High-Dimensional Similarity Retrieval Using Dimensional Choice

Dave Tahmoush and Hanan Samet

University of Maryland, College Park
tahmoush at cs.umd.edu

Abstract

There are several pieces of information that can be utilized in order to improve the efficiency of similarity searches on high-dimensional data. The most commonly used information is the distribution of the data itself, but the use of dimensional choice based on the information in the query as well as the parameters of the distribution can provide an effective improvement in the query processing speed and storage. The use of this method can produce dimension reduction by as much as a factor of n, the number of data points in the database, over sequential search. We demonstrate that the curse of dimensionality is not based on the dimension of the data itself, but primarily upon the effective dimension of the distance function. We also introduce a new distance function that utilizes fewer dimensions of the higher dimensional space to produce a maximal lower bound distance in order to approximate the full distance function. This work has demonstrated significant dimension reduction, up to 70% reduction with an improvement in accuracy or over 99% with only a 6% loss in accuracy on a prostate cancer data set.

1. Introduction

In order to create an effective classification technique for bioinformatics data, methods are needed to efficiently retrieve data based on similarity to a given exemplar or set of exemplars. This type of query is referred to as similarity retrieval. Of these queries, the nearest neighbor query is particularly important, and it is the one that is emphasized in this paper. An apparently straightforward solution to finding the nearest neighbor is to compute a Voronoi diagram for the data points (i.e., a partition of the space into regions where all points in the region are closer to the region's associated data point than to any other data point), and then locate the Voronoi region corresponding to the query point. The problem with this solution is that the combinatorial complexity of the search process in high dimensions, expressed in terms of the number of objects, is prohibitive thereby making it virtually impossible to store the Voronoi diagram which renders its applicability moot.

The problem described above is typical of the issues that must be faced when dealing with high-dimensional data. Multidimensional problems such as these queries become increasingly more difficult to solve as the dimensionality increases. The difficulties that are encountered are attributed to the curse of dimensionality which surfaces in a number of different forms. In essence, the term was coined by Bellman [3] to indicate that the number of samples needed to estimate an arbitrary function with a given level of accuracy grows exponentially with the number of variables (i.e., dimensions) that comprise it. For similarity searching (i.e., finding nearest neighbors), this means that the number of objects (i.e., points) in the data set that need to be examined in deriving the estimate (i.e., the nearest neighbor) grows exponentially with the underlying dimension.

The curse of dimensionality has a direct bearing on similarity retrieval in high dimensions in the sense that it raises the issue of whether or not nearest neighbor searching is even meaningful in such an environment. In particular, it has been shown that for data and queries drawn from a uniform distribution, the distance to the nearest neighbor and the distance to the farthest neighbor tend to converge as the dimension increases [20]. This is why dimension reduction is an important issue in classification.

Assuming that the distance d is a distance metric (which is the case for the commonly used Minkowski metric L_p), and hence that the triangle inequality holds, an alternative way of understanding the ramifications of the curse of dimensionality is to observe that when dealing with high-dimensional data, the probability density function (analogous to a histogram) of the distances of the various elements is more concentrated and has a larger mean value. This means that similarity searching algorithms will have to perform more work. In the worst case, for an arbitrary object x, there is the situation where $d(x,x)=0$ and $d(x,y)=1$ for all $y \neq x$, which means that a similarity query must compare the query object with every object of the set. One way to see why more concentrated probability densities lead to more complex similarity searching is to observe that this means that the triangle inequality cannot be used so often to eliminate objects from consideration. In particular, the triangle inequality implies that every element x such that $|d(q,p)-d(p,x)|>\varepsilon$ cannot be at a distance of ε or less from q (i.e., from $d(q,p) \leq d(p,x)+d(q,x)$). For the probability density function of $d(p,x)$, when ε is small while the probability density function is large at $d(p,q)$, then the probability of eliminating an element from consideration via the use of the triangle inequality is the remaining area under the curve, which is quite small (see Figure 1a in contrast to Figure 1b where the density function of the distances is more uniform).

The high dimensionality of the data also has an effect on the search process which is aided by the presence of indexes. In particular, for uniformly distributed high-dimensional data,

This work was supported in part by the U.S. National Science Foundation under Grants EIA-00-91474, CCF-05-15241, and IIS-0713501, as well as NVIDIA Corporation and Microsoft Research.

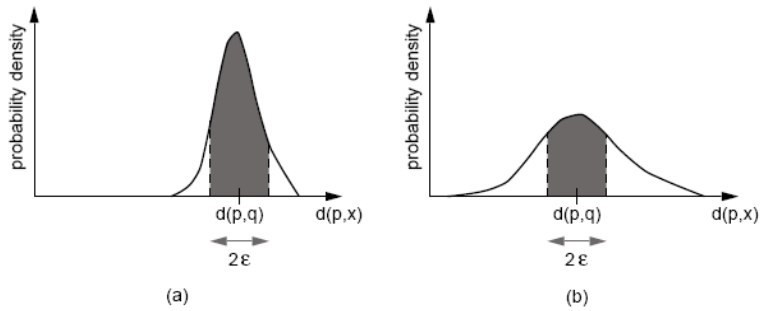

Figure 1: A probability density function (analogous to a histogram) of the distances d(p,x) with the shaded area corresponding to $|d(q,p)-d(p,x)| < \varepsilon$. (a) indicates a density function where the distance values have a small variation, while (b) indicates a more uniform distribution of distance values thereby resulting in a more effective use of the triangle inequality to prune objects from consideration as satisfying the range search query.

most of the data lies near the boundary of the underlying space (e.g., [4]) and thus most indexes result in visiting all of the index blocks. This has led to the use of methods based on a sequential scan (e.g., [5, 11, 29]). However, these methods also make use of a variant of an index in the sense that they resort to the use a compressed index on the data to speed up the sequential scan.

A number of methods have been proposed to overcome the curse of dimensionality. One approach is to observe that the data is rarely uniformly distributed which leads to pointing out that some dimensions are more significant than others thereby focusing on them (e.g., [16, 19, 20]). Such methods are also known as dimension-reduction techniques and some examples include SVD [18] and the Discrete Fourier Transform (DFT) [15]. The traditional and the state-of-the-art dimensionality reduction methods can be generally classified into feature extraction [22, 23, 26] and feature selection [6, 9, 33] approaches. In general, feature extraction approaches are more effective than the feature selection techniques [27, 28, 32] and they have shown to be very effective for real-world dimensionality reduction problems [10, 13, 22, 23]. Many scalable online FE algorithms have been proposed. Incremental PCA (IPCA) [2, 24] is a well-studied incremental learning algorithm. The latest version of IPCA is called Candid Covariance-free Incremental Principal Component Analysis (CCIPCA) [30]. However, IPCA ignores the valuable label information of data and is not optimal for general classification tasks. The Incremental Linear Discriminant Analysis (ILDA) [12] algorithm has also been proposed recently. Another feature extraction algorithm is called Incremental Maximum Margin Criterion (IMMC) [31].

These methods utilize the information in the data and adjust the process to choose the best dimensions, but do not choose the best dimensions for each individual query point in order to improve the performance. Part of this paper explores the effectiveness of adjusting the retrieval process in response to the query point. Making use of the dimensions where the query point is near to a boundary instead of near the middle of the range provides a higher probability of pruning with that dimension. This method is significantly improved when distance functions with a higher order are used because the large contributions of a few dimensions are more relevant in that case. We also try to guarantee to not be worse than sequential search.

Nearest neighbor retrieval is a basic method used for classification. However, because of the curse of dimensionality, the difference in the distance to one class or the other becomes minimal and the accuracy suffers, prompting the use of methods like support vector machines (SVM) [14]. In this paper we compare nearest and farthest neighbor classifications that have been modified with our high-dimension techniques with SVM classifications to determine whether the curse of dimensionality has been reduced.

Nearest neighbor techniques often use the Minkowski metrics to measure similarity between data points. However, the L_2-norm is not necessarily relevant to many emerging applications involving high-dimensional data [1]. Often these are used after dimension-reduction techniques like SVD. We experiment with a new reduced-dimension distance function that is designed to rapidly determine the maximum lower bound on the high-dimensional distance.

In high-dimensional nearest neighbor there are both indexed methods like the GESS method [8] and grid structures [21], and then there are the unindexed methods like the Epsilon Grid Order or EGO [7]. The method in this paper is an unindexed approach.

The rest of this paper is organized as follows. Section 2 discusses the two techniques advanced in this paper, a lower-dimensional approximation to a distance function and an improved method for choosing the best dimensions to use to determine similarity. Section 3 discusses and experiment in colon cancer data retrieval and classification. Section 4 reveals the results and Section 5 discusses the conclusions.

2. Algorithms

Several new approaches are discussed in this paper, including choosing the dimensions to analyze based on the

dimensions that are relevant to both the data and the query point, called dimensional choice, and a new distance function that measures the maximum lower bound on the high-dimensional distance, called the UL-Distance. The combination of improved dimensional ranking and a distance function that uses fewer dimensions is shown to be an effective combination.

2.1 UL-Distance

Normal nearest neighbor approaches break up the feature space well, but are susceptible to bad data points. A typical distance metric that is used is a Minkowski metric of order U as in Equation 1.

1) $$Dist = \sqrt[U]{\sum_{i=0}^{d} \left(| x_i - q_i |\right)^U}$$

where d is the number of dimensions in the feature space. Here $x_i - q_i$ indicates the difference in the data point x and the point to be classified q in the ith dimension. The Euclidean distance function uses a value of U=2, while the Manhattan distance function uses U=1. We define a new non-metric distance function called the UL-Distance which is defined to be a maximum lower limit on a high-dimensional Minkowski distance metric in Equation 2.

2) $$UL - Dist = \sqrt[U]{\sum_{i=0}^{L} \max_i \left(| x_i - q_i |\right)^U}$$

The max function here picks out the ith maximum from the set. The number of dimensions used in the distance is expressly limited to L, which will speed up retrieval, but the contributions are from the dimensions that will maximize the distance in order to maintain as much accuracy as possible. The factor L is the number of dimensions used in the distance, and the U is the order of the distance function. Though this technique provides an approximation to the high-dimensional distance that can be used for pruning, it can also be an effective distance function on its own. Note that it is not a distance metric because the triangle inequality is not guaranteed to work. This distance function is sensitive to the few dimensions that are different, instead of being overwhelmed by the number of dimensions that are similar. A second non-metric distance function called the LL-Distance is nearly identical to the UL-Distance except it calculates the minimum lower limit and is Equation 3.

3) $$LL - Dist = \sqrt[U]{\sum_{i=0}^{L} \min_i \left(| x_i - q_i |\right)^U}$$

In order to provide an example of why using a distance function that uses all of the dimensions but only calculates with a few could be significantly faster than a Euclidean distance, we use a chessboard distance and calculate the farthest neighbor as shown in Figure 2. Though this may not be the most useful calculation, it is the simplest example.

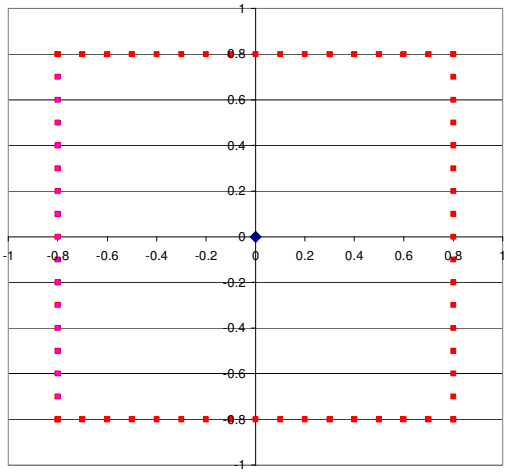

Figure 2: Worst-Case 2D Search. The data in this example are the small squares, while the query point is the diamond at (0,0). The points on the diameter are stored in the data structure, or a single point can be used if space is an issue. All of the data points are equidistant from the query point, requiring all four surfaces to be accessed to find all farthest neighbors in the chessboard distance function.

When doing a farthest-neighbor search, using the diameter of the data can be an effective technique. Since the chessboard distance metric requires finding the point with the maximum difference in only one dimension, storing the points that are on the diameter of the data set allows the lookup of the farthest neighbor in each dimension. The point (or points if multiple points with the same value are stored) on the diameter of each dimension is compared with the query point, the dimension with the maximum is determined, and the farthest neighbor is looked up. This would allow the calculation of the farthest neighbor under this distance metric in O(d) time with O(d) storage. This is the same amount of work necessary to calculate one distance, and is a factor of n better than sequential search. This example demonstrates that the dimensionality of the distance function is very important for the performance of retrievals.

Since the number of dimensions used by the UL-Distance function is limited to L, this distance function is effectively a dimension reduction technique that operates on all of the dimensions. An additional technique is required to make the retrieval faster.

These distance functions do fulfill the properties of positiveness, where D(a,b) >= 0 for all a and b, symmetry, where D(a,b) = D(b,a), and identity, where D(a,a) = 0. However, it is not guaranteed to fulfill the triangle inequality because different dimensions are used.

An alternative approach to limiting the number of dimensions is to only take dimensions that contribute more than a certain threshold. However, that makes comparisons with other techniques difficult. Note that these distance functions are not normalized so that comparing the distances with different values of L can be misleading, which makes the threshold approach more difficult. Normalization is feasible for particular values of U.

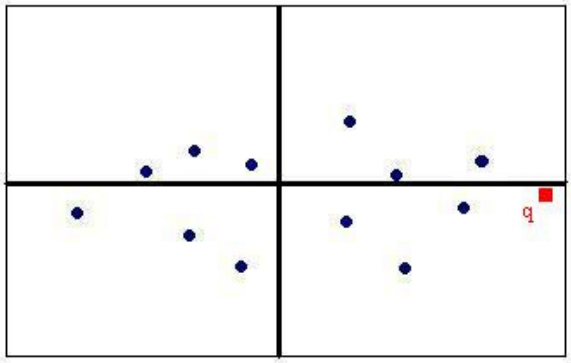

Figure 3: An example 2D data set where choosing to search using the x-dimension is preferred. The red square q is the query point, and the blue circles are the data. In this case the x-dimension is very significant for determining the nearest neighbor, while the contributions from the y-dimension are not as significant and the y-dimension could be neglected. Dimensional Choice would let us choose the x-dimension and ignore the y-dimension unless it is needed.

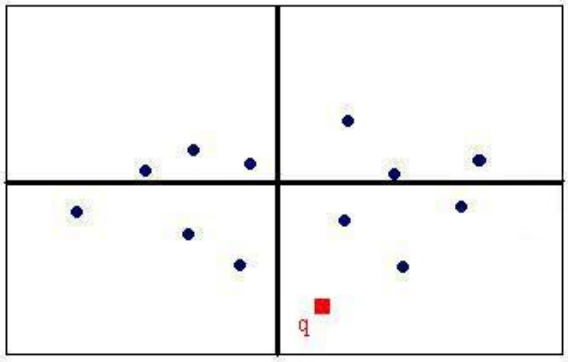

Figure 4: An example 2D data set where choosing to search using the y-dimension is valuable in determining the nearest neighbor to the query point q, even though the data set would indicate that a x-dimension is preferred. The red square q is the query point, and the blue circles are the data. In both the x and y dimensions, the contributions to the total distance can be significant. The difference between this case and the case in Figure 3 is the position of the query point.

2.2 Search using Dimensional Choice

In low-dimensional search, the choice of which dimension to incorporate into the search first is not that important. However, when there are thousands of dimensions in the data set, the choice is much more important. Choosing the best dimension to start the search does require additional work to determine the best dimension, with work of O(d) to O(d log d) to sort the dimensions, as well as knowledge of the diameters of the data set. However, this is only a small amount of work compared to a complete high-dimensional search, which for sequential search is O(nd) where n is the number of data points and d is the number of dimensions. Additionally, if one is using SVD as is highly recommended when using high-dimensional data, the initial transformation or projection into the SVD coordinates dominates the work required to implement dimensional choice.

The underlying data structure must be extremely flexible in order to utilize dimensional choice, which is why it is not used in low-dimensional cases. The idea behind this technique was mentioned by Nene and Nayar [25], where they suggest ordering the analysis of the dimensions in order to minimize the total work. However, they were working with only sixteen dimensions, so we analyze the full effect of this technique on their projection method. Their technique determines the points that are within a distance of ε from the query point by accessing the data in each dimension and winnowing down the potential nearest neighbors. The distance ε that should be used is determined to be rather large when there are sparse data points and a large number of dimensions.

Dimensional choice can be used to first estimate the dimensions that have the largest potential to winnow down the number of potential nearest neighbors without actually analyzing those dimensions. This choice is distribution dependent and could be calculated as such. Note that when the query point is at the edge of the data set, the space that has to be searched is only ε instead of 2ε (since the range is from x-ε to x+ε and in this case half of the space will be empty). So in the case of the uniform distribution (where this technique works the worst), there are dimensions that can as much as double the effective winnowing. In the case of Gaussian distributed data, the effect is even better because the winnowing is done at the tails of the distribution. Additionally, the important dimensions can be determined a priori, so that many dimensions need never be analyzed.

In order to realize the effect of the improved winnowing, an additional adjustment should be included to the Nene and Nayar approach. Their approach continues through all of the dimensions regardless of the number of points remaining in the hyper-cube. A stopping condition should be included so that analyzing the dimensions stops when there are a set number of points left in the hyper-cube. Using dimensional choice reduces the work by at least a factor of two for a uniform distribution in high dimensions, and a significantly better factor for a Gaussian distribution.

PCA analysis utilizes a limited number of the eigenvectors V with the largest eigenvalues λ of the diagonalized covariance matrix D to limit the dimensions. However, this neglects the importance of the query point itself. The difficulty with this is demonstrated by comparing Figures 3 and 4, where the query point determines whether the dimension can be neglected. Dimensional choice can be built as an extension of PCA in the following way. While PCA selects the eigenvectors with the largest variance λ, the query point can be included by selecting the dimensions with the largest value of the difference from the mean (q_i - μ_i) and the largest variance λ. We use a combination factor C to balance these two factors to give us a priority value P

4) $$P - Value = \lambda_i + C(q_i - \mu_i)$$

where i is the appropriate dimension, q is the query point, μ is the mean. Selecting the dimensions based on P-Value instead of λ gives the dimension prioritization a sensitivity to the query point.

Combining the UL-Distance and dimensional choice methods for nearest and farthest neighbor searches can provide significant improvement in speed. In order to determine the farthest neighbor in the UL-Distance, the dimensions of the query point are compared with the mean and variance of that dimension. Then the dimension which has the highest possible contribution is analyzed first to get a distance. The remaining dimensions are checked until the current distance difference cannot be exceeded because the potential contributions from the remaining dimensions are too small. An additional level of approximation can be included by estimating an earlier stopping point. This method operates in O(d log d + na)) where a is the number of dimensions that had to be analyzed at the worst case and is dominated by the initial sort, but can be reduced to O(d + n) ~ O(d) if only a partial sort is done initially. This compares favorably with the O(nd) of sequential scans.

In the case of the Euclidean or Manhattan distance metrics, the gains from adapting to the query point are not as profound as under the UL-distance because all of the dimensions have to be analyzed. However, we have demonstrated that the dimension of the distance function is what drives the difficulty in retrieval. This motivates the creation of new distance metrics like the UL-Distance that emulate Minkowski distance metrics but use a lower dimensionality.

Using Dimensional Choice differs from using PCA in several important ways. First, PCA uses the same dimensions for every classification distance, while Dimensional Choice is adaptive and use a different set of dimensions for each classification distance depending on the dimensions that are important for that particular point as well as those that are important for the data overall. Dimensional Choice works better with distance functions that are inherently lower dimensional like the UL-Distance and the chessboard distance functions because the combination of a limited number of dimensions and an effective choice of those dimensions complement each other.

2.3 Search using Similar Neighbor

Many nearest neighbor applications require the exact nearest neighbor. However, when looking for similarity, often the approximate nearest neighbor is sufficient. Judging whether approximate nearest neighbor is good enough requires an understanding of the underlying structure of similarity that is embedded into the space. The amount of approximation allowed depends on the tolerance of the system for mis-classification of points as similar. A better approach is finding a similar neighbor instead of the nearest neighbor. This avoids the discussion of how much approximation is tolerable by going directly to the question of similarity.

An example of success in similar neighbor would be finding a point that is not the nearest neighbor, but is similar to the query point. An example of failure would be finding any point that is not similar, even if it is the nearest neighbor. This measure of success is less strict in terms of actual distances to the objects that are retrieved but more strict in terms of the similarity of the objects to the query.

3 Experiments

The main questions for these techniques are what the speed improvement is, and what the change in accuracy is. In order to determine the change in accuracy, we look at a nearest neighbor application in recognizing prostate cancer. Here the loss in accuracy is judged by whether the classification loses accuracy, sensitivity, or specificity, instead of determining whether the particular nearest neighbor is exactly the same. This looser definition of accuracy is more of a functional definition as large high-dimensional data sets are increasingly used for classification. The accuracy will be compared with other nearest neighbor and SVM approaches.

We used a data set obtained from Clinical Proteomic Program Databank. The experimental data is a set of prostate cancer samples. The experiment analyzed serum proteomic mass spectra generated by SELDI-TOF to discriminate the sera of men with histopathologic diagnosis of prostate cancer (serum prostate-specific antigen [PSA] $\geq$ 4 ng/mL) from those men without prostate cancer (serum PSA < 1 ng/mL). In this data set, there are 63 normal (non-cancer) samples, and 69 cancer samples.

A SVM was used to compare the accuracy loss for the serum proteomic pattern analysis. A SVM is a blend of linear modeling and instance-based learning. A SVM selects a small number of critical boundary samples, called support vectors, from each category and builds a linear discriminate function that separates them as widely as possible. A kernel is used to automatically inject the training samples into a higher-dimensional space, and to learn a separator in that space [14]. In linearly separable cases, SVM constructs a hyper-plane, which separates the two different categories of feature vectors with a maximum margin, i.e., the distance between the separating hyper-plane and the nearest training vector. The training instances that lie closest to the hyper-plane are support vectors [14]. Linear and polynomial kernels were used. The feature selection method was MIT correlation, which is also known as signal-to-noise statistic [17].

The speed and accuracy improvement was measured on the same computer with the competing algorithms of the PCA nearest neighbor with a Euclidean distance metric versus the UL-Distance of order 2 with Dimensional Choice. The accuracy was also compared with two SVM approaches. Because of the limited supply of data, we used one sample as the test case and the remainder as the training cases and did this for each case. The drawback to this approach is that the result of each individual test is not independent of the results of the other tests.

4 Results

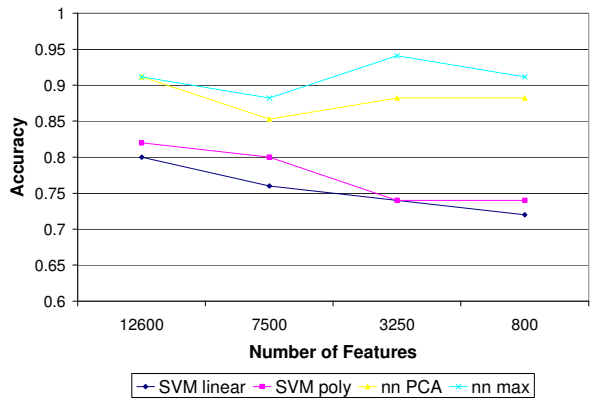

Figure 5: The accuracy of the methods versus the number of features (or dimensions). The nearest neighbor methods performed surprisingly well against the SVM. The nn PCA method is the nearest neighbor with PCA dimensions included, while the nn max uses the UL-Distance. The general flatness of the nearest neighbor methods is encouraging for using nearest neighbor with dimension reduction methods. The nn max does outperform all other methods. The general flatness of the nearest neighbor methods is encouraging for using nearest neighbor with dimension reduction methods.

The results were interesting overall as the nearest neighbor classification performed better overall than both of the SVM techniques. This is shown in Figures 5, 6, and 7. The use of the UL-Distance significantly increased the accuracy of the nearest-neighbors technique at low levels of features, as is shown in Figure 8, but performed at a similar level to the PCA choice of dimensions at high levels of features, as is shown in Figures 5, 6, and 7.

The accuracy of the classification is maintained with the reduction of the number of features from 12600 to 800 with the UL-Distance, while the PCA choice of dimensions shows slight degradation as is shown in Figure 5. This reduction in the dimensionality of the data by almost 70% without a loss of accuracy is encouraging. However, the accuracy is degraded below 100 dimensions as shown in Figure 8, but only by 6%

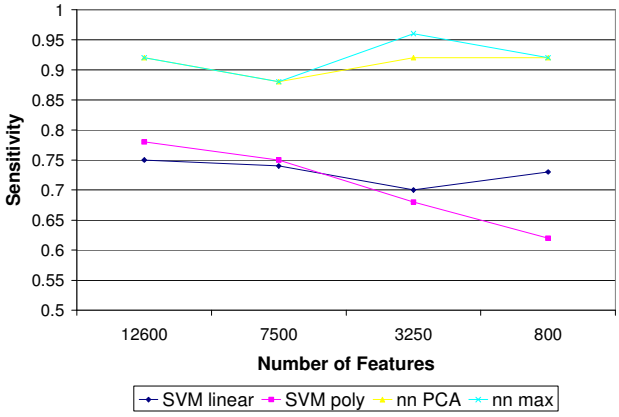

Figure 6: The sensitivity of the methods versus the number of features. The nn PCA method is the nearest neighbor with PCA dimensions included, while the nn max uses the UL-Distance. The similarity of the sensitivity of the two methods suggests that it is the specificity and not the sensitivity that makes the nn max the better technique. Both outperform SVM.

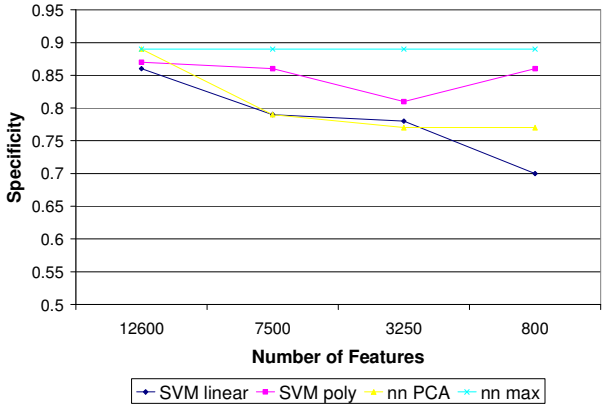

Figure 7: The specificity of the methods versus the number of features. The nn PCA method is the nearest neighbor with PCA dimensions included, while the nn max uses the UL-Distance. The flatness of the nn max method demonstrates that the dimension reduction does not adversely affect the specificity.

in order to achieve a dimension reduction of 99%. Note that using the UL-Distance instead of the PCA technique improved the accuracy by up to 12% at low numbers of dimensions, as is shown in Figure 8. Of course, these results are dependent on the data set used.

The specificity of the classification is surprisingly stable with dimension reduction under the UL-Distance, as is shown in Figure 7. The other methods did not fare as well. The sensitivity of the classification with the UL-Distance and nearest neighbor was surprisingly good, as is shown in Figure 6.

The improvement in the amount of time necessary to run the algorithms is shown in Figure 9. The cost savings is significant but only at the levels of dimension reduction that cause a loss of accuracy. The trade-off between accuracy and time savings is shown in Figure 10.

Farthest neighbor classification did not perform well. However, the farthest neighbor and nearest neighbor classifications did not tend to misclassify the same data points,

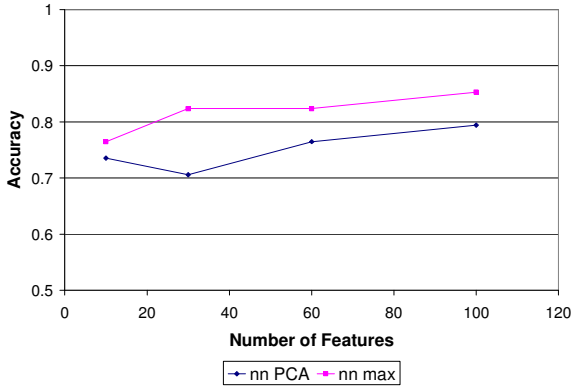

Figure 8: The accuracy of the nearest neighbor methods versus the number of features at a small number of features. The nn PCA method is the nearest neighbor with PCA dimensions included, while the nn max uses the UL-Distance. The loss in accuracy at an extremely small number of features is significant, but not terrible. The performance of the UL-Distance does improve the performance by 3-12% over the PCA features technique.

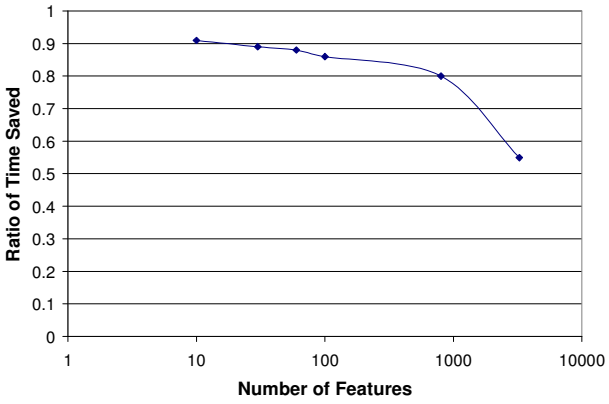

Figure 9: The ratio of the time saved accessing the nearest neighbor using Dimensional Choice versus the time necessary for the full nearest neighbor. The amount of time saved is significant up to 800 features, where the savings start to tail off.

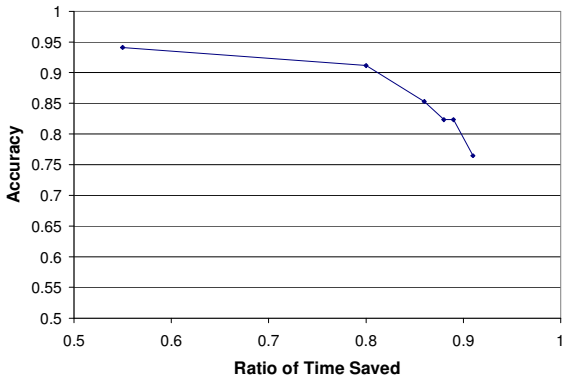

Figure 10: The accuracy of the nearest neighbor methods versus ratio of the time saved. An optimal point appears to be 800 features out of the original 12600 in order to minimize the lost accuracy while maximizing the savings in time.

which implies that the combination of the two might produce a better overall classifier.

5 Conclusion

This work has demonstrated significant dimension reduction, up to 70% reduction in the number of dimensions in the data set with no loss in accuracy or over 99% reduction with only a 6% loss in accuracy. The method can actually perform better with fewer dimensions than the nearest neighbor with all of the dimensions. The data set may be part of the reason, though it is a typical prostate cancer data set.

We have developed a new distance function called the UL-Distance that can be effectively used to replace Euclidean or other Minkowski metrics for high-dimensional nearest neighbor operations. This performed at up to 12% better than alternate approaches.

Combining this new distance function with a technique of Dimensional Choice where the best dimensions to analyze are guessed using information about the underlying data and the query itself in order to minimize the amount of work required to perform the nearest neighbor search with the UL-Distance achieved significant savings in work. The time to perform a nearest neighbor search is reduced by a factor of five with no loss of accuracy, but can be improved up to a factor of ten at some loss of accuracy, as is shown in Figure 10.

We demonstrate that the curse of dimensionality is not based on the dimension of the data itself, but primarily upon the effective dimension of the distance function. The effective dimension of the UL-Distance is set to a factor of L even though it can act on any of the possible d dimensions. We also note that the higher the order U of the UL-Distance function, the better the approximation performs since the small factors that would be included from neglected dimensions are effectively reduced when using a higher order distance function.

We note that this work is preliminary and does require more extensive analysis. However, the combination of a more effective ranking of dimensions using dimensional choice and a dimension-limiting distance function appear to be an effective combination when using high-dimensional data.

References

[1] C. C. Aggarwal. Towards systematic design of distance functions for data mining applications. In Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 9–18, Washington, D.C., August 2003.

[2] M. Artae, M. Jogan, and A. Leonardis, “Incremental PCA for On-Line Visual Learning and Recognition,” Proc. 16th Int’l Conf. Pattern Recognition, pp. 781-784, 2002.

[3] R. E. Bellman. Adaptive Control Processes. Princeton University Press, Princeton, NJ, 1961.

[4] S. Berchtold, C. B¨ohm, and H.-P. Kriegel. Improving the query performance of high-dimensional index structures by bulk-load operations. In Advances in Database Technology — EDBT’98, Proceedings of the 1st International Conference on Extending Database Technology, H.-J Schek, F. Saltor, I. Ramos, and G. Alonso, eds., pages 216–230, Valencia, Spain, March 1998.

[5] S. Berchtold, C. B¨ohm, H.-P. Kriegel, J. Sander, and H. V. Jagadish. Independent quantization: An index compression technique for high-dimensional data spaces. In Proceedings of the 16th IEEE International Conference on Data Engineering, pages 577–588, San Diego, CA, February 2000.

[6] A.L. Blum and P. Langley, “Selection of Relevant Features and Examples in Machine Learning,” Artificial Intelligence, vol. 97, nos. 1-2, pp. 245-271, 1997.

[7] C. Bohm, B. Braunmuller, F. Krebs, and H.-P. Kriegel. Epsilon grid order: an algorithm for the similarity join on massive high-dimensional data. In Proceedings of the ACM SIGMOD Conference, pages 379–390, Santa Barbara, CA, May 2001.

[8] J.-P. Dittrich and B. Seeger. GESS: a scalable similarity-join algorithm for mining large data sets in high dimensional spaces. In Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 47–56, San Francisco, California, August 2001.

[9] K. Daphne and M. Sahami, "Toward Optimal Feature Selection," Proc. 13th Int'l Conf. Machine Learning, pp. 284-292, 1996.

[10] W. Fan, M.D. Gordon, and P. Pathak, "Effective Profiling Of Consumer Information Retrieval Needs: A Unified Framework And Empirical Comparison," Decision Support Systems, vol. 40, pp. 213-233, 2004.

[11] H. Ferhatosmanoglu, E. Tuncel, D. Agrawal, and A. El Abbadi. Vector approximation based indexing for non-uniform high dimensional data sets. In Proceedings of the 9th International Conference on Information and Knowledge Management (CIKM), pages 202–209, McLean, VA, November 2000.

[12] K. Hiraoka, K. Hidai, M. Hamahira, H. Mizoguchi, T. Mishima, and S. Yoshizawa, "Successive Learning of Linear Discriminant Analysis: Sanger-Type Algorithm," Proc. 14th Int'l Conf. Pattern Recognition, pp. 2664-2667, 2000.

[13] R. Hoch, "Using IR Techniques for Text Classification in Document Analysis," Proc. 17th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval, pp. 31-40, 1994.

[14] T. Joachims, "Making large-scale support vector machine learning practical", International Conference on Machine Learning (ICML'99), 200-209 (1999).

[15] N. Gershenfeld. The Nature of Mathematical Modeling. Cambridge University Press, Cambridge, United Kingdom, 1999.

[16] A. Gionis, P. Indyk, and R. Motwani. "Similarity search in high dimensions via hashing". In Proceedings of the 25th International Conference on Very Large Data Bases (VLDB), M. P. Atkinson, M. E. Orlowska, P. Valduriez, S. B. Zdonik, and M. L. Brodie, eds., pages 518–529, Edinburgh, Scotland, September 1999.

[17] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, E. S Lander, Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring, Science, 286, 531-537 (1999).

[18] G. H. Golub and C. F. van Loan. Matrix Computations. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

[19] A. Hinneburg, C. C. Aggarwal, and D. A. Keim. What is the nearest neighbor in high dimensional spaces. In Proceedings of the 26th International Conference on Very Large Data Bases (VLDB), A. El Abbadi, M. L. Brodie, S. Chakravarthy, U. Dayal, N. Kamel, G. Schlageter, and K.-Y. Whang, eds., pages 506–515, Cairo, Egypt, September 2000.

[20] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In Proceedings of the 30th Annual ACM Symposium on the Theory of Computing, pages 604–613, Dallas, May 1998.

[21] D. Kalashnikov and S. Prabhakar. Similarity joins for low- and high- dimensional data. In Proceedings of the 8th International Conference on Database Systems for Advanced Applications (DASFAA'03), pages 7–16, Kyoto, Japan, March 2003.

[22] D.D. Lewis, "Feature Selection and Feature Extraction for Text Categorization," Proc. Workshop Speech and Natural Language, pp. 212-217, 1992.

[23] H. Li, T. Jiang, and K. Zhang, "Efficient and Robust Feature Extraction by Maximum Margin Criterion," Proc. Conf. Advances in Neural Information Processing Systems, pp. 97-104, 2004.

[24] Y. Li, L. Xu, J. Morphett, and R. Jacobs, "An Integrated Algorithm of Incremental and Robust PCA," Proc. Int'l Conf. Image Processing, pp. 245-248, 2003.

[25] S.A. Nene and S.K. Nayar. A Simple Algorithm for Nearest-Neighbor Search in High Dimensions. PAMI, 19(9):989–1003, September 1997.

[26] E. Oja, "Subspace Methods of Pattern Recognition," Pattern Recognition and Image Processing Series, vol. 6, 1983.

[27] R.O. Duda, P.E. Hart, and D.G Stork, Pattern Classification, second ed. John Wiley, 2001.

[28] A.R. Webb, Statistical Pattern Recognition, second ed. John Wiley, 2002.

[29] 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 24th International Conference on Very Large Data Bases (VLDB), A. Gupta, O. Shmueli, and J. Widom, eds., pages 194–205, New York, August 1998.

[30] J. Weng, Y. Zhang, and W.-S. Hwang, "Candid Covariance-Free Incremental Principal Component Analysis," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 25, pp. 1034-1040, 2003.

[31] J. Yan, B.Y. Zhang, S.C. Yan, Z. Chen, W.G. Fan, Q. Yang, W.Y. Ma, and Q.S. Cheng, "IMMC: Incremental Maximum, Marginal Criterion," Proc. 10th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, pp. 725-730, 2004.

[32] J. Yan, N. Liu, B.Y. Zhang, S.C. Yan, Q.S. Cheng, W.G. Fan, Z. Chen, W.S. Xi, and W.Y. Ma, "OCFS: Orthogonal Centroid Feature Selection," Proc. 28th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval, 2005.

[33] Y. Yang and J.O. Pedersen, "A Comparative Study on Feature Selection in Text Categorization," Proc. 14th Int'l Conf. Machine Learning, pp. 412-420, 1997.