Densest low-rank subgraph

Vasily and Evimaria

May 2020

1 Introduction

1.1 Definitions and Notation

Definition 1.1. Let $G = (V, E)$ be a graph. For $S \subset V$ we define $E(S)$ to be the set of edges induced by S. We will call $f(S) = \frac{|E(S)|}{|S|}$ the density of S.

Definition 1.2. Let $S \subset V \subset \mathbb{R}^m$ and $k \geq 1$ an integer. We define A_S to be the matrix with columns given by S, and $e_k(S) = ||A_S - A_{S,k}||$ to be the error of S, where $A_{S,k}$ is the k-rank approximation of A_S, which can be computed using the SVD. If $E \subset \mathbb{R}^m$ is the set of (left) principal vectors of A_k, and $e_k(S) \leq M$ for some M, we say that E $M-$spans S. Note that if $M = 0$, then E simply spans S.

1.2 Useful facts about SVD

Let A be a $n \times m$ matrix of rank r with rows being points in $\mathbb{R}^m$. The SVD of A is

$$\sum_{i \leq r} \sigma_i u_i v_i^T = U \Sigma V^T.$$

Vectors v_i are the (right) singular vectors of A with the property

$$v_i = \arg \max_{||v_i||=1, v_i \perp v_1 \ldots v_{i-1}} ||Av_i||,$$

where $||Av_i|| = \sigma_i$ is sum of squares of lengths of projections of points onto v_i.

The relationship between the Frobenius norm and the singular values is

$$||A||^2 = \sum_{i,j} a_{i,j}^2 = \sum \sigma_i^2.$$

Given $A_k = \sum_{i \le k} \sigma_i u_i v_i^T$, the best k-rank approximation of A in both the 2-norm and the Frobenius norm,

$$||A - A_k|| = ||\sum_i \sigma_i u_i v_i^T - \sum_{i \le k} \sigma_i u_i v_i^T|| = ||\sum_{i>k} \sigma_i u_i v_i^T|| = \sum_{i>k} \sigma_i^2.$$

Thus,

$$e_k(S) = \sum_{i>k} \sigma_i^2 = \text{ sum of squared lengths of projections onto the first } k \text{ singular vectors,}$$

where σ_i are the singular values of S viewed as a matrix. This gives an algebraic and a geometric way to interpret the error.

1.3 Questions

For the questions below, $k \ge 1$ is an integer and $G = (V, E)$ is a graph with $V \subset \mathbb{R}^m$.

Question 1.3 (Main question)**.** Find a subset of vertices $S \subset V$ that maximizes $\lambda f(S) - e_k(S)$.

Question 1.4 (Main question 2)**.** Given an integer $n \ge 1$, find a subset of vertices $S \subset V$ of size n that maximizes $\lambda f(S) - e_k(S)$.

Question 1.5 (Forgetting about density)**.** Find subset of vertices $S \subset V$ of size n that minimizes $e_k(S)$. The binary version of the problem is: "*Given $M \ge 0$ and integers $k, n \ge 1$, does there exists $S \subset V$ of size n such that $e_k(S) \le M$?*" Is this problem **NP**-hard?

Answer. Case $M = 0$ is not **NP**-hard for any given k, by algorithm 1, but the runnning time is exponential in k. Also, if the dimension m is fixed, the problem is no **NP**-hard by 4.2. □

Question 1.6. Suppose the data V is normalized in some way. When will the SVD of V give good info about the problem?

2 Ideas

2.1 Useful examples

Let $k = 1$, $M = 0$ and $\mathbb{R}^m = \mathbb{R}^3$. Further, let V consist of two groups V_1 and V_2, where $|V_2| >> |V_1| = n$ (for example, $|V_2| = 2^n$). The vectors in V_1 are all equal to $e_1 = (1, 0, 0)$. The vectors in V_2 are equal to $e_2 = (0, 1, 0)$ with some small noise in the third component, such that no two vectors in V_2 are collinear. Thus, the first principal component of V is (approximately) e_2, and V_1 is the set of outliers. But set V_1 is the only set that has zero-error low-rank approximation $e_k(V_1)$.

Here is a related example. Let $P >> 1$ be large parameter and $\epsilon << 1$ a small parameter, and $|V_1| = |V_2| = n$, with V_1 as above, and V_2 consisting of two groups: V_2^+ and V_2^-. Vectors in V_2^+ are all equal to $P \cdot (0, 1, \epsilon)$, and vectors in V_2^- are all equal to $P \cdot (0, 1, -\epsilon)$. The first principal component of $V = V_1 \cup V_2$ is (approximately) e_2. But $e_1(V_2) \approx \sum_{i \le n} P^2\epsilon^2 = nP^2\epsilon^2$. By letting $\epsilon = 1/\sqrt{P}$ and $P \to \infty$, we get arbitrarily large error $e_1(V_2)$, while $e_1(V_1) = 0$.

2.2 Connection with MCD

The MCD problem from [1] is related to our problem in the following algebraic way. For a given set S and the matrix it spans (call it S too), its SVD is given by $S = U\Sigma V^T$, where $SS^T = U\Sigma^2 U^T$ is the covariance matrix. Thus, the objective function in MCD is

$$\det(S) = \det\left(SS^T\right) = \det\left(U\Sigma^2 U^T\right) = \det\left(\Sigma^2\right) = \prod_i \sigma_i^2 = \text{ product of singular values squared.}$$

The objective function to be minimized in our problem is

$$e_k(S) = \sum_{s \in S} ||s||^2 - \sum_{i \le k} \sigma_i^2.$$

We note that if $||s|| = 1$ for all points s, then the objective in our problem is to maximize the sum of principal components $\sum_{i \le k} \sigma_i^2$.

2.3 Randomized

Given set $V \subset \mathbb{R}^m$, consider the following algorithm. Initialize $S = \{v_1\}$ for a random $v_1 \in V$. For $i = 2, ..., k$: randomly sample v_i from V, proportional to $\sum_{v \in S} \langle v_i, v \rangle$, and let $S = S \cup \{v_i\}$.

2.4 Iterative algorithm (similar to [4])

Initialize a bunch (α) of subsets $S \subset V$. Each S is initialized as follows: first, select k vectors from V u.i.r., and then find $n - k$ points in V that are the closest to the span of the k initial vectors. Now, for each set S, compute the set $E = \{e_1, \ldots, e_k\}$ of the first k singular vectors, and choose n points from V that are the closest to the span of E. Repeat until one of the α trajectories converges. Notice that eventual convergence is guaranteed because at each step the error $e_k(S)$ does not increase.

2.5 Divide and Conquer algorithm

Let k be a power of 2 and $|V|$ be a power of 2, and suppose there exists an algorithm (may be approximate) that, given $V \subset \mathbb{R}^m$ will find subset $S \subset V$ of a given size that minimizes $e_1(S)$, i.e. rank-1 approximation. Then can randomly split V into V_1 and V_2 of equal size, (recursively) find subsets S_1, S_2 of V_1 and V_2 that have low-rank approximation, and let $S = S_1 \cup S_2$.

2.6 SVD as an optimization problem

SVD is a minimization problem: for $k = 1$, given n vectors $x_1, ..., x_n \in \mathbb{R}^d$, find vector e_1 that minimizes the sum (distance from point i to $e_1)^2$, which is equivalent to maximizing the sum (length of projection$)^2$.

If the set of vectors $x_1, ..., x_n$ changes slightly (e.g. one vector is swapped out), then e_1 will also not change much (e_1 is continuous as a function of $x_1, ..., x_n$). Can do a few steps of gradient descent to find the right e_1.

3 Resources

1. overview of matrix norms
2. Evimaria's paper on approximating (submodular - linear) function
3. Greedy for dense subgraphs
4. incremental SVD (incSVD and EincSVD and AEincSVD)
5. incremental SVD (recover SVD from SVD's of blocks)
6. Wikipedia has a surprisingly good article on PCA
7. hardness results
8. amazing SVD overview with geometric intuition
9. Petros Drineas' paper on CUR relative error. Section 4 concerns the approximation error guarantees. Their algorithm is probabilistic but perhaps we can use the mere existence of a good CUR decomposition to our advantage.
10. Existence of rows with good approximation guarantee. Theorem 1.4 gives the existence.

References

[1] Thorsten Bernholt and Paul Fischer. "The complexity of computing the MCD-estimator". In: *Theoretical Computer Science* 326.1-3 (2004), pp. 383–398.

[2] Sanjay Chawla and Aristides Gionis. "k-means–: A unified approach to clustering and outlier detection". In: *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM. 2013, pp. 189–197.

[3] Luis Rademacher, Santosh Vempala, and Grant Wang. "Matrix Approximation and Projective Clustering via Iterative Sampling". In: (Dec. 2005).

[4] Peter J Rousseeuw and Katrien Van Driessen. "A fast algorithm for the minimum covariance determinant estimator". In: *Technometrics* 41.3 (1999), pp. 212–223.

4 Appendix

4.1 naive algorithms

Answer to 1.5. Consider the following algorithm that returns S if exists S with $e_k(S) = 0$, and “false” otherwise:

Algorithm 1

1: **procedure** FINDS(V,n,k)
2: **for** each set $\{e_1, ..., e_k\} \subset V$ of size k **do**:
3:   $M = \begin{pmatrix} e_1, ..., e_k \end{pmatrix}$
4:   $S = E$
5:   **for** $v \in V \setminus \{e_1, ..., e_k\}$ **do**
6:    **if** $Mx = v$ has a solution **then**
7:     $S = S \cup \{v\}$
8:   **if** $|S| \geq n$ **then**
9:    **return** S
10: **return** false

Note that line 6 checks if $v \in \operatorname{span}(\{e_1, ..., e_k\})$, and takes time $\Theta(m^2k)$ or $\Theta(mk)$ if a (LUP or any other) factorization of M is precomputed. The overall complexity of algorithm 1 is $O(\binom{N}{k}Nkm)$ (by precomputing a factorization of M to make solving $Mx = v$ faster), where $N = |V|$. For small $k << N$ and $m = O((k-1)!)$ the complexity is $O(N^{k+1})$.

It is not immediately clear that the algorithm is correct. For a set $S \subset V$ $e_k(S) = 0$ is equivalent to S being spanned by k vectors. If such S does not exist, the algorithm will not find one. If such S exists, there are k vectors that span S. So $d := \dim \operatorname{span}(S) \leq k$. So there exist d vectors in S that span S. Thus the k vectors can be chosen from S, and the algorithm will find them by checking all subsets of V of size k.

Now consider the following modification of algorithm 1, which attempts to find S that minimizes $e_k(S)$.

Algorithm 2

1: **procedure** FINDS(V,n,k)

2: **for** each set $E = \{e_1, ..., e_k\} \subset V$ of size k **do**:

3: $M = \Big(e_1, ..., e_k\Big)$

4: $D =$ empty array

5: **for** $v \in V \setminus \{e_1, ..., e_k\}$ **do**

6: $x =$ solution to $M^T M x = M^T v$ (so $x = \text{proj}_{\text{span}(E)}(v)$)

7: append $||v - Mx||^2$ to D

8: $S_E = E \cup$ the $n - k$ smallest values of D

9: $e_E =$ sum of $n - k$ smallest values of D

10: **return** S_E with smallest e_E.

Algorithm 2 is a simple modification of algorithm 1. On line 6, instead of solving the linear system $Mx = v$ exactly, we find the closest point x in the span of $\{e_1, ..., e_k\}$ (which is the projection of v onto $\text{span}\left(\{e_1, ..., e_k\}\right)$), and record the square of distance from x to v.

Note that this algorithm is not exact, as opposed to algorithm 1. This is because there might exist a set $\{e_1, ..., e_k\}$ that $M-$spans (has error $\leq M$) a set S, with $\{e_1, ..., e_k\}$ not being a subset of V. To be explicit, consider $V = \{(-1, 2), (1, 2)\}$, $n = 2$, $k = 1$, $M = 3$. Then there exists vector $e_1 = (0, 1)$, for which the error is 2, while if e_k were to be picked from V, the error would be 3.2. By [3] this algorithm is a 4-approximation if $k = 1$ and all vectors in V are of unit length.

Clearly, the runtime of algorithm 2 is the same as algorithm 1, and is equal to $O(N^{k+1})$, where $N = |V|$. $\square$

4.2 Our problem is in P

[2] notes that a set solving k-means– for $k = 1$ is selectable by a sphere. Similarly, a set solving out problem is selectable by a cylinder for $k = 1$. For an arbitrary k, a set solving our problem is selectable by a the quadric that describes the points r away from some subspace E.

To be precise, let $e_1, \ldots, e_k$ be an orthonormal basis for the subset E that solves out problem. Then set S that solves our problem consists of n points closest to E. Then exists quadric Q that selects S, i.e. exist coefficients $a_i, a_{i,j}$ for $1 \leq i \leq j \leq m$ s.t. $A_{i,j} = a_{i,j}$ is a symmetric matrix, and $b_i = a_i$ is a column vector, and for all points in S, $x^T A x + x^T b + a_0 \leq 0$ and for all points not in S, $x^T A x + x^T b + a_0 > 0$.

Let's compute this quadric. Let $r > 0$ be the cut-off distance, i.e. n closest points lie within r of E, and all other points lie further than r from E. Then a quadric selecting S is given by $||x - \text{proj}_E(x)||_2^2 - r^2 = 0$. But

$||x - \text{proj}_E(x)||_2^2 = ||x||_2^2 - ||\text{proj}_E(x)||_2^2$. Note that $||x||_2^2 = \sum_{i=1}^d x_i^2$ and

$$
\begin{aligned}
||\text{proj}_E(x)||_2^2 &= \sum_{\ell=1}^k \langle x, e_\ell \rangle^2 = \sum_{\ell=1}^k \left(\sum_{i=1}^m x_i e_{\ell,i} \right)^2 = \sum_{\ell=1}^k \left(\sum_{i=1}^m x_i^2 e_{\ell,i}^2 + 2 \sum_{1 \le i < j \le m} x_i x_j e_{\ell,i} e_{\ell,j} \right) \\
&= \sum_{i=1}^m x_i^2 \left(\sum_{\ell=1}^k e_{\ell,i}^2 \right) + 2 \sum_{1 \le i < j \le m} x_i x_j \left(\sum_{\ell=1}^k e_{\ell,i} e_{\ell,j} \right)
\end{aligned}
$$

So the coefficients for quadric Q are

$$
\begin{aligned}
a_0 &= -r^2, \ a_i = 0 \text{ for } 1 \le i \le m \\
a_{i,i} &= 1 - \left(\sum_{\ell=1}^k e_{\ell,i}^2 \right) \text{ and} \\
a_{i,j} &= - \left(\sum_{\ell=1}^k e_{\ell,i} e_{\ell,j} \right) \text{ for } i \ne j.
\end{aligned}
$$

Suppose that V is in *general quadric position*, i.e. no hyperplane in $\mathbb{R}^\nu$ contains more than ν points of $\widehat{\mathcal{X}}$, just like they assume in [1]. Then we have an analogue of lemma 2.2. in [1].

Lemma 4.1. Given $V \subset \mathbb{R}^m$ in general quadric position, and a quadric selecting $S \subset V$. Then there exists set $T \subset V$, $|T| = \nu := m(m+3)/2$ such that for the quadric $Q(T)$ the following holds. Let A, b, a_0 define $Q(T)$. Then

$$
\begin{aligned}
&x^T A x + x^T b + a_0 \le 0, \ x \in S, \\
&x^T A x + x^T b + a_0 \ge 0, \ x \in V \setminus S, \\
&x^T A x + x^T b + a_0 = 0 \text{ for at most } \nu \text{ points } x \in V \setminus S.
\end{aligned}
$$

The proof is the same as for lemma 2.2 in [1]. Since a set S that solves our problem is selectable by a quadric, the lemma applies, and the algorithm proposed in [1] finds S in time $O(N^{\nu+1})$ (the only modification is that instead of selecting a set that gives the smallest covariance determinant, we select a set that has the smallest error of rank-k approximation).