

Efficient Computation of Statistical Significance of Query Results in Databases

Vishwakarma Singh¹, Arnab Bhattacharya², and Ambuj K. Singh¹
vsingh@cs.ucsb.edu arnabb@iitk.ac.in ambuj@cs.ucsb.edu

¹ Department of Computer Science, University of California, Santa Barbara, USA.

² Department of Computer Science and Engineering, Indian Institute of Technology (I.I.T.), Kanpur, India.

Abstract. Queries such as database similarity searches return results satisfying certain properties of distances or scores. For domain scientists, the absolute values of scores are seldom sufficient. Statistical significance or *p-value* of the result is a more useful criterion. This can be computed using an appropriate model of random objects. The problem of computing p-values becomes more acute when queries have multiple components. In this case, the returned score is an aggregate of individual scores. The simple way of calculating the p-value by enumerating all random possibilities fails for large database and query sizes. We propose an efficient method to calculate the approximate p-value of a multi-attribute result when the distribution of scores for the database objects is non-parametric. Experimental evaluation on large databases shows that our method is practical, runs 5 orders of magnitude faster than the basic approach, and has an error of less than 5% in p-value computation.

1 Motivation and Problem Statement

Many database systems retrieve results based on some distance or score measure between the query object and the database objects. Score is a quantitative measure of the similarity between objects based on multiple attributes. It has been widely used for ranking results in content-based multimedia retrieval systems. However, with the growing interest in analyzing the results of a database similarity query, computing rigorous statistical properties of the results is more meaningful.

Statistical significance helps the domain scientists in understanding the nature of the query and the statistical properties of the database objects. The most well known example is BLAST [1]. A standard measure of statistical significance is the *p-value*. The p-value of score s of a query result from a database is defined as the probability of randomly obtaining a result from the database with a score s or higher for the same query. It is the area under the probability distribution function (pdf) of the scores of random objects greater than s .

For a database management system (DBMS) serving single object queries, the score *pdf* can be characterized or calculated, and so, the p-value can be computed. However, there are database systems of complex objects where each object consist of multiple attributes or components. Such systems support queries with multiple attributes or objects and the score of a result is some aggregate function (e.g., sum) of the individual

```

Algorithm PRUNE
Input: Query  $Q = \cup_{i=1}^r Q_i$ , Score  $s$ , Database  $D$ , Number of bins  $b$ 
Output: P-value  $p$ 
1. for  $i = 1$  to  $r$ 
2.    $D_i := 1\text{-NN}(Q_i, D)$ 
3.    $h_i := \text{BinHistogram}(D_i, b)$ 
4. end for
/*  $\sigma_i$  is the sum pdf of bin histograms  $1, \dots, i$  */
5.  $\sigma_1 := h_1$ 
6. for  $i = 2$  to  $r$ 
7.    $B(\sigma_i) := s - \sum_{j=i+1}^r \max(h_j)$ 
8.    $B(h_i) := B(\sigma_i) - \max(\sigma_{i-1})$ 
9.    $B(\sigma_{i-1}) := B(\sigma_i) - \max(h_i)$ 
10.   $\sigma_i := \text{Convolute}(\text{all bins } \sigma_{i-1,j} \geq B(\sigma_{i-1}), \text{all bins } h_{i,k} \geq B(h_i))$ 
11. end for
12.  $p := \text{Sum of probabilities in all bins } \sigma_{r,j} \geq s$ 

```

Fig. 1. The PRUNE algorithm.

scores of each query component against its corresponding result component [5]. These queries are common for region based image retrieval (RBIR) systems [3] and information retrieval systems [9]. For example, in an RBIR system, a query region is composed of a number of sub-regions (e.g., tiles) [2, 10]. The database images are also split into sub-regions. Each component sub-region has a corresponding score of its match with a query sub-region. The score of a result is the sum of the individual scores.

For a given query object Q of size r , a random database for computing the p-value can be modeled by considering all possible aggregates of size r composed of components from the database. To find the p-value, we need to calculate the score pdf for this random database. This simple method has a running time that grows exponentially with database size and query size and is, therefore, impractical. In this paper, we propose and solve the following problem: “Given a query Q composed of r objects $Q_i, i = 1, \dots, r$, database objects $D_j, j = 1, \dots, n$, scoring functions $f_i : Q_i \times D \rightarrow \mathfrak{R}$, compute the p-value of obtaining a score s for a result $R = \cup_{i=1}^r R_i$, where $s = \sum_{i=1}^r f(Q_i, R_i)$, for a random database of objects, each having r component objects.”

Methods have been proposed for obtaining a single measure of statistical significance by combining the individual p-values. For example, the method in [4] requires finding the correlation among the attributes, which is done by sampling for large datasets. We adopt a more direct approach. We find the sum pdf of the individual pdfs of the components of the query. Then we calculate the p-value from this sum score pdf. Since score pdf of each component is independent of the other, this pdf is the *convolution* of all the individual pdfs. For most databases, the nature and the parameters of this pdf cannot be computed. We consider such cases where the probability distribution function of the cumulative scores is non-parametric.

2 Algorithm

For a multiple object query, the p-value can be found from the sum pdf of its components. The basic approach of calculating the sum pdf is to calculate the pdf of each query component and then find their convolution. Two score pdfs h_1 and h_2 can be convoluted to produce the sum score pdf h : the probability corresponding to score s considers all possible scores s_1 and s_2 from h_1 and h_2 such that $s = s_1 + s_2$. The cost of computing this convolution is, thus, *quadratic* in the number of distinct scores in the constituent pdfs. Hence, we can see that the convolution of multiple pdfs incurs a multiplicative cost on the size of the pdfs, and therefore, can be large. Assume that a query has r components, and each component has b distinct score values. The convolution of the first two components requires $b \times b = b^2$ operations and produces up to b^2 distinct scores. Convolving this result with the third component requires $b^2 \times b = b^3$ operations, and so on. The total running time, therefore, is $b \times b \times \dots \times b = O(b^r)$.

In order to speed up the p-value computation, we consider the two aspects of the problem—computing the score distribution for each query component and convoluting the distributions—separately. The first sub-problem is handled by pre-processing and maintaining a separate score pdf for each object component in the database. This can be done offline. For each component of the query, we approximate its score pdf by the pdf corresponding to its nearest component in the database. The nearest database component can be retrieved very efficiently by indexing the feature vectors of the objects using R-trees [7].

To efficiently convolute the pdfs and compute the p-value, we developed an approximation technique PRUNE (Fig. 1). There are three main steps in the algorithm: (i) Use *histograms* to approximate the score probability distribution functions of each query object, (ii) Progressively *cascade* the convolution of query object histograms to obtain the score histogram for the entire query, and (iii) Use *bounds* to convolute the histograms. We next explain each step in detail.

2.1 Use of Histograms to Approximate Distributions

Since the cost of convoluting two pdfs is a quadratic function of the number of distinct values in the pdfs, instead of using an actual score pdf, we approximate it by a histogram with a fixed number of bins as shown in step 3 of Algorithm PRUNE (Fig. 1). For speed, simplicity, and convenience, we choose equi-width histograms. The whole score range is divided into a fixed number of equi-width bins. The accuracy of the approximation depends on the number of bins maintained. More bins have less error, but higher running time. Section 3 considers the effect of the number of bins on the running time and the error in calculating the p-value.

2.2 Cascaded Convolution of Histograms

As described earlier, the simple way of directly convoluting r histograms has a time complexity which is exponential in r . To avoid such high costs, we convolute the histograms in a progressive fashion. Initially, the histograms of two query component objects are convoluted to yield another score histogram, which is again binned into b bins.

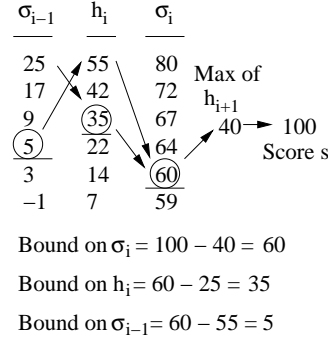


Fig. 2. Efficient convolution of histograms. $\sigma_{i-1} \oplus h_i = \sigma_i$. The bins below the score thresholds (shown inside circles) can be pruned to save time.

Then, this histogram is convoluted with the next histogram and so on till all the r histograms have been convoluted.

Denoting the i^{th} histogram by h_i and the convolution of histograms up to i components by σ_i , we compute $\sigma_i = \sigma_{i-1} \oplus h_i$ up to $i = r$. Each histogram convolution requires *quadratic* number of operations in terms of the number of bins in the histograms. The total time complexity, therefore, is $O(b^2r)$. To make it even more efficient, we apply a bounding procedure which is described next.

2.3 Convolution of Bounded Histograms

The bounding method is based on the observation that computing the p-value for a score s requires counting only those scores that are greater than or equal to s . Scores in the histogram of a query object that cannot add up to s even when combined with the best scores of the histograms of other query objects need not be considered. Therefore, the bins in the histogram whose scores fall below this *threshold score* can be deleted. The bounding method achieves this pruning of histogram bins by evaluating the threshold score at each stage. This reduces the number of bins, and thus, the running time.

Fig. 2 shows an example of how such thresholds are computed. Assume that the histogram σ_{i-1} is convoluted with h_i to yield σ_i . Also, assume that the score s for which the p-value is being calculated is 100. If the maximum score in h_{i+1} is 40, then any score below $100 - 40 = 60$ in σ_i cannot add up to s . This is the threshold score for that histogram, and is highlighted in the figure. Thus, all scores below 60 can be deleted from σ_i . By analyzing this bounding behavior backwards for the histograms σ_{i-1} and h_i , it can be seen that such contributing pairs of scores need not be calculated at all. The maximum score in h_i is 55. Since we do not need any score in σ_i that is below 60, all scores below $60 - 55 = 5$ in σ_{i-1} , when added to any score in h_i will be less than 60, and hence, can be deleted. Continuing this reasoning, all scores below 35 in h_i can be deleted. The threshold scores are highlighted in the figure.

In this example, the number of bins in σ_{i-1} and h_i are reduced from 6 to 4 and 3 respectively. This translates to a saving of $6 \times 6 - 4 \times 3 = 24$ bin convolution operations. Steps 7 to 10 of Algorithm PRUNE (Fig. 1) apply bounding to the cascaded convolu-

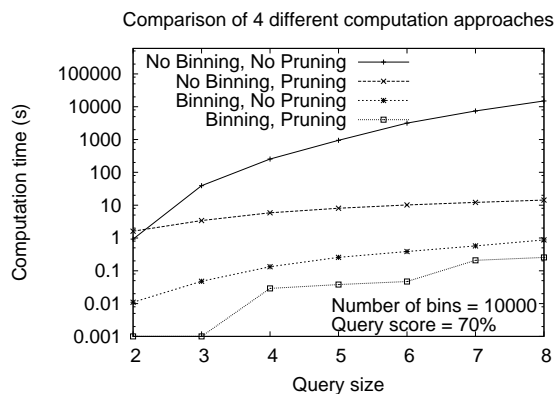


Fig. 3. Comparison of the various approaches of p-value computation.

tion. As shown in the next section, the overall saving for r histogram convolutions is significant.

Note that the two sources of error in the p-value computation are the use of nearest neighbors and histogram binning. The bounding method does not introduce any error.

3 Experiments

In this section, we demonstrate the effectiveness of our PRUNE method over alternate approaches. We explain the empirical results in the context of region-based image retrieval (RBIR) system for a biomedical image database of fluorescent micrographs of feline retinas labeled with different antibodies [6]. The dataset consists of 805,272 tiles. The score between two tiles is a decreasing function of the L_1 distance between the color histogram features of the tiles. The tiles are the component objects in our system. The score of the alignment of a query region to a database region is the sum of the scores of the alignment of the individual tiles. The details of the dataset preparation, the features, the scoring function, and the retrieval system are explained in [10].

3.1 Running Time

The basic approach of online computation of score pdfs of each query tile and their convolution yields impractical time. Therefore, we do not consider it. Instead, we maintain a database of the pre-computed pdf of each database component. We use the following parameters for the analysis of running time: (i) the number of bins in the score histograms, (ii) the query score for which the p-value is computed, measured as a percentage of the maximum score that can be achieved by the query, and (iii) the query size, which is the number of tiles in the query image.

First, we compare the four different approaches of computing the p-value: (i) Using actual pdf without pruning, (ii) Using actual pdf with pruning, (iii) Using binned pdf without pruning, and (iv) Using binned pdf with pruning (PRUNE). Fig. 3 shows their running times for different query sizes. The pruning strategy shows a gain of about

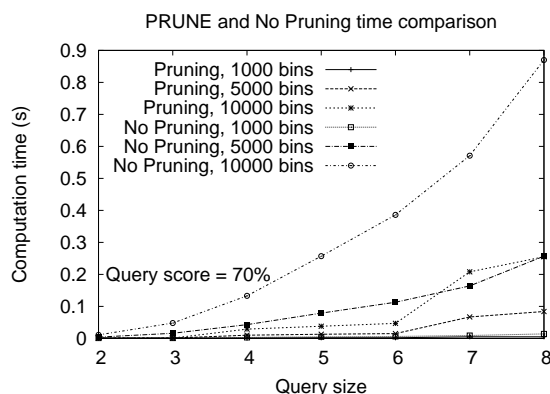


Fig. 4. The effect of pruning on the running time of p-value computation.

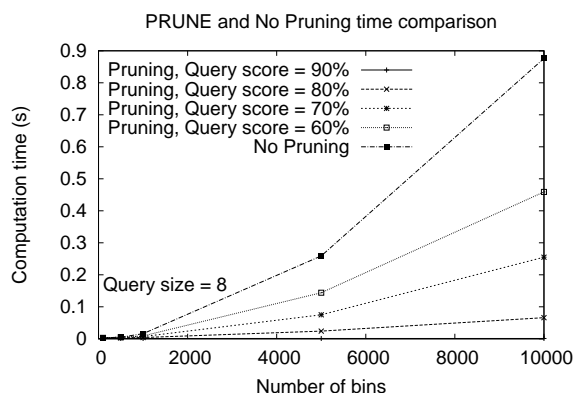


Fig. 5. The effect of query score and number of bins on the running time of p-value computation.

10^3 for a query size of 8 without binning. Binning improves the computation time by 2 orders of magnitude with pruning and 5 orders of magnitude without pruning. In all cases, the PRUNE strategy finished in practical times—at most 255 ms.

Since the PRUNE strategy outperforms all other approaches we analyze it further with respect to other parameters. Fig. 4 shows that the efficiency of pruning increases with the increase in query size across varying number of bins in the histogram. Up to medium query sizes of 6, and number of bins 5000, the scalability is linear or better.

The next experiment (Fig. 5) shows that the pruning strategy performs better when the query score increases, across varying number of bins. When the query score is 80% of the maximum score, the pruning strategy is very effective for all histogram bin sizes. The scalability is better for higher query scores. Thus, the empirical results strongly suggest that our PRUNE method is efficient and practical.

3.2 Error

We next performed experiments to measure the error in p-value computation induced by binning. Fig. 6 shows the error percentage across varying number of bins. When the

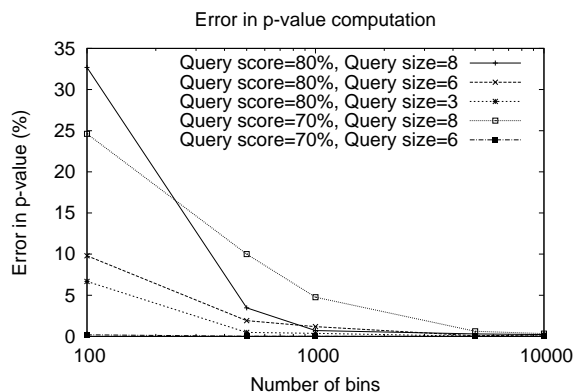


Fig. 6. The percentage error in p-value computation due to binning.

query size is large, using less number of bins accumulates the error over more number of steps, resulting in more than 20% error. Increasing the number of bins to 1000 reduces the error to at most 5%, irrespective of the query size and the query score. This proves the effectiveness of our strategy.

4 Conclusion

In this paper, we defined the problem of efficiently computing the p-value for multi-object query results for non-parametric distributions. We proposed an approximate bounding procedure PRUNE and showed that it is faster than the alternate approaches by more than 5 orders of magnitude with the error in computation less than 5%. Possible future avenues of work include sampling to obtain the score histograms, computing bounds for other aggregate functions like max, and examining the order in which the component object histograms should be convoluted in order to minimize the number of operations.

We presented a $O(b^2r)$ complexity algorithm for convoluting r histograms of b bins each. In future, we plan to examine the computation time and the possibility of optimization by utilizing the convolution theorem which states that the discrete Fourier transform (DFT) of the convolution of two equi-width histograms is equal to the product of the individual DFTs of the histograms [8]. This implies that the convolution of r histograms can be achieved in $O(rb \lg b)$ time using fast Fourier transform and its inverse [8].

References

1. S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic Local Alignment Search Tool. *J. Molecular Biology*, 215(3):403–410, 1990.
2. C. Dagli and T. S. Huang. A Framework for Grid-Based Image Retrieval. In *Proc. 17th Int. Conf. on Pattern Recognition (ICPR '04)*, volume 2, pages 1021–1024, 2004.
3. R. Datta, J. Li, and J. Z. Wang. Content-Based Image Retrieval: Approaches and Trends of the New Age. In *MIR '05: Proc. 7th ACM SIGMM Int. Workshop on Multimedia Information Retrieval*, pages 253–262, 2005.

4. R. DeLongchamp, T. Lee, and C. Velasco. A Method for Computing the Overall Statistical Significance of a Treatment Effect Among a Group of Genes. *BMC Bioinformatics*, 2006.
5. R. Fagin, A. Lotem, and M. Naor. Optimal Aggregation Algorithms for Middleware. *J. Computer and System Sciences*, 66(4):614–656, 2003.
6. S. K. Fisher, G. P. Lewis, K. A. Linberg, and M. R. Verardo. Cellular Remodeling in Mammalian Retina: Results from Studies of Experimental Retinal Detachment. *Progress in Retinal and Eye Research*, 24(3):395–431, 2005.
7. A. Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. In *Special Interest Group on Management of Data (SIGMOD)*, pages 47–57, 1984.
8. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, 1992.
9. G. Salton. *Automatic Text Processing: The Transformation Analysis and Retrieval of Information by Computer*. Addison-Wesley, 1988.
10. V. Singh, A. Bhattacharya, A. K. Singh, C. Banna, G. P. Lewis, and S. K. Fisher. QUIP: Querying Significant Patterns from Image Databases. Technical report, Dept. of Computer Science, University of California, Santa Barbara, 2007.