Approximate Nearest Neighbor Searchtowards Removing the Curse of Dimensionality Query-Aware and Locality-Sensitive Hashing

[1]S Ratna Kumari, [2]D Durga Prasad
[1,2]Dept. of CSE, Baba Institute of Technology and Sciences, Visakhapatnam, India

Abstract
We show a simple vector quantizer that consolidates low mutilation with speedyrecreation and apply it to Approximate Nearest Neighbor (ANN) look in high dimensional spaces. Utilizing the extremely same information structure that is utilized to give non-comprehensive hunt, i.e., disturbed records or a multi list, the thought is to locally streamline an individual Product Quantizer (PQ) per cell and utilize it to encode residuals. Local optimization is over turn and space deterioration; strikingly, we apply a parametric arrangement that accept a typical dissemination and is to a great degree quick to prepare. With a sensible space and time overhead that is consistent in the information estimate, we set another state-of-the-art on a few open datasets, including a billion-scale one. The Approximate Nearest Neighbor (ANN) look plays out the quick and effective recovery of information as the span of information develop increments quickly. It investigates the quantization centroids on numerous relative subspace. We propose an iterative way to deal with limit the quantization blunder keeping in mind the end goal to make a novel quantization plot, which beats the state-of-the-art calculations. The computational cost of our strategy is likewise practically identical to that of the contending strategies.

Keywords
Approximate Nearest Neighbor, High Dimensions, Subspace Clustering, Productive Quantizer.

I. Introduction

Approximate Nearest Neighbor (ANN) look in high dimensional spaces isn't just a repeating issue in computer vision, yet in addition experiencing critical advance. An expansive assortment of techniques keep up all information focuses in memory and depend on effective information structures to register just a set number of correct separations, that is in a perfect world settled. At the other outrageous, mapping information focuses to smaller double codes isn't just proficient in space however may accomplish quick thorough pursuit in hamming space. Product Quantization (PQ) is an option reduced encoding strategy that is discrete however not twofold and can be utilized for thorough or non-comprehensive hunt through altered ordering or multi-ordering. As is valid for most hashing techniques, better fitting to the fundamental appropriation is simple in seek execution, and one such approach for PQ is streamlined Product Quantization (OPQ) [9] and its comparable Cartesian k-implies. How are such preparing strategies valuable? Diverse criteria are material, yet the hidden rule is that all bits dispensed to information focuses ought to be utilized sparingly. Since pursuit can be made quick, such techniques ought to be at last As such, k-implies, portrayed is a vector quantization strategy where by determining k centroids, log2 k bits can speak to a subjective information point in R d for any measurement d; however guileless inquiry is O(dk) and low mutilation implies extensive k. By obliging centroids on a pivot adjusted, m-dimensional network, PQ accomplishes k m centroids keeping seek at O(dk); however as delineated, a large number of these centroids stay without information bolster e.g. on the off chance that the circulations on m subspaces are not autonomous. OPQ enables the framework to experience discretionary revolution and reordering of measurements to better adjust to information and adjust their change crosswise over subspaces to coordinate piece designation that is additionally adjusted. A firmly multimodal appropriation may not profit by such arrangement. Our answer in this work is locally upgraded product quantization (LOPQ). Following a very normal hunt choice of [1-2], a coarse quantizer is utilized to record information by upset records, and residuals between information focuses and centroids are PQ-encoded. Be that as it may, inside cell appropriations are to a great extent unimodal; we locally advance an individual product quantizer per cell. Under no presumptions on the circulation, essentially all centroids are upheld by information, adding to a lower bending. LOPQ requires sensible space and time overhead contrasted with PQ, both for disconnected preparing/ordering, and online questions; yet all overhead is consistent in information measure. It is embarrassingly easy to apply and helps execution on a few open datasets. A multi-list is simple for vast scale datasets and consolidating with LOPQ is less inconsequential, however we give a versatile arrangement by the by.

II. Related Work

There are various papers that utilization slope climbing or kNN diagrams for nearest neighbor look, however to the best of our insight, utilizing slope hopping on k-NN charts is another thought. Papadias [2000] expect that each point (e.g., a picture) is indicated as a gathering of parts (e.g., objects). Each point has the type of $X_i = (V_1,...,V_m)$, where each V_j is a protest and can take esteems from a limited set (e.g., an arrangement of squares of various sizes). The goal is to discover the point in the dataset that has the nearest setup to the inquiry Q. Papadias [2000] says X_i and X_j are neighbors on the off chance that one can be changed over to the next by changing the estimation of one of its factors. At that point a few heuristics to perform slope getting on such a diagram are proposed [Papadias, 2000]. Paredes and Chvez [2005] go for limiting the quantity of separation calculations amid the nearest neighbor look. A k-NN diagram is worked from dataset focuses and when questioned with another point, the chart is utilized to assess the separation of all focuses to the inquiry, utilizing the way that the most limited way between two hubs is an upper bound on the separation between them. Utilizing the upper and lower bound appraisals, Paredes and Chvez [2005] take out focuses that are far from the question point and thoroughly look in the rest of the dataset. Lifshits and Zhang [2009] characterize a perceivability diagram and afterward perform nearest neighbor seek by a voracious directing over the chart. This is a comparable way to deal with our technique, with two contrasts. To begin with, Lifshits and Zhang [2009] seek over the perceivability chart, while we look on the k-NN diagram. k-NN charts are prominent information structures that are utilized as a part of exception identification, VLSI configuration, design acknowledgment and numerous different applications [Paredes and Chvez, 2005]. The second contrast is that Lifshits and Zhang [2009] make the accompanying solid presumption about the dataset. Because of the challenges of concocting proficient calculations for the correct form of the issue, there has been broad work on approximate calculations. Under the approximate setting, restoring any indicate whose separation the question is inside a factor of 1 + ▬ of the separation between the inquiry and the genuine nearest neighbor is worthy. A considerable lot of similar systems are utilized by approximate calculations. Techniques in view of tree-based space partitioning (Arya et al., 1998) and local pursuit (Arya and Mount, 1993) have been created; in the same way as other correct calculations, their question times additionally scale exponentially in the surrounding dimensionality. Locality-Sensitive Hashing (LSH) (Indyk and Motwani, 1998; Datar et al., 2004; Andoni and Indyk, 2006) partitions the space into normal cells, whose shapes are verifiably characterized by the decision of the hash work. It accomplishes an inquiry time of O(dn⇢) utilizing O(dn1+⇢) space, where d is the encompassing dimensionality, n is the dataset estimate and ⇢⇡ 1/(1 + ▬)2 for huge n in Euclidean space, however the reliance on characteristic dimensionality isn't made express. Practically speaking, the execution of LSH debases on datasets with substantial varieties in thickness, because of the uneven dissemination of focuses crosswise over cells. Thusly, different information subordinate hashing plans have been proposed (Pauleve et al. ' , 2010; Weiss et al., 2009; Andoni and Razenshteyn, 2015); not at all like information autonomous hashing plans, in any case, they don't enable dynamic updates to the dataset. A related approach (Jegou et al. ' , 2011) disintegrates the space into commonly orthogonal pivot adjusted subspaces and freely partitions every subspace. It has a question time straight in the dataset estimate and no known certification on the likelihood of accuracy under the correct or approximate setting. An alternate approach (Anagnostopoulos et al., 2015) ventures the information to a lower dimensional space that approximately protects approximate nearest neighbor connections and applies other approximate calculations like BBD trees (Arya et al., 1998) to the anticipated information. Its question time is additionally straight in encompassing dimensionality and sublinear in the dataset measure. Dissimilar to LSH, it utilizes space straight in the dataset measure, at the cost of longer question time than LSH. Tragically, its inquiry time is exponential in inherent dimensionality.

III. K- Nearest Neighbor Classification

The KNN method is a very intuitive method that classifies unlabeled examples based on their similarity with examples in the training set [3]. It is a very intuitive method that classifies unlabeled examples based on their similarity with examples in the training set. KNN is a type of instance-based learning, or lazy learning where the function is only approximated locally and all computation is deferred until classification [7]. This algorithm is amongst the simplest of all machine learning algorithms: an object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of its nearest neighbor. The same method can be used for regression, by simply assigning the property value for the object to be the average of the values of its k nearest neighbors [4]. It can be useful to weight the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. A common weighting scheme is to give each neighbor a weight of 1/d, where d is the distance to the neighbor. This scheme is a generalization of linear interpolation. The neighbors are taken from a set of objects for which the correct classification (or, in the case of regression, the value of the property) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required [5]. The KNN algorithm is sensitive to local structure of the data. KNN rules in effect compute the decision boundary in an implicit manner. In cases such as text classification, another metric such as the overlap metric (or Hamming distance can be used [2]. Another popular approach is to scale features by the mutual information of the training data with the training classes [3].

A. Distance Metric

The three kinds of distances Manhattan Distance (Man), Euclidean Distance (Euld), Minkowski Distance (Mkw) are considered for a KNN algorithm. The Manhattan distance between two items is the sum of the differences of their corresponding components [3].

B. Performance

Assessment assessing accuracy of a remote sensing output is one of the most important steps in any classification exercise. Without accuracy assessment, the output or results is of little value. A user of the imagery who is particularly interested in class A, say, might wish to know what proportion of pixels assigned to class A were correctly assigned. Producer's accuracy is a measure of how much of the land in each category was classified correctly. Different people have different interpretations as to what is a good level of agreement.

1. Approximate Near Neighbor

In this section, we describe data structures for the approximate near neighbor problem (i.e., (c,r)-NN) in `ds. They can be used as subroutines in the reduction from the earlier section. Note that by scaling we can assume r = 1. We start by describing two simplifying reductions, followed by two algorithms for the approximate near neighbor problem.

2. Reductions

Coordinate range reduction We start by describing a method for reducing the range of coordinate values of points in P∪{q}, for the case of the `$_1$ norm. It is essentially due to Bern [1], who used it for the purpose of designing approximate nearest neighbor algorithms with approximation factor polynomial in d. This subroutine improves the efficiency of the data structures.

Fix a >1, and suppose there is a data structure for (c,1)-NN in $[0,a]^d$ under the `$_1$ norm that uses space *S(n,d,c,r,f)*, has query time *Q(n,d,c,r,f)* and has failure probability f. Then there exists a data structure for the same problem over ℓ_1^d, with asymptotically the same time and space bounds, and failure probability f +1/a.

Proof. First, we impose an orthogonal cubic grid on ℓ_1^d, where each grid cell has side length a. The grid is translated by a random vector chosen uniformly at random from $[0,a]^d$. Consider any pair of points p,$q \in \ell_1^d$ such that kp−qk$_1$ ≤ 1. The probability that both p and q fall into the same grid cell is at least

$|p_i - q_i|$ 1

$1 - \sum_i$ $_a \geq 1 -$ $_{a.}$

Therefore, we can proceed as follows:

- For each grid cell C such that C∩P 6= 0/, build a separate data structure for the set C∩P.
- In order to answer a query q∈C, query the data structure for C∩P.

In this way we reduced the (c,1)−NN problem for points in ℓ_1^d to the case when the coordinates of points live in a cube of side a. The query time is increased by an additive term O(d), which (as per the assumptions in Section III.B) is subsumed by *O(Q(n,d,c,r, f))*. Since (as per the assumptions in Section III.B) *S(n,d,c,r, f)* is at least linear in dn, it follows that the space bound of the new data structure remains *O(S(n,d,c,r, f))*. The probability of failure is increased by an additive term of 1/a.

Hamming cube Next, we show that the problem can be simplified further by assuming that the points live in a low-dimensional Hamming cube.

Lemma 3.2. Fix δ >0, a >1, and suppose there is a data structure for (c,r)-NN in a dM-dimensional Hamming cube for M = O(ad/δ) that uses space *S(n,d,c,r, f)*, has query time *Q(n,d,c,r, f)* and has failure probability f. Then there exists a data structure for (c(1+δ),1)-NN for ℓ_1^d with the same time and space bounds, and failure probability f +1/a. we can assume all points live in [0,a]d. If we round all coordinates of points and queries to the nearest multiple of δ/d, then the inter-point distance are affected only by an additive factor of δ. Thus, solving a (c,1+δ)-NN on the resulting point-set and queries yields a solution to (c(1+δ),1)-NN over the original space ℓ_1^d.

In order to solve the (c,1+δ)-NN problem, define a scaling factor s = d/δ. Observe that, if we multiply all coordinates of the data points and queries by s, then all coordinates become integers in the range {0,...,M}. In this case, we use a mapping unary : $\{0,...,M\}^d \to \{0,1\}^{dM}$, such that unary(($x_1$,...,$x_d$)) = unary($x_1$),...,unary($x_d$), where unary(x) is a string of x ones followed by a string of M −x zeros. Observe that the `$_1$ distance between any two points is equal to the Hamming distance of their images. Thus, it now suffices to solve an (c,(1+δ)s)-NN on the point-set and queries mapped by unary.

In the following we focus on solving the problem in `$_s$ norms for s ∈{1,2}. The generalization to any s ∈ [1-2] follows from the theorem of, that states that for any d,γ, there exists a mapping f :$\ell_s^d \to \ell_1^{d'}$, d0 = O(dlog(1/γ)/γ2), so that for any p,$q \in \ell_1^d$ we have

kp − $qk_s \le$ kf(p)− f(q)$k_1 \le$ (1+γ)kp−qk_s.

Note that the mapping f is defined for all points in `$d_s$, even if they are not known during the preprocessing. The mapping f is chosen at random from a certain distribution over linear mappings from $\ell_s^d$ to $\ell_{1\ d0}$, which takes time O($d_{d0}$). This allows us to reduce (c(1+γ),r)-NN in `ds to (c,r)-NN in$\ell_1^{d'}$.

3. Locality-sensitive Hashing

In this section, we describe an algorithm based on the concept of Locality-Sensitive Hashing (LSH). The basic idea is to hash the data and query points in a way that the probability of collision is much higher for points that are close to each other than for those which are far apart. Formally, we require the following.

Definition 3.3. A family H = {h :X → U} is (r_1,r_2, p_1, p_2)-sensitive for (X,D) if for any q, p ∈X we have

- if D(p,q) ≤ r_1 then Pr_H[h(q) = h(p)] ≥ p_1,
- if D(p,q) > r_2 then Pr_H[h(q) = h(p)] ≤ p_2.

In order for a locality-sensitive family to be useful, it has to satisfy inequalities $p_1 > p_2$ and $r_1 < r_2$.

We can solve the (c,r)-NN problem using a locality-sensitive family as follows. We choose r_1 = r and r_2 = c·r. Given a family H of such hash functions for these parameters, we amplify the gap between the "high" probability p_1 and "low" probability p_2 by concatenating several functions. In particular, for kspecified later, define a function family G = {g : X → U^k} such that g(p) = (h_{i1}(p),...,h_{ik}(p)), where hit ∈H, for I = {i_1,...,i_k}⊂{1,...,|H|}. For an integer L we choose L functions g_1,...,g_L from G independently and uniformly at random. Let I_j denote the multi-set defining g_j. During preprocessing, we store a pointer to each p∈P in the buckets g_1(p),...,g_L(p). Since the total number of buckets may be large, we retain only the non-empty buckets by resorting to "standard" hashing.

To process a query q, we carry out a brute-force search for the neighbor of q in buckets g_1(q),...,g_L(q). As it is possible that the total number of points stored in those buckets is large, we interrupt the search after finding the first 3L points (including duplicates). Let p_1,...,p_t be the points encountered during this process. If there is any point p_j such that D(p_j,q) ≤ r_2 then we return the point p_j, else we return null.

IV. Proposed System

In this paper, we propose a novel vector quantization method for ANN search which enables faster and more accurate retrieval on publicly available datasets. We define vector quantization as a multiple affine subspace learning problem and explore the quantization centroids on multiple affine subspaces. We propose an iterative approach to minimize the quantization error in order to create a novel quantization scheme. Compared with existing algorithms, the computational cost of our method is too small.

A. Dataset

The dataset contains movie reviews along with their associated binary sentiment polarity labels. It is intended to serve as a benchmark for sentiment classification. The core dataset contains 50,000 reviews split evenly into 25k train and 25k test sets. The overall distribution of labels is balanced (25k positive and 25k negative)also include an additional 50,000 unlabeled documents for unsupervised learning.

B. Pre-processing

Following are steps in preprocessing.

1. Stop word removal
2. Symbol removal
3. POS tagging (Part Of Speech).

- Stanford POS tagging is used for our study.
- This method finds actual parts of speech using the English parser mode.
- The POS Tagging on the input sentence and uses Verb, Adverb and Adjectives only.
- It uses the standard Penn Treebank POS tag sets. For example: The movie was not quite good. After the Removal of stop word Output is [Movie, not, quite, good] after POS Tagging result is [Movie/NN, not/RB, quite/JJ, good/JJ].

C. Feature Extraction

When the input data is too large to be processed then transforming the input data into the set of feature is called feature extraction. If the features extracted properly from the data than it is expected that will perform the needed task. The feature extraction method, extracts the feature (adjective) from the dataset. Then this adjective is used to show the positive and negative polarity in a sentence which is useful for determining the opinion/sentiment of the individuals using unigram model .Unigram model extracts the adjective and separates it. It discards the preceding and successive word occurring with the adjective in the sentences.

D. Training and Classification

Supervised learning is an important technique for solving classification problems. It works in two phases i.e. Training and Testing. The training phase incorporates training number of positive and negative comments using IMDB review dataset. After that assign weights to each comment in the training phase and also apply fuzzy logic to remove the negations like not, never etc. It helps to gain the accuracy in terms of correlations and dependencies. The main purpose of training is to create the dictionary of weights of positive comments. It is need to be trained till the original positive comment will become positive. The next phase is to test the reviews. The reviews will be tested on the basis of the trained weighted dictionary. ANN preforms propagation i.e. back propagation, to train the system, by activation of neurons on hidden layer. This step begins training process of BPN by usingtraining data set. The back-propagation algorithm includes a forward pass and a backward pass. The purpose of the forward pass is to obtain the activation value and. The backward pass is to adjust weights and biases according to the difference between the desired and actual network outputs. These two passes will go through iteratively until the network converges.

The feed-forward network training by back-propagation algorithm can be summarized as follows.

1. For each training pattern (presented in random order):
   - Apply the inputs to the network.
   - Calculate the output for every neuron from the input layer, through the hidden layer(s), to the output layer.
   - Calculate the error at the outputs.
   - Use the output error to compute error signals for pre-output layers.
   - Use the error signals to compute weight adjustments.
   - Apply the weight adjustments.
2. Periodically evaluate the network performance.
3. After the training we test whether the review is positive or negative.

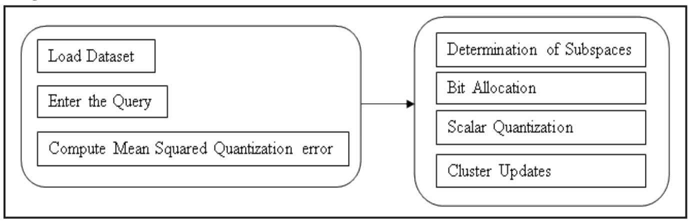

Fig. 1: Proposed System Architecture

V. Conclusion

In this, a novel vector quantization algorithm is proposed for the approximate nearest neighbor search problem. The proposed method explores the quantization centers in affine subspaces through an iterative technique, which jointly attempts to minimize the quantization error of the training samples in the learnt subspaces, while minimizing the projection error of the samples to the corresponding subspaces. It is also shown that, dimension reduction is an important source of quantization error, and by exploiting subspace clustering techniques the quantization error can be reduced, leading to a better quantization performance. This method has proven to outperform the state-of-the-artmethods, with comparable computational cost and additional storage.

References

[1] Dimitris Papadias,"Hill climbing algorithms for content-based retrieval of similar configurations", In Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '00, pp. 240–247, 2000.

[2] Rodrigo Paredes, Edgar Chvez,"Using the k-nearest neighbor graph for proximity searching in metric spaces", In In Proc. SPIRE'05, LNCS 3772, pp. 127–138, 2005.

[3] YuryLifshits, Shengyu Zhang,"Combinatorial algorithms for nearest neighbors, near-duplicates and small-world design", In Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA'09, pp. 318–326, 2009.

[4] P. Indyk, R. Motwani,"Approximate nearest neighbors: Towards removing the curse of dimensionality," In Proc. 30th Annu. ACM Symp. Theory Comput., 1998, pp. 604–613.

[5] J. Wang, H. T. Shen, J. Song, J. Ji,"Hashing for similarity search: A survey", arXiv preprint, 2014, p. 1408.2927.

[6] M. Datar, N. Immorlica, P. Indyk, V. S. Mirrokni, "Locality sensitive hashing scheme based on P-stable distributions", In Proc. 20th Annu. Symp. Comput. Geom., 2004, pp. 253–262.

[7] K. Terasawa, Y. Tanaka,"Spherical LSH for approximate nearest neighbor search on unit hypersphere," In Proc. 10th Int. Conf. Algorithms Data Struct., pp. 27–38, 2007.

[8] X. He, D. Cai, S. Yan, H. Zhang,"Neighborhood preserving embedding," In Proc. 10th IEEE Int. Conf. Comput. Vis., 2005, pp. 1208–1213.

[9] H. Jegou, M. Douze, C. Schmid, P. Perez,"Aggregating local descriptors into a compact image representation," In Proc. IEEE Conf. Comput. Vis. Pattern Recog., 2010, pp. 3304–3311.

[10] W. Dong, M. Charikar, K. Li,"Asymmetric distance estimation with sketches for similarity search in high-dimensional spaces," In Proc. 31st Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, 2008, pp. 123–130.

S Ratna Kumari Holds a B.Tech certificate in Computer Science Engineering from the University of Jntu Kakinada. She presently Pursuing M.Tech (CSE) department of computer science engineering from Baba Institute of Technology and Sciences, Visakhapatnam.

D Durga Prasad is working as an Assistant Professor in the Department of Computer Science and Engineering in Baba Institute of Technology and Sciences.