**History PCA: A new algorithm for Streaming PCA**

Streaming PCA tries to find k-dimensional subspace which can explain the most variation of the d-dimensional data points into memory sequentially.

Data set X. For X with large N and d, we may not be able to store matrices of size N×d or d×d due to limited memories. Then streaming PCA come to play. They only need to use O(d) or O(Bd) memory to store current sample or current block of samples.

Streaming PCA often suffer from slow convergence. 🡨 not fully utilize information in previously streamed data, which came in sequentially and contain valuable info.

New algorithm: compute the top-k PCs in streaming setting with O(Bd) memory requirement (B = block size, eg. 1 or 10); much faster convergence speed (effective use of past info), memory cost is O(d)

When full data is given in advance, optimal way to estimate PC is to conduct eigen decomposition on sample cov. mat. 1/N·(XTX). But it’s often impossible to form d×d sample cov. mat. for large scale data. So we compute 1/N·(XTX)v = 1/N·XT(Xv) in power method without explicitly forming sample cov. mat.

VR-PCA: main idea is to reformulate PCA as a stochastic optimization problem and apply variance reduction techniques;

VR-PCA looks similar to streaming PCA but it cannot obtain first update until the second pass 🡪 not suitable in streaming setting

Streaming PCA algorithms

Oja’s algorithm: traditional stochastic gradient decent method: each data point that come into memory can be viewed as a random sample drawn from an underlying distribution

Block stochastic power method: use each block of data to construct an estimator of cov. mat. and applies power method in each iteration to update eigenvectors.

Our method is also under the setting of streaming i.i.d. data samples and it differs from other streaming algorithms: History PCA exploits key summaries in historical data to achieve better performance

Block stochastic power method has limitations: block size B needs to be very large for this algorithm to work and B depends highly on the target accuracy; it cannot converge to the true eigenvectors of

Our algorithm: use more past info; do matrix-vector multiplication iteratively either until the algorithm converges or for a pre-specified number of times 🡪 we can further reduce the variance of estimates; (we assign equal weights to all data samples/blocks in each iteration)

Algorithm

form blocks of sample, each with size B, as data streaming in, and denote the ith data block by a B×d matrix Xi=(xi1, xi2, …, xiB)T where xit is a data point of dimension d for t = 1,…,B

main (rank-1): [Notes]

extended (rank-k): [Notes]; replace d-dimensional vector w with d×k matrix Q; replace normalization update by QR-decomposition

*Theoretical analysis (skipped)*

largest-principal-angle metric measures the distances b/w the estimated principal vectors and the ground truth

Experiment findings:

history PCA is the best for the case of recovering top eigenvector; the error from history PCA continues to decrease as the sample size of streaming data increases

Performance of block power method depends highly on B (performance improves as B increases); but History PCA does not depend on block size

[QUESTIONS]

1. largest-principal-angle-based distance function?

**Stochastic PCA and SVD algorithm with an exponential convergence rate**

VR-PCA: uses computationally cheap stochastic iterations, yet converges exponentially fast to the optimal solution

PCA: given a data matrix X whose columns consist of n instances in Rd, we are finding a k-dimensional subspace on which the projection of the data has largest possible variance.

When dimension d and size n are modest, this problem can be solved by SVD of X, with a runtime of O(min{nd2, n2d}) 🡪 impossible for large data sets;

Common alternative: iterative methods (eg. power iterations). If cov. mat. has bounded spectral norm and an eigengap λ b/w its first and second eigenvalues, then these algorithms can be shown to produce a unit vector that is ϵ-far from v1. But this is prohibitive for large data when λ is small.

An alternative to these are stochastic and incremental algorithms. It performs cheaper iterations by choosing some xi and updating current iterate using only xi. But the convergence rate, if known, is slow, with the number of required iterations scaling linearly with 1/ϵ and additional problem parameters. This is prohibitive for high-accuracy solution.

VR-PCA combines both advantages above. The setting in which we apply this is different: previous ones relied on strong convexity of optimization problem (and often assumed unconstrained domains); this algorithm tries to minimize the function that is nowhere convex (it is concave everywhere).

Algorithm: [Notes] (based on Oja’s algorithm for stochastic PCA optimization)

Slow convergence rate is because of the constant variance of stochastic term added in each step. We changed it in a way which encourages the variance of stochastic term to decay over time.

Experiments: convergence rate for VR-PCA is exponential but it is sub-exponential for Oja’s algorithms.

For MNIST and CCAT data, k=1, decaying step size of Oja’s algorithm is more suitable for the initial phase, and the resulting hybrid (initializes VR-PCA with the result of running n iterations of Oja’s algorithm) can perform better than each alone.

**Fast stochastic algorithms for SVD and PCA: Convergence Properties and Convexity**

Problem: compute the subspace spanned by the top k left singular vectors of a d×n matrix, where k << min{n, d}. (equivalent as computing the subspace of top k eigenvectors of XXT)

For large scale matrices X, exact eigendecomposition is infeasible. We use either deterministic algorithms or stochastic algorithms.

Deterministic algorithms are accurate but require a full pass over the matrix for each iteration (and in worst case many such passes are required 🡪 polynomial in the eigengap)

Stochastic algorithms have cheap iterations and the number is indep. of size of matrix, but they are not suitable for obtaining a high accuracy solution

A twice-differentiable function F on a subset of Rd is convex if its Hessian is always positive semidefinite. If it’s always positive definite and ≻ λI for some λ > 0, that function is λ-strongly convex; if the Hessian is always ≺ sI for some s ≥ 0, then the function is s-smooth.

VR-PCA algorithm and a Block Version

From k=1 to k>1 case, a simple technique is deflation (recover leading eigenvectors one by one, each time using k=1 algorithm).

Disadvantage: it requires a positive eigengap b/w all top k eigenvalues; otherwise the algorithm is not guaranteed to converge. 🡪 an algorithm which simultaneously recovers all k leading eigenvectors is preferable

Algorithm for k>1: [Notes]

numeric rank (nrank) of A: for any d×d matrix A, nrank(A) is at most the rank of A (which is at most d).

Variance-reduced gradient descent methods to non-convex SVD and PCA problems:

Even in block case where we attempt to recover subspace of k>1 eigenvectors simultaneously, such methods enjoys a convergence rate comparable to k=1 case.

Preceding the algorithm with a single power iteration can significantly improve the resulting bounds.

On a suitably chosen convex neighborhood of a global optimum, the function is strongly-convex. But the radius around the optimum where this happens scales with the eigengap, which precludes a better runtime with currently available algorithms.

[QUESTIONS]

1. What does ≽ mean?
2. Detailed explanation of Theorem 1 and 2.
3. Rayleigh quotient (section 5)

**Convergence of Stochastic Gradient Descent for PCA**

Problem of PCA in a streaming stochastic setting (to find a direction of approx. maximal variance, based on a stream of i.i.d. data points in Rd).

Simple and computationally cheap algorithm is stochastic gradient descent (SGD), which incrementally updates its estimate based on each new data point. But analyzing its performance is challenging due to non-convex nature of the problem. Existing guarantees rely on non-trivial eigengap assumption on cov. mat., which is unnecessary.

This paper provides the eigengap-free convergence guarantees for SGD in context of PCA.

SGD is highly efficient in terms of memory and runtime per iteration: require storage of a single d-dimensional vector and perform vector-vector and vector-scalar products in each iteration.

SGD has another property: it is statistically optimal, attaining the same statistical estimation error rate as exact empirical risk minimization.

Finite-time convergence guarantees are obtained under an eigengap assumption that the difference b/w largest and 2nd largest eigenvalues of E[**xx**T] are separated by a fixed value λ > 0.

Main results:

1. Provided the first SGD convergence guarantee which doesn’t pose an eigengap assumption. Proved that if step size is chosen appropriately, then after T iterations starting from random initialization, with positive prob., SGD returns an optimal solution, where p is a parameter depending on how algorithm is initialized:
   1. initialized from a warm-start point w0 s.t. for some leading eigenvector v of cov. mat., then p = O (1);
   2. uniform random initialization on unit Euclidean sphere, p = O (d);
   3. more sophisticated initialization (require first O(d) iterations but no warm-start point), p = O (nA), (nA is the numerical rank of cov. mat.).
2. When λ > 0, proved an SGD convergence guarantee of O(p/λT) with positive prob.

Convergence without an eigengap assumption

Convergence with an eigengap assumption