} \Sigma U x = 1$. Using the substitution $q = \Sigma U x$ and using the Hadamard product $\hat{q} = (q \circ q)$ then leads to a simple vector equation:

$$\sum_{i=1}^{d} \frac{\hat{q}_i}{(\sigma_i - 2\lambda)^2} = 1. \tag{6}$$

where $\hat{q}_i$ is the ith component of $\hat{q}$ and σ_i is the ith eigenvalue of Σ. As is clear, (6) is convex in λ and can be efficiently solved by the *Newton-Raphson* method for a suitable initialization (assuming $P^T P$ is reasonably well-conditioned). On an average, 3–6 times speedup was observed against the LSTRS method.

2.4. Computing the Uncertainty Matrix

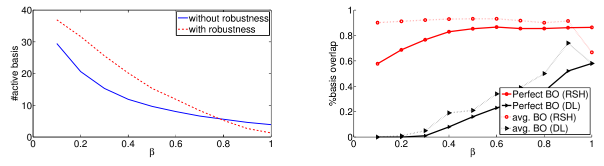
The parameter P of our formulation plays a crucial role in quantifying uncertainty and thus care must be taken in learning this parameter. In this paper, we propose a supervised approach by setting P to the *Lowner-Jones* uncertainty ellipsoid computed on matching data pairs. That is: suppose we have a training set of matching data pairs $\{(x_i, x'_i), i = 1, 2, \cdots, N\}$, where we assume $x_i = \bar{x}$ is the nominal data point and $x'_i = \bar{x} + \delta_i$ for some noise δ_i. We compute the difference $\delta_i = x_i - x'_i, i = 1, 2, \cdots, N$. An uncertainty elliposid can be defined as an ellipsoid of *minimum volume* enclosing all the δ_i's. If P represents the parameters of this ellipsoid and if c is its centroid, then

$$\min_{P \geq 0, c} -\log det(P) \text{ s. t. } (\delta_i - c)^T P(\delta_i - c) \leq 1 \tag{7}$$

for $i = 1, \cdots, N$. The problem (7) can be solved via the well-known *Khachiyan* algorithm. Since we work with zero-mean data, c is generally seen to be very close to the origin and thus can be neglected.

3. EXPERIMENTS

Before describing our experiments, let us characterize our performance metrics. Let $J(x_1)$ and $J(x_2)$ represent the robust sparse codes for two data points x_1 and x_2 respectively. If x_1 and x_2 are true nearest neighbors, then for the success of RSH, we would expect $J(x_1) = J(x_2)$. Towards this end, we define *Basis Overlap* (BO) as: $BO(x_1, x_2) := |J(x_1) \cap J(x_2)|/|J(x_1) \cup J(x_2)|$, where $|\,|$ stands for the set cardinality. When $BO = 1$, we call it *Perfect Basis Overlap*. Next, let us define the performance metric on a dataset. Due to the unavailability of the ground truth for our datasets, the baselines were decided by a Euclidean linear scan. Suppose our query set has q inputs. For each query item i, assume k ground truth neighbors (G_i^k) were found using linear scan, followed by k nearest neighbors (A_i^k) are retrieved using the respective algorithm. Then we define $Accuracy = \frac{1}{q}\sum_i |G_i^k \cap A_i^k|/|G_i^k|$.



(a) (b)

Fig. 1. Simulation results: (a) Active set size against an increasing regularization β, (b) Average NN performance against increasing regularization.

Simulations: We would like to first highlight the importance of our robust formulation against robustness achieved via increasing the regularization constant in the LASSO formulation. Recall that regularization controls the degree to which the active coefficients corresponding to less correlated bases are shrunk towards zero, a larger regularization means shorter and more robust sparse codes. Since the length of the sparse codes play a significant role in the uniqueness of our hashing formulation, shorter codes are not desired. To show that RSH leads to longer and robust codes, we did the following experiment: we simulated 50K matching data pairs (40D) using a known 40×120 dictionary. These vectors are later corrupted independently using Gaussian and Laplacian noise (as suggested in [4]). A training set of 40K was used to learn a dictionary using the RSH framework. Next, we applied RSH to generate sparse codes for the data pairs in the test set. Figure 1(a) shows the average active basis set size for increasing regularization, while Figure 1(b) plots the average BO between the bases in the noisy pairs, averaged on a randomly chosen 1K test vectors from the 10K test set. As is clear RSH leads to longer hashcodes and at the same time improved basis overlaps against a possible robustness achieved by increasing the regularization.

Real Data Experiments: The primary goal of this section is to show the robustness introduced by RSH on real data under various noise models; the idea being to show that a single robust uncertainty model (as in RSH) provides the needed robustness, that could otherwise have needed different probabilistic denoising models. Towards this end, we chose the SIFT benchmark dataset[1], which consists of eight image cat-

[1]www.robots.ox.ac.uk/ vgg/research/affine/index.html

egories, each category consisting of six image pairs of the same scene, undergoing a deformation of a specific type. But before going into the details of this experiment, let us first provide details of our experimental setup and how we learned the parameters of RSH for the SIFT database.

We used 5M SIFT descriptors from Holidays Dataset for learning the dictionary. The dictionary size was selected empirically, i.e., we trained multiple dictionaries of sizes varying from 256 to 4096 at steps of 256 and measured the NN performance on a small test set of 10K SIFT descriptors, before deciding the size as 2048. We used a regularization value of $\beta = 0.2$. To decide the uncertainty ellipsoid matrix P, we randomly selected a set of 20K points from the dataset and computed their one-nearest neighbor using a linear scan with Euclidean distance, subsequently computing the Lowner-Jones elliposid using (7).

Nearest Neighbor Performance: We chose to compare our method with (i) KDT (Best-Bin-First based on KD-Trees), (ii) Kernelized LSH, (iii) LASSO based NN (DL), and (iv) Product Quantization (PQ) method [8]. The latter technique was recently shown to outperform other well-known methods like spectral hashing and Hamming embedding. As we alluded to previously, each category in our dataset had six images each undergoing a specific transformation: (i) *BARK* (magnification), (ii) *BIKES* (atmospheric blur), (iii) *LEUVEN* (illumination), (iv) *TREES*(Gaussian blur), (v) *BOAT*, (similarity transform) and (vi) *GRAPHIC* (affine), (vii) *UBC* (compression) and (viii) *WALL* (view angle). SIFT descriptors were extracted from each of the images in a category. Image pairs were chosen from each category and NN is applied on the descriptors from these pairs, where the descriptors from the first image is used as the database and the other as the query. We used the following approximate NN strategy to ensure an ϵ-neighborhood. We first computed the NNs for a given query point using linear scan. Suppose d_{ls} denotes the distance of the NN for a query point q to its closest neighbor in the database. If d_q is the distance of the closest neighbor found by an algorithm, then we say the NN is correct if $\frac{d_{ls}}{d_q} > \epsilon$ (we used $\epsilon = 0.9$ for all the algorithms). The results are shown in Fig. 2. As is clear, RSH is seen to outperform all the other methods.

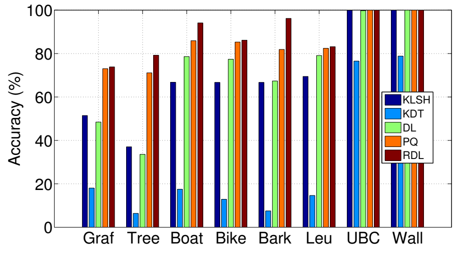


Fig. 2. Performance of RSH for NN on noisy data for various types of distortion models.

Computational Cost: Our algorithms were implemented mainly in MATLAB. On a Pentium 4 machine with 3GHz CPU and 3GB RAM, it took on an average 9ms (in MATLAB) for the finding the robust directions (u) and less than 100μs for sparse coding a SIFT vector using the SPAMS toolbox[2], followed by less than 1ms for querying a hashtable of 1M SIFT descriptors. This performance seems comparable to those reported in [8].

4. CONCLUSION

In this paper, we proposed a novel sparse coding formulation based on robust optimization principles for improved NN performance. We provided a new algorithm for efficiently solving our formulation, followed by simulated and real-world experiments showing the benefits of our method. Going forward, we would like to investigate the adequacy of other uncertainty models such as polyhedral uncertainty towards NN problems.

5. ACKNOWLEDGEMENTS

This research has been supported in part by the National Science Foundation through grants #IIP-0443945, #CNS-0821474, #IIP-0934327, #CNS-1039741, #IIS-1017344, #IIP-1032018, and #SMA-1028076.

6. REFERENCES

[1] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," *Proc. of the Thirtieth Annual ACM Symposium on Theory of Computing*, pp. 604–613, 1998.

[2] M. Elad and M. Aharon, "Image denoising via learned dictionaries and sparse representation," *CVPR*, vol. 1, pp. 895–900, 2006.

[3] I. Ramirez, F. Lecumberry, and G. Sapiro, "Universal priors for sparse modeling," in *CAMSAP*, 2009, pp. 197–200.

[4] A. Cherian, S. Sra, and N. Papanikolopoulos, "Denoising Sparse Noise via Online Dictionary Learning," in *ICASSP*, 2011.

[5] C. Caraminis, H. Xu, and S. Mannor, *Optmization for Machine Learning*, chapter Robust Optimization in Machine Learning, MIT Press, 2011.

[6] S.P. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge Univ Pr, 2004.

[7] M. Rojas, S.A. Santos, and D.C. Sorensen, "LSTRS: Matlab software for large-scale trust-region subproblems and regularization," *ACM Trans. Math. Software*, vol. 34, no. 2, pp. 11, 2008.

[8] H Jegou, M Douze, and C Schmid, "Product quantization for nearest neighbor search," *TPAMI*, vol. 33, no. 1, pp. 117–128, 2011.

[2]http://www.di.ens.fr/willow/SPAMS/