Efficient Approximate Indexing in High-Dimensional Feature Spaces

Simone Santini *

Escuela Politécnica Superior
Universidad Autónoma de Madrid

Abstract. In this paper we present a fast approximate indexing method for high dimensional feature space that uses the error probability as an independent variable.

The idea of the algorithm is to define a low-dimensional feature space in which a significant portion of the inter-distance variance is concentrated, to search for the nearest neighborhood of the query in this space, and then to extend the search by a factor ζ to include a number of objects "near" this nearest neighborhood. We shall show that, under reasonable hypotheses on the distribution of items in the feature space, it is possible to derive a relation between the value ζ and the error probability.

We study the error probability and the complexity of the algorithm, validate the model using a data set of images, and show how the results can be used to design indexing schemes.

1 Introduction

Attempts to create fast indices in high dimensional metric feature spaces have been (possibly inevitably) hampered by the peculiar characteristics of the metric of such spaces. In particular, given a data base of N elements $D = \{q_1, \dots, q_N\}$ in the space, a point x, and the distance between x and its nearest neighbor $\bar{\Delta} = \min_{q \in D}\{\Delta(x, q)\}$, it is known that a fraction $1 - \epsilon$ of the elements of D will be within a distance $\bar{\Delta} + \epsilon$ from x, where $\epsilon = \Theta(1/N)$ [11].

This phenomenon, colloquially known as the *curse of dimensionality* all but prevents the efficient application of *divide and conquer* techniques that partition the space. Tree-based techniques such as R- and R^*-trees [9,2], K-D-trees [3], SS-trees [13], M-trees [6], etc. or techniques based on variations of Linear Hashing such as *Interpolation-Based Index Maintenance (IBIM)* [4] perform reasonably well for spaces of limited dimensionality, but their performance degrades as the dimensionality of the space increases and, in very high dimensional spaces such as those typical of multimedia data, they are basically equivalent to a linear search, that is, they entail the determination of the distance between the query and all elements in the data base.

Finding the actual nearest neighborhood in high dimensional feature spaces seems inherently hard [5], and there has been a certain interest in approximate algorithms.

* This work was supported by the *Ministerio de Educacion y Ciencia* under the grant N. TIN2011-28538-C02, *Novelty, diversity, context and time: new dimensions in next-generation information retrieval and recommender systems.*

N. Brisaboa, O. Pedreira, and P. Zezula (Eds.): SISAP 2013, LNCS 8199, pp. 194–205, 2013.

Approximate nearest neighbor algorithms [1] guarantee that they will return a point at a distance at most $(1 + \epsilon)$ times that of the nearest neighbor, but have been shown to suffer the curse of dimensionality as well [7]. Probably Approximately Correct (PAC) algorithms [7] return a point with a relative error of $(1 + \epsilon)$ with probability $(1 - \delta)$, and ar more immune from the curse of dimensionality. The algorithm presented in this paper belongs to this family, but it is based on different assumptions than other common algorithms in this class. The algorithm in [7], for example, perform search in the full space, and gains efficiency by finding an early candidate and then using the probability distibution of the distance to cut further search. Here we use a *filtering* approach, similar to that of [12]: we perform a search in a low-dimensional feature space to retrieve a set of *candidates* among which we search the nearest neighbor in the full feature space. We show that, under reasonable hypotheses on the distribution of the points in the space, this allows us to guarantee that the nearest neighbor is found with a certain probability and that, in case it is not found, its distance from the actual nearest neighbor is limited with a given probability.

2 The Approximate Indexing Model

Consider a data base with N objects, each described by a m-dimensional feature vector. A standard procedure in data base indexing is to rotate the feature space in such a way that the variance of the data base is concentrated in a relatively small number of axes. Let $D \in \mathbb{R}^{N \times d}$ a matrix containing all the feature vectors. We can apply singular value decomposition [8] obtaining a decomposition $D = U\Sigma V'$, where $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_d)$ is a $d \times d$ diagonal matrix containing the eigenvalues of D ($\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_d$), V is a $d \times d$ matrix containing the corresponding eigenvectors in the columns, and U is a $N \times d$ orthogonal matrix.

The matrix V defines a transformation into a new feature space in which the data base has variance σ_i in the ith axis. Since the eigenvalues are in descending order, most of the variance is concentrated in the lowest dimensions of the space. In many applications, this property is used for dimensionality reduction, discarding the last dimensions and using only the first ones as a reduced feature space.

This facilitates indexing, as it reduces the dimensionality of the feature space, but it presents two main disadvantages:

i) in order to guarantee acceptable results, the dimensionality reduction is in general insufficient to guarantee efficient search; that is, even in the reduced space, search is linear, and the algorithm must analyze the whole data base in order to determine the nearest neighbor;

ii) discarding the higher elements may result, in certain cases, in retrieving a false nearest neighbor; this is not necessarily a problem in most application (the method that we present here does the same), but a simple pruning of the feature space does not allow the designer to control or at least determine the probability of error.

The method that we present here is based on the same idea of operating in spaces of reduced dimensionality, but it permits to operate a first search on a much smaller space (thereby taking better advantage of fast search methods), and permits the designer to

tune the method to reach a controllable compromise between efficiency and precision. We partition the feature space F into two orthogonal subspaces X and Y (viz. $F = X \oplus Y \approx X \times Y$) where X contains the first few axes of the transformed space, with a variance σ_X in the distribution of features, and Y contains the rest of dimensions with a variance σ_Y. We choose the dimension of X in such a way that

$$\nu = \frac{\sigma_X^2}{\sigma_Y^2} > 1 \tag{1}$$

We then proceeds as follows:

i) given a query point q, we find the nearest neighbor n using only the feature space X; let $d(q, n)$ be the distance between q and n in the sub-space X;

ii) we increase the square of the distance by an amount α, and we do a range query on the space X again for all items x such that $d^2(q, x) \leq d^2(q, n) + \alpha$; let T be the set of items resulting from this search, i.e.

$$T_q = \{x | d^2(q, x) \leq d^2(q, n) + \alpha\} \tag{2}$$

iii) We use the whole space, with a distance measure Δ, to find the closest neighbor to q in T_q:

$$r = \arg\min_{x \in T_q} \Delta^2(q, x) \tag{3}$$

If X has a small dimensionality, the first and the second steps can be done in time $|T_q| \log N$, while the third requires at most time $|T_q|$.

The hypothesis that we make is that, if $\nu > 1$ (viz. if the variance of the data base is sufficiently concentrated in X), most of the times the true nearest neighbor will be close enough to the one that we estimated in step i) using only the distance function of X, therefore even with a small value of α (that is, with a small T_q), we shall be able to retrieve the true nearest neighbor with sufficient probability.

Let distances be Euclidean. Let $d : X \times X \to \mathbb{R}^+$ be the distance in the space X, $\delta : Y \times Y \to \mathbb{R}^+$ the distance function in Y, and $\Delta : F \times F \to \mathbb{R}^+$ the distance function in F, with $\Delta^2 = d^2 + \delta^2$.

In step i), we do a search on X, and we find a "bogus" nearest neighbor, b, at a distance $\Delta^2(q, b) = \Delta_b^2 = d_b^2 + \delta_b^2$. Suppose we have missed the true nearest neighbor. Then there is an item t at a distance $\Delta_t^2 = d_t^2 + \delta_t^2$ with $\Delta_t^2 < \Delta_b^2$.

We have found b as the nearest neighborhood on X, which entails $d_b \leq d_t$. Not only, we have checked all items that, on X, were within a distance $d_b^2 + \alpha$ from the query. Had the true nearest neighbor been within this distance on X, it would have been included in T_q and in step iii) we should have found it. If we have made a mistake, then $t \notin T_q$, that is, $d_t^2 > d_b^2 + \alpha$. The situation is illustrated in figure 1. The true nearest neighbor, t, is, in X, at a distance $d_t^2 > d_b^2 + \alpha$, therefore will not be included in T_q and will not be detected. Nevertheless, because of its very small component δ_t, its distance from the query is smaller than that of the detected nearest neighbor.

We are interested in the probability that this happens, that is, in the probability

$$\mathbb{P}_E = \mathbb{P}\{\Delta_t^2 < \Delta_b^2 | d_t^2 - d_b^2 > \alpha\} = \mathbb{P}\{d_t^2 - d_b^2 < \delta_b^2 - \delta_t^2 | d_t^2 - d_b^2 > \alpha\} \tag{4}$$

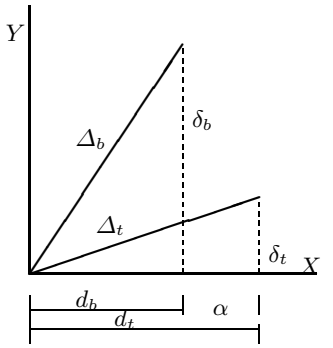

Fig. 1. A situation in which the algorithm leads to a wrong detection of nearest neighbor: the nearest neighbor that is being detected on X is b, with a distance Δ_b from the query. In X, the true nearest neighbor is at a distance greater than $d_b^2 + \alpha$ from the query, so it is not retrieved. Nevertheless, the component δ, which is not considered in the determination of T_q, is much smaller than that of the detected nearest neighbor, so that $\Delta_t < \Delta_b$.

If the objects have Gaussian distribution in the data base, the distances have a χ^2 distribution with a number of degrees of freedom that depends on the dimension of the respective sub-spaces[1]. We shall make a significant approximation, and assume that the distances have a χ^2 distribution with two degrees of freedom, that is, that they are exponentially distributed. The variance of these distributions is the variance of the data bases in the respective sub-spaces, i.e.

$$p_d(u) = \frac{1}{2\sigma_X^2}\exp(-\frac{u}{2\sigma_X^2}) \quad p_\delta(u) = \frac{1}{2\sigma_Y^2}\exp(-\frac{u}{2\sigma_Y^2}) \tag{5}$$

The differences of distances that appear in $\mathbb{P}_E$ have a Laplace distribution:

$$\begin{aligned} p_{\Delta d}(u) &= \frac{1}{4\sigma_X^2}\exp(-\frac{|u|}{2\sigma_X^2}) \\ p_{\Delta\delta}(u) &= \frac{1}{4\sigma_Y^2}\exp(-\frac{|u|}{2\sigma_Y^2}) \end{aligned} \tag{6}$$

So we have

$$\begin{aligned} \mathbb{P}_E &= \int_\alpha^\infty dx p_{\Delta d}(x) \int_x^\infty dy p_{\Delta\delta}(y) \\ &= \int_\alpha^\infty dx \frac{1}{4\sigma_X^2}\exp(-\frac{x}{2\sigma_X^2}) \int_x^\infty dy \frac{1}{4\sigma_Y^2}\exp(-\frac{y}{2\sigma_Y^2}) \\ &= \frac{1}{2}\frac{\sigma_Y^2}{\sigma_X^2+\sigma_Y^2}\exp\left[-\frac{1}{2}\frac{\sigma_X^2+\sigma_Y^2}{\sigma_Y^2}\frac{\alpha}{\sigma_X^2}\right] \end{aligned} \tag{7}$$

[1] Not all data sets have a Gaussian distribution. However, as we shall see when comparing with sampled data, in many cases the distributions that we obtain assuming a Gaussian distribution do have a reasonable fit with the distribution of the actual data.

The parameter α is the additional distance that we consider in the space X. The parameter $\zeta = \alpha/\sigma_X^2$ can be seen as a distance expressed in units of σ_X^2, that is, as a form of natural distance, a number that does not depend on the unit that we use to measure distances and coordinates in X, but only on the distribution of the data base. With this definition we have:

$$\mathbb{P}_E = \frac{1}{2(1+\nu)} \exp\left[-\frac{1+\nu}{2}\zeta\right] \tag{8}$$

If we look for the nearest neighbor in X and then we "peek" forward for a (natural) distance ζ, this is the probability that we will make a mistake in the determination of the nearest neighbor.

In general, we are more interested in using this formula the other way around: given a target error probability p, we want to know how far we have to *peek* in the data base in order to attain it. This value is given by

$$\zeta = \frac{2}{1+\nu} \log \frac{1}{2(1+\nu)p} \tag{9}$$

3 Complexity

if X is a low-dimensional feature space, the search for the nearest neighbor in X takes time $\log N$. If there are m objects within a distance ζ from the nearest neighbor, we can find them in time m (this can be done, for example, using methods based on linear hashing [4]). A further time m goes into finding the nearest neighbor in the whole feature space among these objects (we assume that the whole feature space is high dimensional, so that no indexing method performs better than linear scan). The total execution time is therefore $\log N + 2m$. The first term is fixed, so in order to determine the complexity of the algorithm we have to estimate the value of m. We find the nearest neighbor at a distance d_b^2 from the query, and we keep searching for a distance α. That is, we put an element u in T_q if $d_u^2 - d_b^2 < \alpha$. With the distribution for the difference of distances given in 6, we have

$$\begin{aligned}
\mathcal{P}\{d_u^2 - d_b^2 < \alpha | d_u^2 > d_b^2\} &= \frac{\mathcal{P}\{d_u^2 - d_b^2 < \alpha, d_u^2 > d_b^2\}}{\mathcal{P}\{d_u^2 > d_b^2\}} \\
&= 2\frac{1}{4\sigma_X^2} \int_0^\alpha dx \, \exp[-\frac{x}{2\sigma_X^2}] \\
&= 1 - \exp[-\frac{\alpha}{2\sigma_X^2}] \\
&= 1 - \exp[-\frac{\zeta}{2}] \\
&\stackrel{\text{def}}{=} \pi(\zeta)
\end{aligned} \tag{10}$$

The probability that in a data base of N elements, n are at a distance greater than ζ is then

$$\begin{aligned}
\mathbb{P}_\zeta[n] &= \binom{N}{n} \pi(\zeta)^n (1-\pi(\zeta))^{N-n} \\
&\approx \frac{(N\pi(\zeta))^n}{n!} \exp[-N\pi(\zeta)]
\end{aligned} \tag{11}$$

where the Poisson approximation is valid for N large. Then

$$m(\zeta) = \mathbb{E}[\mathbb{P}_\zeta[n]] = N\pi(\zeta) = N\left(1 - \exp\left(-\frac{\zeta}{2}\right)\right) \tag{12}$$

If we are interested in the complexity for a prescribed error probability p, we plug in eq. (9) obtaining:

$$\begin{aligned} m(p) &= N\left(1 - \exp\left(-\frac{1}{1+\nu}\log\frac{1}{2(1+\nu)p}\right)\right) \\ &= N\left[1 - (2(1+\nu)p)^{\frac{1}{1+\nu}}\right] \\ &\stackrel{\text{def}}{=} N\mu(\nu, p) \end{aligned} \tag{13}$$

The value $\mu(\nu, p)$ represents the fraction of the data base that is necessary to search in order to obtain the nearest neighborhood with probability p.

* * *

Before proceeding to the validation of the model and to some general considerations on its predictions, we summarize here the results obtained. Remember that we divide the search space into a low-dimensional one in which the distance measures have a variance σ_X^2 and a high dimensional one in which the variance is $\sigma_Y^2 < \sigma_X^2$.

i) If we search the nearest neighbor in the low-dimensional space and we extend the search for a distance $\alpha = \zeta\sigma_X^2$ we find the nearest neighbor with a probability:

$$\mathbb{P}_S(\zeta) = 1 - \mathbb{P}_E(\zeta) = 1 - \frac{1}{2(1+\nu)}\exp\left(-\frac{1+\nu}{2}\zeta\right) \tag{14}$$

ii) Correspondingly, if we want to find the nearest neighbor with probability $1 - p$ we shall have to find it X and then extend the search up to a distance $\alpha = \zeta\sigma_X^2$, with

$$\zeta = \frac{2}{1+\nu}\log\frac{1}{2(1+\nu)p} \tag{15}$$

iii) Given a value of ζ, the average complexity of the search (assuming that search is efficient in the low-dimensional space) is

$$C(\zeta) = \log N + N\left(1 - \exp\left(-\frac{\zeta}{2}\right)\right) \tag{16}$$

iv) If we want to find if we want to find the nearest neighbor with probability $1 - p$, the complexity is

$$C(p, \nu) = \log N + N\left[1 - (2(1+\nu)p)^{\frac{1}{1+\nu}}\right] = \log N + N\mu(\nu, p) \tag{17}$$

4 Validation

For the purpose of model validation, we use the *Im2Text* data base [10]. The data base contains 1.000.000 annotated images, although in this case we didn't use the captions. All the statistical results are based samples of 10.000 images selected at random from the data base, with several independent sampling used in order to confirm that the values are significant.

As a feature vector, we used a block histogram on the luminance channel of the images. Images were resized to a standard size of 1024 by 1024 and divided into blocks of size 128 (8 by 8 blocks). For each block, the mean and variance of the histogram were computed and united to form a feature vector of 128 double precision numbers. Singular value decomposition was then applied so as to obtain a feature space (the *transformed space*) in which the variance was concentrated in a relatively small number of dimensions.

The distributions of interests for this data base are shown in figure 2 for one of the samples. All the distribution have been obtained through random samples of the data set [10] and cross-validated using other data sets: they are highly representative of the distributions that one obtains when applying singular value decomposition to actual data sets in a wide range of multimedia applications (which are the ones that motivated this work), and reasonably representative of other high-dimensional data sets.

Figure 2(a) shows the singular values obtained through singular value decomposition; figure 2(b) shows the distribution of the distance d^2 between images (computed over the whole feature space), and figure 2(c) shows the distribution of the distance difference $d_i^2 - d_j^2$. The latter is superimposed to the theoretical model that we use in this paper.

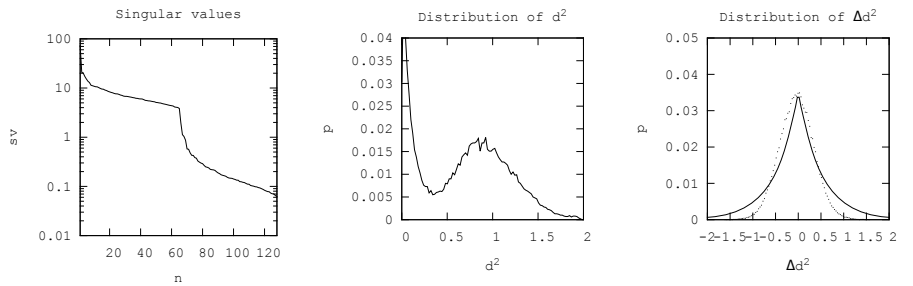

Fig. 2. The distributions of interest for the validation set: the singular values (a), the distribution of the inter-image distances d^2 (b), and the distribution of the distance differences $d_i^2 - d_j^2$ (c). The distance differences are superimposed to the theoretical Laplace distribution, for comparison purposes.

The coincidence is not perfect (for high dimensional feature spaces the actual difference is a Normal distribution). Moreover, the distribution of d^2 doesn't look at all

like the exponential that the model hypothesizes. The distribution of d^2 for features of reduced dimensions (the first n components of the features in the transformed space) is shown in figure 3 This is not necessarily a problem for our model, as the distribution

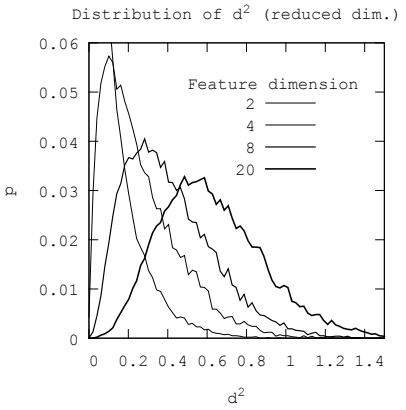

Fig. 3. The distribution of the square of the inter-image distance (d^2) for feature spaces of reduced dimensionality n

of d^2 never enters the equations: only the distribution of Δ^2 does and this, even in the worst case of a high dimensional feature space, is close enough to the actual distribution so that the advantage of a closed-form solution, afforded by the Laplace distribution outweighs the disadvantages of the imprecision of the model.

Figure 4 shows the comparison between the theoretical value of the error probability and the measured one as a function of ν for several values of ζ.

The model underestimates somewhat the actual performance for the highest values of ν (viz., in this experiments, when the initial search is done in a space of dimension 10 or more), as the experiment always finds the nearest neighbor. . As ν increases, the theoretical model converges rapidly to 1, so the model and the measurements converge once again (albeit trivially: the area of very high ν will result in searches that are just as inefficient as those carried out on the whole feature vector).

We have attempted a validation of the search complexity (viz. of the number $m(\zeta)$). The results obtained were formally significant, but the variance of the number of objects actually retrieved in the data base was so high that almost any reasonable estimation of m would have been correct. This might indicate that the complexity depends on parameters other than ν and ζ, parameters that, left uncontrolled in our validation, cause the high variance. For the purposes of the following section, we shall take a formalist point of view and assume that, since the prediction is well within the variance of the data, it can be assumed to be correct.

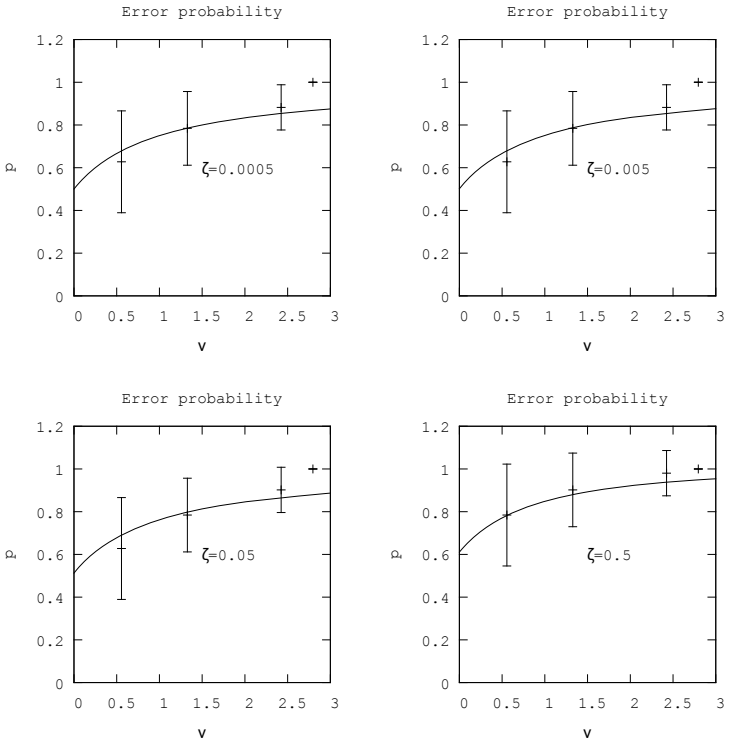

Fig. 4. Comparison between the error probability in finding the nearest neighbor predicted by the model and that measured on the Im2Text data base using the features described in the paper

5 Some Analytical Considerations

We begin by noticing that, in equation (14), if $\zeta = 0$, then

$$\begin{aligned} \mathbb{P}_E(0) &= \frac{1}{2(1+\nu)} \stackrel{\text{def}}{=} \mathbb{P}_E^{\uparrow}(\nu) \\ \mathbb{P}_S(0) &= \frac{1+2\nu}{2(1+\nu)} \stackrel{\text{def}}{=} \mathbb{P}_S^{\downarrow}(\nu) \end{aligned} \tag{18}$$

$\mathbb{P}_S^{\downarrow}(\nu)$ is the minimal success probability (for a given value of ν) that requires looking beyond the nearest neighbor found in the first step of the algorithm. If the desired success probability is less than $\mathbb{P}_S^{\downarrow}(\nu)$ (or, equivalently, if the desired error probability is higher than $\mathbb{P}_E^{\uparrow}(\nu)$) then one can simply search for the nearest neighbor in the reduced feature space. This cut probability is shown in table 1 for various values of ν. As expected, for high values of ν it is often a good strategy to simply ignore the rest of the feature space (this is what $\zeta = 0$ amounts to) and do the search in the reduced space. For smaller ν and/or if a very high success probability is desired, then the full algorithm should be applied.

Table 1. The probability of success $\mathbb{P}_S^{\downarrow}$ beyond which it is convenient just to search the reduced feature space, as a function of ν

ν	0.2	0.5	1	2	4	8	16	20
$\mathbb{P}_S^{\downarrow}$	0.583	0.667	0.750	0.833	0.900	0.944	0.971	0.976

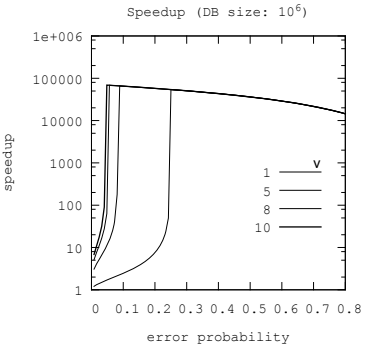

Fig. 5. Speedup of the method proposed here with respect to the baseline. The saturation point depends on the size of the data base, being roughly proportional to $N/\log N$. The values shown are for $N = 10^6$. After saturation the curve decreases as $N(1-p)/\log N$.

In order to compare the proposed method with a baseline, we consider a very simple method that also finds the nearest neighbor with a prescribed probability. The method is simply this: given the desired error probability p, reduce the data base considering a random sample of $N(1-p)$ objects, and search the nearest neighbor only in this reduced data base. It is easy to see that the search will yield the actual nearest neighbor with a probability p and even with a linear method, we have reduced the search time by a factor $(1-p)$.

The complexity of the method proposed here, for a prescribed error probability, is given by eq. (17). The speed-up of this method, with respect to to the baseline, is:

$$\rho(p,\nu) = \frac{N(1-p)}{\log N + N\mu(p,\nu)} = \frac{1-p}{\frac{\log N}{N} + \mu(p,\nu)} \tag{19}$$

For large data bases, $(\log N)/N$ is very small, so we can approximate the speed-up as

$$\rho(p,\nu) = \frac{1-p}{\mu(p,\nu)} = \frac{1-p}{1-(2(1+\nu)p)^{\frac{1}{1+\nu}}} \tag{20}$$

It is easy to see that, for all $\nu > 0$, $(2(1+\nu)p)^{\frac{1}{1+\nu}} > p$, from which we derive $\rho(p,\nu) > 1$, i.e. using the low-dimensional space as a filter always yields better performance (apart from the negligible factor $(\log N)/N$). Figure 5 plots the logarithm of the speed-up as a function of p for various values of ν. Note that in the graph the value of ρ as given

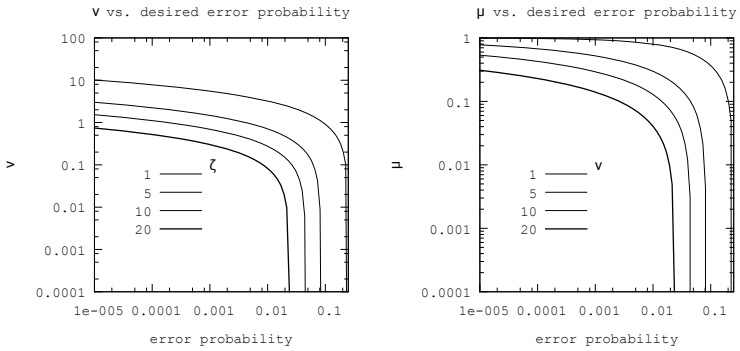

Fig. 6. Values of ζ (a) and μ (b) as a function of the desired error probability for several values of ν

by (20) goes to infinity. This doesn't happen in practice because, when $\mu(p, \nu)$ becomes zero, that is, for $p = \mathbb{P}_S^{\downarrow}(\nu)$, the term $(\log N)/N$ dominate. The speed up, therefore, increases with p according to (20) for $p < \mathbb{P}_S^{\downarrow}(\nu)$, and has a value $\rho = N(1-p)/\log N$ for $p > \mathbb{P}_S^{\downarrow}(\nu)$.

Finally, we propose some of the data to be used to the design of indexing solution using the method introduced here. For the designer, the most important independent variable is the error probability p. Given a system with a low dimensional space (with an efficient index) and a desired error probability p, the designer will derive the value of ζ used in the second step of the method of page 196 and then will estimate the complexity of the method, that is, the value of μ. Figure 6.a shows the predicted values of ζ as a function of the desired error probability for given values of ν. Figure 6.b shows the value of μ (which determines the execution time of the algorithm) as a function of the desired error probability for various values of ν.

6 Some Final Remarks

One of the major uncertainties of the model, as it stands now, is the high variance measured in the measurement of the number of accessed data items as a function of ζ for a given value of ν. This makes the model formally correct, but it makes its complexity predictions quite imprecise. The most likely explanation is that the complexity depends on variables other than ν and ζ and the fact that these variables are not controlled causes the high variance. A more detailed model is necessary to reveal what are these variables.

The algorithm, as it is, begins by looking for the nearest neighbor in the space X. This element is needed only in order to determine its distance d^2 from the query so that we can determine the range to search in X as $d^2 + \zeta\sigma_X^2$. It should be possible to avoid this step–and consequently eliminate the factor $\log\ N$ from the complexity expression– by deriving a statistical characterization of d^2. This would make the algorithm faster

for a given value of ζ, but the additional uncertainty introduced by the statistical determination will probably require a larger ζ for a given target error probability. It will be necessary to analyze whether, and in what circumstances, this is advantageous.

References

1. Arya, S., Mount, D.M., Netanyahu, N.S., Silverman, R., Wu, A.Y.: An optimal algorithm for approximate nearest neighbor searching in xed dimensions. Journal of the ACM 45(6), 891–923 (1998)
2. Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B.: The r^*-tree: an efficient and robust access method for points and rectangles. In: Proceedings of the ACM SIGMOD International Conference on the Management of Data (1990)
3. Bentley, J.L.: K-D trees for semidynamic point sets. In: Proceedings of the Sixth ACM Annual Symposium on Computational Geometry (1990)
4. Burkhard, W.: Interpolation/based index maintenance. In: Proceedings of the ACM Symposium on Principles of database systems (PODS), pp. 76–89 (1983)
5. Chávez, E., Navarro, G., Baeza-Yates, R., Marroquín, J.L.: Searching in metric spaces. ACM Computing Surveys (CSUR) 33(3), 273–321 (2001)
6. Ciaccia, P., Patella, M., Zezula, P.: M-tree: An efficient access method for similarity search in metric spaces. In: Proceedings of the 23rd VLDB (Very Large Data Bases) Conference, Athens, Greece (1997)
7. Ciaccia, P., Patella, M.: Pac nearest neighbor queries: Approximate and controlled search in high-dimensional and metric spaces. In: Proceedings of the International Conference on Data Engineering, ICDE (2000)
8. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. Journal of the American Society for Information Science 41(6), 391–407 (1990)
9. Gravila, D.M.: The R-Tree index optimization. In: Waugh, T., Healey, R. (eds.) Advances in GIS Research. Taylor & Francis (1994)
10. Ordoñez, V., Kulkarni, G., Berg, T.L.: Im2text: Describing images using 1 million captioned photographs. Neural Information Processing Systems (2011)
11. Pestov, V.: On the geometry of similarity search: dimensionality curse and concentration of measure. Information Processing Letters 73(1-2), 47–51 (2000)
12. Seidl, T., Kriegel, H.-P.: Optimal multi-step k-nearest neighbor search. In: Proceedings of the ACM SIGMOD, International Conference on Management of Data, Seattle, USA, pp. 154–165 (1998)
13. White, D., Jain, R.: Similarity indexing with the SS-tree. In: Proc. 12th IEEE International Conference on Data Engineering (1996)