Streaming, Memory-Limited PCA

Constantine Caramanis

The University of Texas at Austin
constantine@utexas.edu

Joint work with Ioannis Mitliagkas and Prateek Jain

September 18, 2013

a simple problem

- $\mathbf{y}_i \sim N(0, \theta \mathbf{v}\mathbf{v}^\top + I_{p\times p}),\ i = 1, \ldots, n.$
- $\theta = O(1)$, $\mathbf{v} \in \mathbb{R}^p$.
- goal: compute **v**.

a simple problem

- $\mathbf{y}_i \sim N(0, \theta \mathbf{v}\mathbf{v}^\top + I_{p\times p}),\ i = 1, \ldots, n.$
- $\theta = O(1)$, $\mathbf{v} \in \mathbb{R}^p$.
- goal: compute **v**.
- algorithm: SVD of $(1/n)\sum \mathbf{y}_i \mathbf{y}_i^\top$.
- sample complexity: $n = O(p)$.

why it's interesting

- noise dominates each sample: $\mathbf{y}_i \sim N(0, \theta \mathbf{v}\mathbf{v}^\top + I_{p \times p})$

$$\mathbf{y}_i \perp \mathbf{y}_j \perp \mathbf{v}$$

- in fact,

$$\frac{\text{signal}}{\text{noise}} = \frac{\theta}{\sqrt{p}} \to 0.$$

why it works

- $\mathbf{w}_i \sim N(0, I_{p \times p}),\ i = 1, \ldots, n$
- $\sigma_{\max}\left(\frac{1}{n}\sum_i \mathbf{w}_i \mathbf{w}_i^\top\right) \approx 1 + \sqrt{p/n}$.
- For $\theta = O(1)$, $n = O(p)$, the spike sticks out.

what more do we want?

- storage (memory) requirements:

$$\text{input } O(p) \longrightarrow \underbrace{\text{algorithm } O(p^2)}_{\text{covariance matrix}} \longrightarrow \text{output } O(p)$$

what more do we want?

- storage (memory) requirements:

$$\text{input } O(p) \longrightarrow \underbrace{\text{algorithm } O(p^2)}_{\text{covariance matrix}} \longrightarrow \text{output } O(p)$$

- memory complexity (storage): $O(p^2)$

what more do we want?

- storage (memory) requirements:

$$\text{input } O(p) \longrightarrow \underbrace{\text{algorithm } O(p^2)}_{\text{covariance matrix}} \longrightarrow \text{output } O(p)$$

- memory complexity (storage): $O(p^2)$

- want memory complexity: $O(p)$.

why? a problem i want to solve

- outlier / change point detection in high dimensional data.
- data are images, scenes, or community behavior.
- building instrumented (UT Austin)

- high-rate but inexpensive sensors. computing equipment?

p vs p^2?

- iPhone: 512Mb RAM, 32 Gb Storage
- Desktop: 16 Gb RAM, 1,000 Gb Storage
- Server: 256 Gb RAM, XX,000 Gb Storage

- not p vs p^2

another motivation: distributed PCA

- data distributed on k servers. communication complexity?

another motivation: distributed PCA

- data distributed on k servers. communication complexity?

- $\sqrt{p}$ servers each have $\sqrt{p}$ data points:

$$\mathbf{y}_1^{(j)}, \ldots, \mathbf{y}_{\sqrt{p}}^{(j)}, \qquad \mathbf{y}_i^{(j)} \sim N(0, \theta \mathbf{v}\mathbf{v}^\top + I).$$

- communication complexity to compute $\mathbf{v}$?

another motivation: distributed PCA

- data distributed on k servers. communication complexity?
- $\sqrt{p}$ servers each have $\sqrt{p}$ data points:

$$\mathbf{y}_1^{(j)}, \ldots, \mathbf{y}_{\sqrt{p}}^{(j)}, \qquad \mathbf{y}_i^{(j)} \sim N(0, \theta \mathbf{v}\mathbf{v}^\top + I).$$

- communication complexity to compute $\mathbf{v}$?
- individually, no server can compute anything but noise.

$$\sigma_{\max}(\text{noise}) \sim p^{1/4}.$$

another motivation: distributed PCA

- data distributed on k servers. communication complexity?
- $\sqrt{p}$ servers each have $\sqrt{p}$ data points:

$$\mathbf{y}_1^{(j)}, \ldots, \mathbf{y}_{\sqrt{p}}^{(j)}, \qquad \mathbf{y}_i^{(j)} \sim N(0, \theta \mathbf{v}\mathbf{v}^\top + I).$$

- communication complexity to compute $\mathbf{v}$?
- individually, no server can compute anything but noise.

$$\sigma_{\max}(\text{noise}) \sim p^{1/4}.$$

- $O(p)$-memory algorithm implies each server sends one single vector.

sketching and randomized linear algebra

- Drineas, Kannan, Mahoney, Tropp, Woodruff, and co-authors
- sketch: subsample or project rows/columns/elements
- applications to streaming linear algebra

sketching and randomized linear algebra

- can compute with low memory:

$$Y \times Q = \underbrace{\begin{pmatrix} | & | & \cdots & | \\ \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_n \\ | & | & \cdots & | \end{pmatrix}}_{n \approx p} \underbrace{\begin{pmatrix} | & \vdots & | \\ \mathbf{q}_1 & \vdots & \mathbf{q}_k \\ | & \vdots & | \end{pmatrix}}_{k \ll p}$$

- Q random Gaussian matrix, or random $+/-$ matrix.

sketching and randomized linear algebra

- can also subsample columns, rows, or elements of Y

sketching and randomized linear algebra

- upper and (nearly matching) lower bounds using communication complexity.
- but: worst-case bounds.

sketching and randomized linear algebra

- data are guassian: subsampling, combining, projecting won't work.

sketching and randomized linear algebra

- data are guassian: subsampling, combining, projecting won't work.

- bounds are of the form:

$$\|Y - \tilde{Y}_k\| \leq (1+\varepsilon)\|Y - Y_k\|_F$$

sketching and randomized linear algebra

- data are guassian: subsampling, combining, projecting won't work.

- bounds are of the form:

$$\|Y - \tilde{Y}_k\| \leq (1+\varepsilon)\|Y - Y_k\|_F$$

- $\mathbf{y}_i \sim N(0, \theta \mathbf{v}\mathbf{v}^\top + I) \rightarrow$ spectrum does not decay:

$$\|Y - Y_k\|_F = O(p).$$

sketching and randomized linear algebra

- data are guassian: subsampling, combining, projecting won't work.

- bounds are of the form:

$$\|Y - \tilde{Y}_k\| \leq (1+\varepsilon)\|Y - Y_k\|_F$$

- $\mathbf{y}_i \sim N(0, \theta \mathbf{v}\mathbf{v}^\top + I) \rightarrow$ spectrum does not decay:

$$\|Y - Y_k\|_F = O(p).$$

- does not help recover the spike: $\mathbf{v}$

online learning

- Warmuth and Kuzmin: online PCA
- goal: regret minimization.
- memory: $O(p^2)$

stochastic gradient ascent

- SVD solves

$$\max_{\|\mathbf{v}\|=1} : \mathbf{v}^\top \left[\sum \mathbf{y}_i \mathbf{y}_i^\top\right] \mathbf{v}$$

stochastic gradient ascent

- SVD solves

$$\max_{\|\mathbf{v}\|=1} : \mathbf{v}^\top \left[\sum \mathbf{y}_i \mathbf{y}_i^\top\right] \mathbf{v}$$

- instead, stochastic gradient ascent:

$$\hat{\mathbf{v}}^{(t+1)} = \hat{\mathbf{v}}^{(t)} + \eta_t \langle \hat{\mathbf{v}}^{(t)}, \mathbf{y}_t \rangle \mathbf{y}_t.$$

stochastic gradient ascent

- SVD solves

$$\max_{\|\mathbf{v}\|=1} : \mathbf{v}^\top \left[\sum \mathbf{y}_i \mathbf{y}_i^\top\right] \mathbf{v}$$

- instead, stochastic gradient ascent:

$$\hat{\mathbf{v}}^{(t+1)} = \hat{\mathbf{v}}^{(t)} + \eta_t \langle \hat{\mathbf{v}}^{(t)}, \mathbf{y}_t \rangle \mathbf{y}_t.$$

- standard Robbins-Monro results: asymptotic convergence guaranteed.

stochastic gradient ascent

- SVD solves

$$\max_{\|\mathbf{v}\|=1} : \mathbf{v}^\top \left[\sum \mathbf{y}_i \mathbf{y}_i^\top \right] \mathbf{v}$$

- instead, stochastic gradient ascent:

$$\hat{\mathbf{v}}^{(t+1)} = \hat{\mathbf{v}}^{(t)} + \eta_t \langle \hat{\mathbf{v}}^{(t)}, \mathbf{y}_t \rangle \mathbf{y}_t.$$

- standard Robbins-Monro results: asymptotic convergence guaranteed.

- rate...?

stochastic gradient ascent

- GROUSE (Balzano, Nowak, Recht & Balzano, Wright)
- takes steps on Grassmanian.
- also handles missing data (imputation)
- local convergence results w/o noise.
- in simulations, works with noise.

stochastic gradient ascent

- note: can write a $1-d$ non-linear recursion for $c_t = \langle \hat{\mathbf{v}}(t), \mathbf{v} \rangle$, but seems difficult(?) to analyze.

stochastic gradient ascent

- dualizing gives a convex problem:

$$\begin{aligned} \max : \quad & \langle \sum \mathbf{y}_i \mathbf{y}_i^\top, V \rangle \\ \text{s.t.} : \quad & 0 \preceq V \preceq I \\ & \text{trace}(V) = 1. \end{aligned}$$

- stochastic gradient ascent: $V_{t+1} = \mathscr{P}(V_t + \eta \mathbf{y}_i \mathbf{y}_i^\top)$.

- see Arora, Cotter, Srebro '13.

the algorithm: streaming power method

- challenge with stochastic gradient: variance.

the algorithm: streaming power method

- $\underbrace{\mathbf{y}_1, \ldots, \mathbf{y}_T}_{B_1}, \underbrace{\mathbf{y}_{T+1}, \ldots, \mathbf{y}_{2T}}_{B_2}, \ldots, \underbrace{\mathbf{y}_{T(\log p-1)+1}, \ldots, \mathbf{y}_{T\log p}}_{B_{\log p}}$
- pick a random $\mathbf{v}(0) \in \mathbb{R}^p$.
- $\mathbf{v}(t+1) = \left[\frac{1}{B_t}\sum_{i=1}^{B_t} \mathbf{y}_i \mathbf{y}_i^\top\right] \mathbf{v}(t)$

the algorithm: streaming power method

- $\underbrace{\mathbf{y}_1, \ldots, \mathbf{y}_T}_{B_1}, \underbrace{\mathbf{y}_{T+1}, \ldots, \mathbf{y}_{2T}}_{B_2}, \ldots, \underbrace{\mathbf{y}_{T(\log p - 1)+1}, \ldots, \mathbf{y}_{T \log p}}_{B_{\log p}}$
- pick a random $\mathbf{v}(0) \in \mathbb{R}^p$.
- $\mathbf{v}(t+1) = \left[\frac{1}{B_t} \sum_{i=1}^{B_t} \mathbf{y}_i \mathbf{y}_i^\top \right] \mathbf{v}(t) = \frac{1}{B_t} \sum_{i=1}^{B_t} \mathbf{y}_i (\mathbf{y}_i^\top \mathbf{v}(t))$.

the algorithm: streaming power method

- $\underbrace{\mathbf{y}_1, \ldots, \mathbf{y}_T}_{B_1}, \underbrace{\mathbf{y}_{T+1}, \ldots, \mathbf{y}_{2T}}_{B_2}, \ldots, \underbrace{\mathbf{y}_{T(\log p - 1)+1}, \ldots, \mathbf{y}_{T \log p}}_{B_{\log p}}$

- pick a random $\mathbf{v}(0) \in \mathbb{R}^p$.

- $\mathbf{v}(t+1) = \left[\frac{1}{B_t} \sum_{i=1}^{B_t} \mathbf{y}_i \mathbf{y}_i^\top\right] \mathbf{v}(t) = \frac{1}{B_t} \sum_{i=1}^{B_t} \mathbf{y}_i (\mathbf{y}_i^\top \mathbf{v}(t))$.

- power method with slightly different matrices.

key steps

- initial random guess: $|\langle \mathbf{v}(0), \mathbf{v} \rangle| \geq c/\sqrt{p}$.
- letting $B_t = O(p)$ gives $\| \frac{1}{B_t} \sum_{i=1}^{B_t} \mathbf{y}_i \mathbf{y}_i^\top - \Sigma \| < \varepsilon$.
- need further concentration thanks to independence of error.
- issues: initial error decay is slow, but then becomes linear.
- straightforward extension to recovering top k principal components.

extensions

- erasures: coordinates of $\mathbf{y}_i$ erased iid w.p. q

extensions

- erasures: coordinates of $\mathbf{y}_i$ erased iid w.p. q
- unbiased covariance estimator:

$$\Sigma = (q^{-1} - q^{-2})\mathrm{diag}\left[\frac{1}{n}\sum_i \mathbf{y}_i \mathbf{y}_i^\top\right] + q^{-2} \cdot \frac{1}{n}\sum_i \mathbf{y}_i \mathbf{y}_i^\top.$$

can use this for power method in streaming fashion.

extensions

- erasures: coordinates of $\mathbf{y}_i$ erased iid w.p. q

- unbiased covariance estimator:

$$\Sigma = (q^{-1} - q^{-2})\mathrm{diag}\left[\frac{1}{n}\sum_i \mathbf{y}_i\mathbf{y}_i^\top\right] + q^{-2} \cdot \frac{1}{n}\sum_i \mathbf{y}_i\mathbf{y}_i^\top.$$

  can use this for power method in streaming fashion.

- essentially same results: $1/q^2$ adjustment, as expected.

extensions

- distributed PCA with communication constraints: data distributed on k servers.
- $\sqrt{p}$ servers each have $\sqrt{p}$ data points:

$$\mathbf{y}_1^{(j)}, \ldots, \mathbf{y}_{\sqrt{p}}^{(j)}, \qquad \mathbf{y}_i^{(j)} \sim N(0, \theta \mathbf{v}\mathbf{v}^\top + I).$$

- each server transmits 1 (k) vector(s).

extensions

- distributed PCA with communication constraints: data distributed on k servers.
- $\sqrt{p}$ servers each have $\sqrt{p}$ data points:

$$\mathbf{y}_1^{(j)}, \dots, \mathbf{y}_{\sqrt{p}}^{(j)}, \qquad \mathbf{y}_i^{(j)} \sim N(0, \theta \mathbf{v}\mathbf{v}^\top + I).$$

- each server transmits 1 (k) vector(s).
- how to parallelize?

extensions

- coordinate-wise corruptions?
- sample-wise corruptions?
- tracking?

conclusion

find out more from:

`http://users.ece.utexas.edu/~cmcaram/`

or e-mail:

`constantine@utexas.edu`