Discovering Strong Skyline Points in High Dimensional Spaces

Zhenjie Zhang Xinyu Guo Hua Lu Anthony K.H. Tung Nan Wang
School of Computing
National University of Singapore, Singapore
{zhangzh2,guoxy,luhua,atung,wangnan}@comp.nus.edu.sg

ABSTRACT

Current interests in skyline computation arise due to their relation to preference queries. Since it is guaraneed that a skyline point will not lose out in all dimensions when compared to any other point in the data set, this means that for each skyline point, there exists a set of weight assignments to the dimensions such that the point will become the top user preference.

We believe that the usefulness of skyline points is not limited to such application and can be extended to data analysis and knowledge discovery as well. However, since the skyline of high dimensional datasets (which are common in data analysis applications) can contain too many points, various means must be developed to filter off the less interesting skyline points in high dimensions. In this paper, we will propose algorithms to find a set of interesting skyline points called **strong skyline points**. Extensive experiments show that our proposal is both effective and efficient.

Categories and Subject Descriptors

H.2.8 [**DATABASE MANAGEMENT**]: Database Applications—*Data Mining*

General Terms

Algorithms

Keywords

Skyline, High Dimensional Space

1. INTRODUCTION

Given a n-dimensional space which has an order (or partial order) associated with each dimension, a point x is said to dominate a point y if x is ranked higher than y in all dimensions. A skyline point from a n-dimensional dataset D refers to a point x which is not dominated by any other point from D. We call the set of such points in D, the **skyline** of D, whose calculation has been studied extensively [4, 6]. Current interests in skyline points arise due to their relation to preference queries [3, 5, 7, 8, 9, 10]. Since it is guaranteed that a skyline point will not lose out in all dimensions when compared to any other point in D, this means that for each skyline point, there exists a set of weight assignments to the dimensions such that the point will become the top user preference [3].

While this analysis can be conducted on low dimensional datasets, doing this for high dimensional datasets (which are common in data analysis applications) remains a challenge. This is because the skyline of high dimensional datasets can contain too many points since it becomes increasingly difficult for one point to dominate another as the dimensionality of the dataset increases.

To overcome this problem, we introduce some new concepts in this paper. Given a space S, a subspace is said to be a **δ-subspace** if its skyline contains less than δ points. The union of the skyline points in all δ-subspace of S are called **strong skyline points**. Due to such constraint on the skyline point, the number of strong skyline points is much less than the original skyline points.

Based on the property of δ-subspace which is very similar to *Apriori* property of frequent itemset [2], we propose two subspace searching algorithms with breadth first strategy and depth first strategy respectively, to find those δ-subspaces. We also improve the existing *BNL* algorithm [3] to efficiently determine whether a given subspace is δ-subspace by exploiting the properties of strong skyline points.

The threshold δ is the most important parameter in our methods. Setting it to too low a value will remove too many skyline points while setting it to too high will have the opposing effect. In this paper, we provide a guidance for setting δ by extending the analysis from [4] to correlated subspaces and provide an upper bound on setting δ.

2. ALGORITHMS

It is not difficult to show that the subspaces of a δ-subspace must also be δ-subspaces, which is very similar to the *Apriori* property of frequent itemsets in association rule mining. This gives us the intuition to use the searching strategy of frequent itemset mining in strong skyline problem.

Breadth first search (BFS) [2] and depth first search (DFS) [1] are two basic subspace searching algorithms in frequent itemset mining. The former method finds all the subspaces with less dimensions before moving onto subspaces with more dimensions, while the latter method extends the subspaces according to the subspace lexicographical order.

CIKM'05, October 31–November 5, 2005, Bremen, Germany.
ACM 1-59593-140-6/05/0010.

To use BFS in our problem, we split all the subspaces into levels. Level i ($1 \leq i \leq n$), L_i, consists of all the δ-subspaces with i dimensions. The search process is conducted from low level to high level, since the necessary condition of a δ-subspace is that all of its subspaces are δ-subspaces. We can find the candidate subspaces on L_k by joining every pair of subspaces on L_{k-1} if the union of their dimensions forms a new subspace on level k.

At the beginning, every dimension of S is checked and those dimensions with more than δ points ordered on the top of the dimension are pruned. Then, to extend from level k to level $k+1$, candidate subspaces are generated by joining pairs of subspaces in level k if two subspaces share the first $k-1$ dimensions, which is called *Prefix Property*.

To use DFS in our problem, we only need to find all the max δ-subspaces, which is not subspace of any other δ-subspace, instead of searching skylines in all δ-subspaces. On the other hand, if a subspace Ψ is the subset of a max δ-subspace Φ, we can assert Ψ must be a δ-subspace without necessity to find the exact skyline in Ψ. Intuitively, a depth first search strategy may save the time by reducing the number of skyline searches without missing any δ-subspace.

Before the start of the depth first search algorithm, we first construct an empty stack St for δ-subspaces and an empty list Lt of max δ-subspaces. Then, we find all the one dimension δ-subspaces like what is done in the BFS algorithm. All the 1-dimension δ-subspaces are pushed into the stack St in the reverse order of the dimensions. Every round, we pop out one subspace Ψ from St and try to extend to new subspaces by adding new dimension at the tail of the dimension list. If the dimensions of the candidate subspace are covered by some max subspace in Lt, it is directly pushed into the stack with the skylines in Ψ. If no max subspace can cover the candidate, the skyline search algorithm is invoked to test whether it is a δ-subspace. If no extension from a subspace is successful and no max subspace can cover Ψ, Ψ is a max δ-subspace and it is inserted into Lt.

To further speed up the algorithms, we also propose an improved BNL algorithm by exploiting the properties of strong skyline points. When we try to merge two δ-subspaces Ψ and Φ with prefix property, since all skyline points in Ψ and Φ must be in the skyline of Ω, we can first calculate the union of the skyline sets of Ψ and Φ, if the size of the union exceeds the threshold, we can discard Ω immediately. The complexity of this process is $O(\delta)$, which we can usually omit since δ is generally much smaller than the data set size. We can also abandon the calculation of the skyline in a subspace immediately if we have already found $\delta + 1$ skyline points.

3. PARAMETER SETTING

We assume that $SP(D, S)$ is the number of skyline points in data set D in space S, where D is a data set with N points.

Theorem 1: If a new independent dimension d is added into a correlated subspace Φ, $\Pr(|SP(D, \Phi \cup \{d\})| \leq \delta) \leq \ln N/(\ln N - \delta)^2$

From Theorem 1, we can see that the fewer skyline points are in subspace $\Phi \cup \{d\}$, the more likely d is independent of the correlated subspace Φ.

Theorem 1 also tells us that with specified confidence requirement, a proper threshold δ can be calculated. This turns out to be a useful tool to determine the parameter δ according to the requirement in practice.

4. EXPERIMENT

We use the NBA all-time players statistics data as our real data for test. The NBA all-time players statistics is a 17 dimension dataset with 16644 records. It contains all players' season statistics since NBA's first season in 1946. There are only 13 strong skyline points if $\delta = 2$, all of which are legend superstars in NBA's history. This shows that our definitions do find some interesting points in the data set.

When the threshold δ is 5, we find that Michael Jordan is a strong skyline point in the subspace with positive "Points", negative "Defensive Rebounds" and positive "Total Rebounds" attributes. This result shows that Jordan belongs to the group of very few players who can get a lot of points and total rebounds but has very few defensive rebounds, which implies that Jordan concentrated on attacking more than defending in the games.

In the synthetic data set tests, BFS algorithm outperforms DFS algorithm in almost all the cases. Although DFS invokes much less times of the skyline searches on δ-subspaces, it invokes the skyline searches on those false extension subspaces much more frequently. This phenomenon is due to the fact that DFS often underestimates the number of strong skyline points in most of the δ-subspaces when they are covered by a max δ-subspace, which reduces the pruning ability of the union of skylines in subspaces.

5. CONCLUSION

In this paper, we introduce the strong skyline point to find those special points of a large data set in high dimensional spaces. Strong skyline points are skyline points in subspaces which have very few skyline points. Compared with conventional skyline definition, the new concept removes most of the less meaningful points. To efficiently find those strong skyline points and their corresponding correlated subspaces, we present bottom-up subspace search algorithm and improve the efficiency of skyline point calculation for a certain subspace. A good bound on the threshold for the subspaces in probability is also derived to help setting the parameter on the theoretical basis. We show that strong skyline point can not only find special points in a large data set but also facilitate some new data analysis methods and knowledge discovery. The experiments indicates that our current algorithms can work well on different types of data sets.

6. REFERENCES

[1] R. Agarwal, C. Aggarwal, and V. Prasad. Depth First Generation of Long Patterns. *SIGKDD*, 2000.

[2] R. Agrawal, and R. Srikant. Fast algorithms for mining association rules. *VLDB*, 1994.

[3] S. Borzonyi, D. Kossmann, and K. Stocker. The skyline operator. *ICDE*, 2001.

[4] J. L. Bentley and H. T. Kung and M. Schkolnick, and C. D. Thompson. On the Average Number of Maxima in a Set of Vectors and Applications. *Journal of ACM*, 25(4), 1978.

[5] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang. Skyline with presorting. *ICDE*, 2003.

[6] H. T. Kung, F. Luccio, and F. P. Preparata. On finding the maxima of a set of vectors. *Journal of ACM*, 22(4), 1975.

[7] D. Kossmann, F. Ramsak, and S. Rost. Shooting stars in the sky: An online algorithm for skyline queries. *VLDB*, 2002.

[8] D. Papadias, Y. Tao, G. Fu, and B. Seeger. An optimal and progressive algorithm for skyline queries. *SIGMOD*, 2003.

[9] D. Papadias, Y. Tao, G. Fu, and B. Seeger. Progressive skyline computation in database systems. *ACM TODS*, 30(1):41–82, 2005.

[10] K. Tan, P. Eng, and B. Ooi. Efficient progressive skyline computation. *VLDB*, 2001.