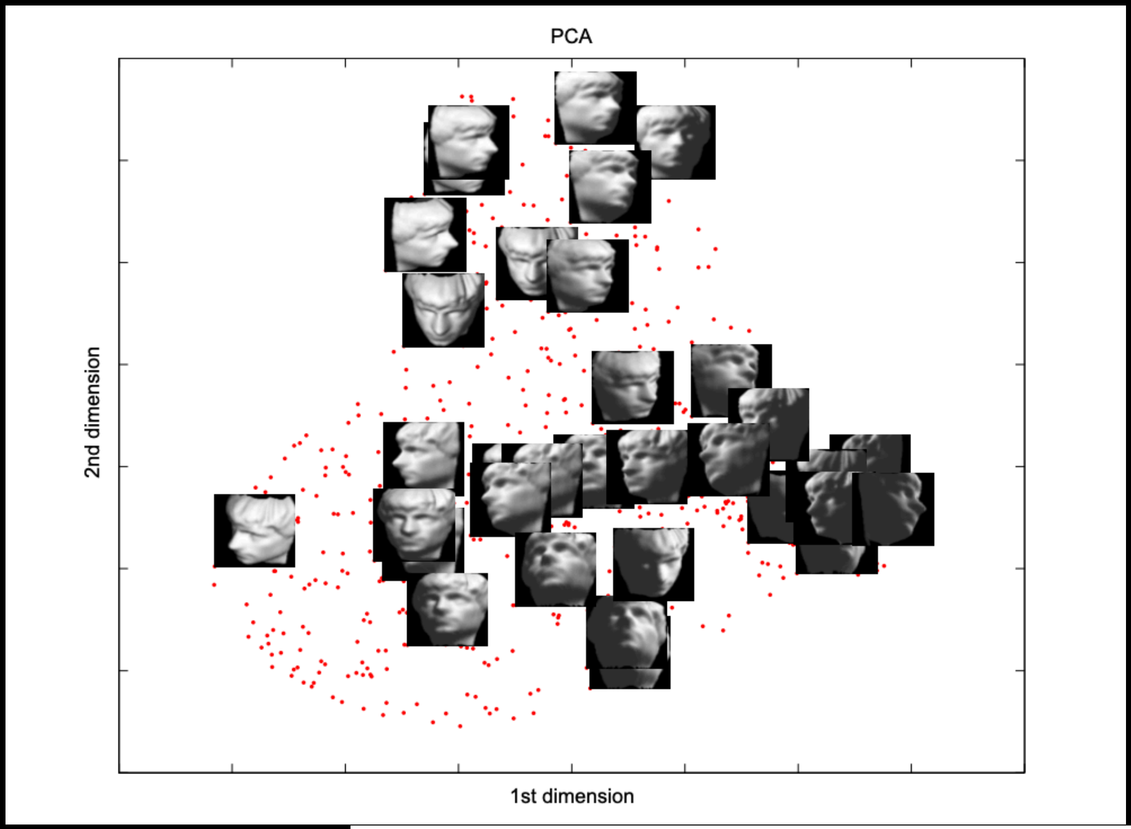
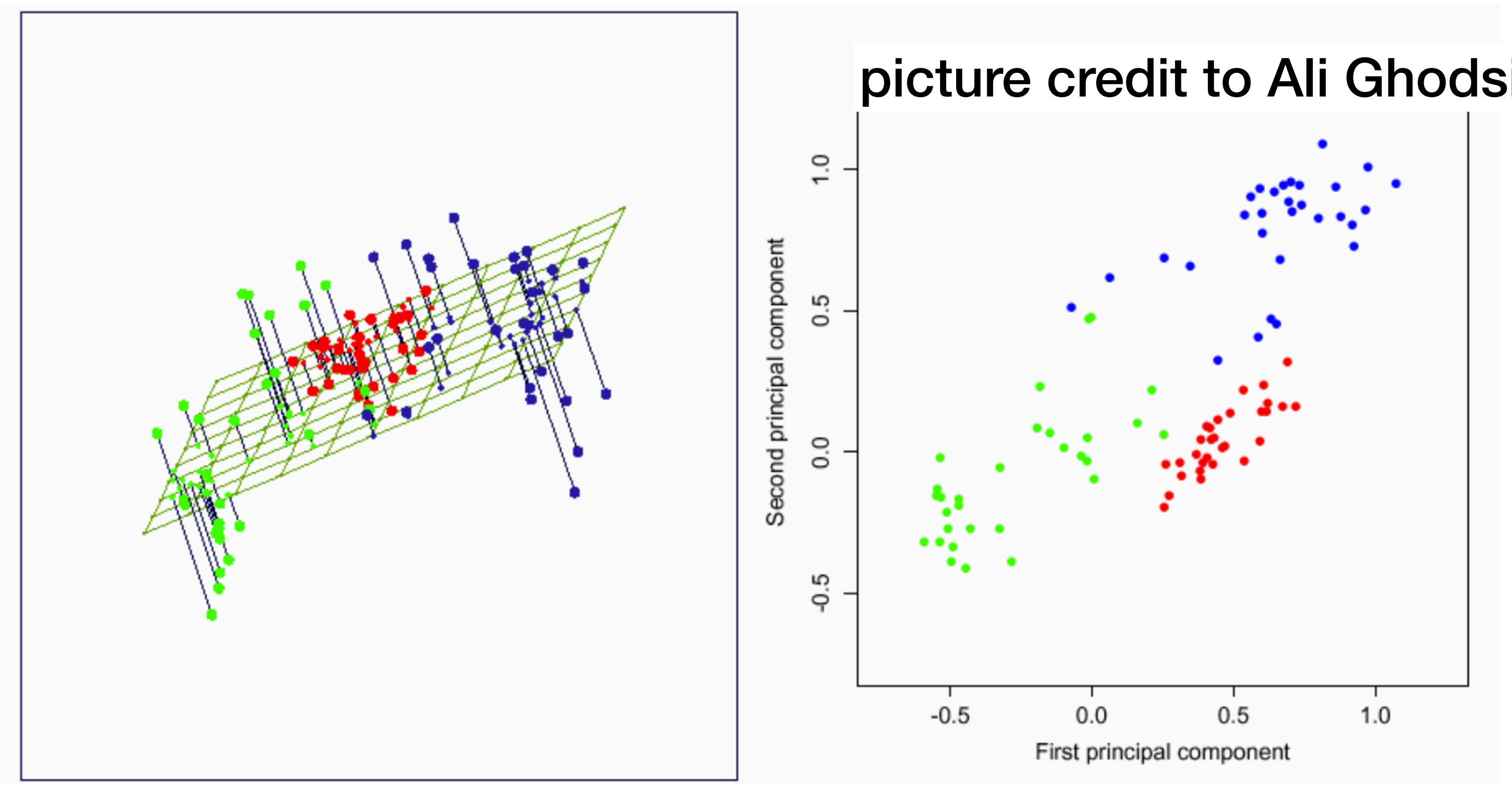
Exponentially convergent stochastic k-PCA *without* variance reduction

Cheng Tang
Amazon (AWS AI Labs)



k-PCA: popular dimension reduction method with many applications



The best rank-two linear approximation

Batch vs Online k-PCA

Batch k-pca

- Apply k-PCA on the entire dataset
- Usually done by SVD

Online k-PCA

- Approximate k-PCA without observing all data
- Data points come one-by-one

Two popular online 1-PCA algorithms & their convergence rates

Oja's rule

- Online version of power method (has a *length normalization* step)
- Converges in order O(1/t) for 1-PCA
- Connection to neural networks

$$w^t \leftarrow \frac{w^{t-1} + \eta^t x x^T w^{t-1}}{\|w^{t-1} + \eta^t x x^T w^{t-1}\|}$$

Balsubramani-Dasgupta-Freud 2013

Krasulina's method

- Stochastic gradient ascent on Rayleigh quotient for 1-PCA
- Converges in order O(1/t) for 1-PCA

$$w^t \leftarrow w^{t-1} + \eta^t (x x^T - \frac{(x^T w^{t-1})^2}{\|w^{t-1}\|^2} I_d) w^{t-1}$$

Balsubramani-Dasgupta-Freud 2013

Two popular online 1-PCA algorithms & their convergence rates

Oja's rule

- Easy to generalize to k-PCA
- Converges in order O(1/t) for k-PCA by recent analysis

Allen-Zhu 2017

Krasulina's method

- Not obvious how to generalize to k-PCA

VR-PCA

- Mixed Oja's rule (online) & power iteration (batch)
- Exponential convergence rate

Shamir 2016

Algorithm 1 VR-PCA: Vector version ($k = 1$)

Parameters: Step size η, epoch length m
Input: Data matrix $X = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$; Initial unit vector $\tilde{\mathbf{w}}_0$
for $s = 1, 2, \ldots$ **do**
$\quad \tilde{\mathbf{u}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i \left(\mathbf{x}_i^\top \tilde{\mathbf{w}}_{s-1} \right)$
$\quad \mathbf{w}_0 = \tilde{\mathbf{w}}_{s-1}$
$\quad$ **for** $t = 1, 2, \ldots, m$ **do**
$\quad\quad$ Pick $i_t \in \{1, \ldots, n\}$ uniformly at random
$\quad\quad \mathbf{w}'_t = \mathbf{w}_{t-1} + \eta \left(\mathbf{x}_{i_t} \left(\mathbf{x}_{i_t}^\top \mathbf{w}_{t-1} - \mathbf{x}_{i_t}^\top \tilde{\mathbf{w}}_{s-1} \right) + \tilde{\mathbf{u}} \right)$
$\quad\quad \mathbf{w}_t = \frac{1}{\|\mathbf{w}'_t\|} \mathbf{w}'_t$
$\quad$ **end for**
$\quad \tilde{\mathbf{w}}_s = \mathbf{w}_m$
end for

Algorithm 2 VR-PCA: Block version

Parameters: Rank k, Step size η, epoch length m
Input: Data matrix $X = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$; Initial $d \times k$ matrix $\tilde{W}_0$ with orthonormal columns
for $s = 1, 2, \ldots$ **do**
$\quad \tilde{U} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i \left(\mathbf{x}_i^\top \tilde{W}_{s-1} \right)$
$\quad W_0 = \tilde{W}_{s-1}$
$\quad$ **for** $t = 1, 2, \ldots, m$ **do**
$\quad\quad B_{t-1} = VU^\top$, where $USV^\top$ is an SVD decomposition of $W_{t-1}^\top \tilde{W}_{s-1}$
$\quad\quad\quad \triangleright$ Equivalent to

$$B_{t-1} = \arg \min_{B^\top B = I} \|W_{t-1} - \tilde{W}_{s-1} B\|_F^2$$

$\quad\quad$ Pick $i_t \in \{1, \ldots, n\}$ uniformly at random
$\quad\quad W'_t = W_{t-1} +$
$\quad\quad\quad \eta \left(\mathbf{x}_{i_t} \left(\mathbf{x}_{i_t}^\top W_{t-1} - \mathbf{x}_{i_t}^\top \tilde{W}_{s-1} B_{t-1} \right) + \tilde{U} B_{t-1} \right)$
$\quad\quad W_t = W'_t \left(W_t'^\top W'_t \right)^{-1/2}$
$\quad$ **end for**
$\quad \tilde{W}_s = W_m$
end for

Information-theoretic lower bound

- This implies a O(1/t) upper bound on convergence rate of *ALL* online k-PCA algorithms

$$\mathbb{E}[\Delta^n] \geq \Omega(\frac{\sigma^2}{n})$$

$$\text{for } \frac{\lambda_1 \lambda_{k+1}}{(\lambda_k - \lambda_{k+1})^2},$$

Vu and Lei, AISTATS, 2012

Our observation

When data is of rank 1, Krasulina's method for 1-PCA can be viewed as

$$W^t \leftarrow W^{t-1} + \eta^t s^t (r^t)^T$$

$$Var(s^t r^t) \propto \text{loss}$$

Krasulina's method is already doing variance-reduction on low-rank data!

Information-theoretic lower bound — a closer look

- The lower bound to-the-right implies a 1/t upper bound on convergence rate of online k-PCA algorithms **in the general case**
- Our result says otherwise when data is **low-rank**

$$\mathbb{E}[\Delta^n] \geq \Omega(\frac{\sigma^2}{n})$$

$$\text{for } \frac{\lambda_1 \lambda_{k+1}}{(\lambda_k - \lambda_{k+1})^2},$$

Vu and Lei, AISTATS, 2012

Matrix Krasulina

Projection

Residual

$$W^t \leftarrow W^{t-1} + \eta^t s^t (r^t)^T$$

$$\text{Orthonormalize rows of } W^t$$

Makes sure r is orthogonal to the k-subspace

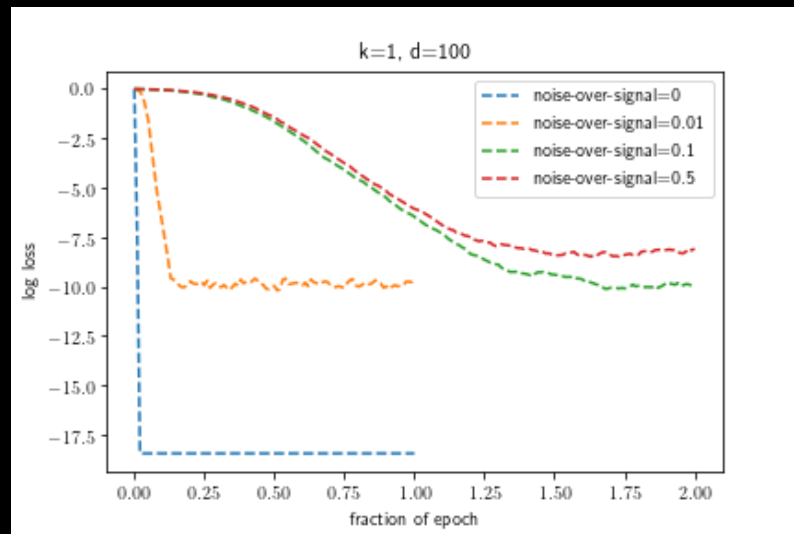
Our main results

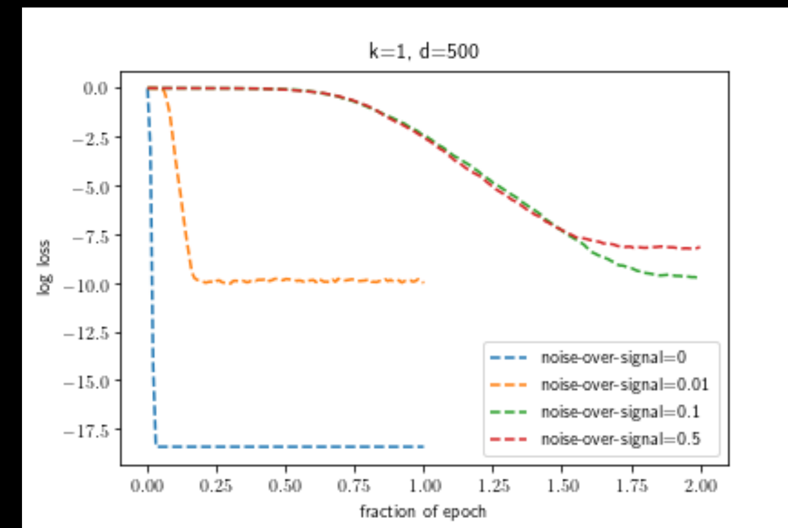
If Cov(X) is of low rank
Matrix Krasulina converges
exponentially for k-PCA

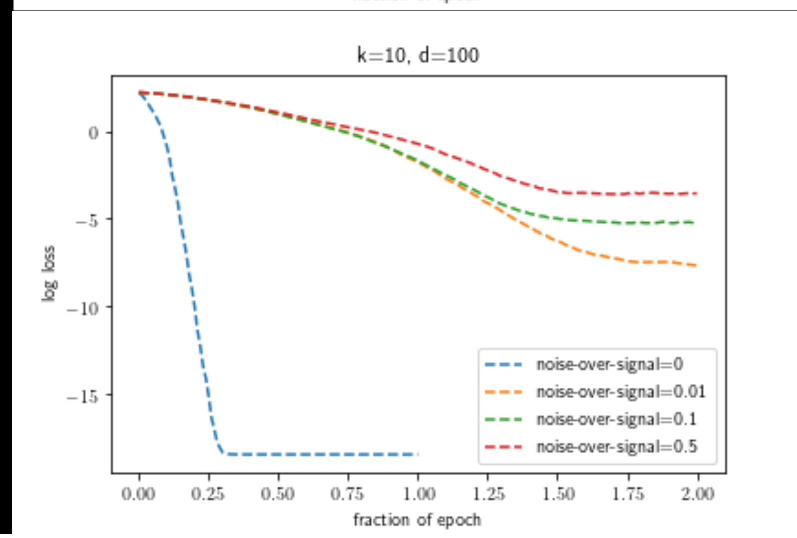
- ☆ Set learning rate to be constant

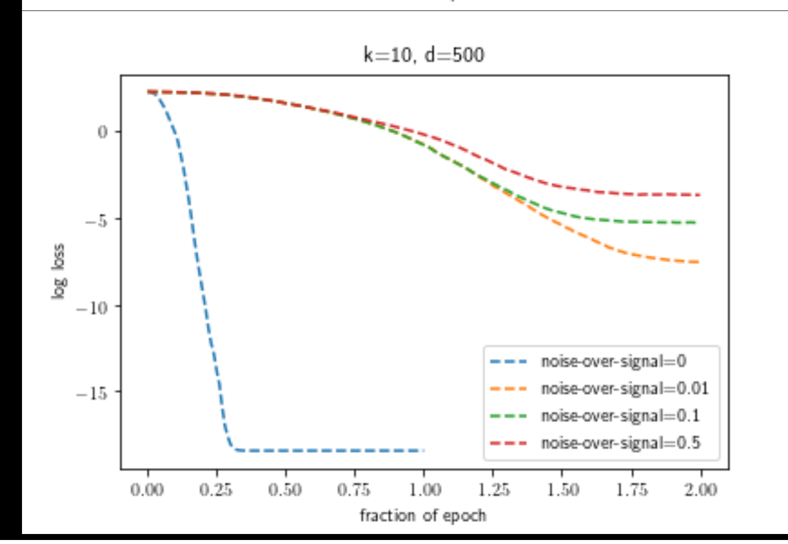
In general case
Matrix Krasulina converges
of order O(1/t) for k-PCA

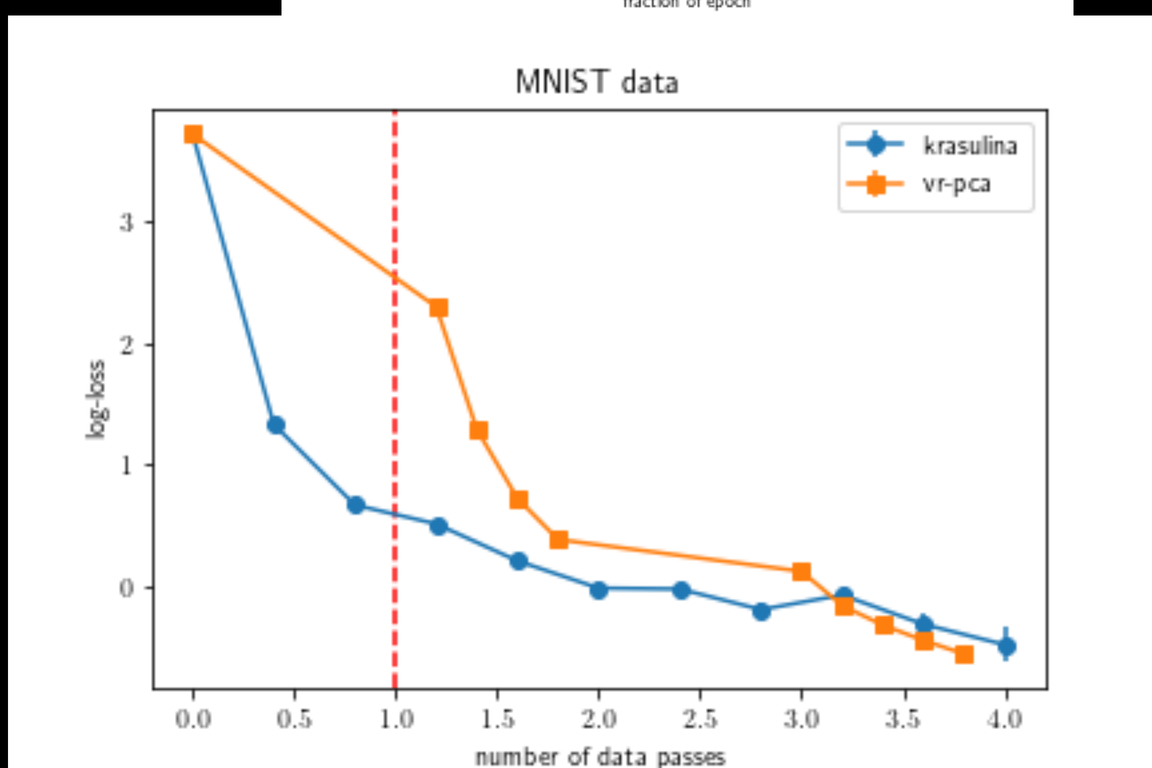
- ☆ Set learning rate to be ~1/t

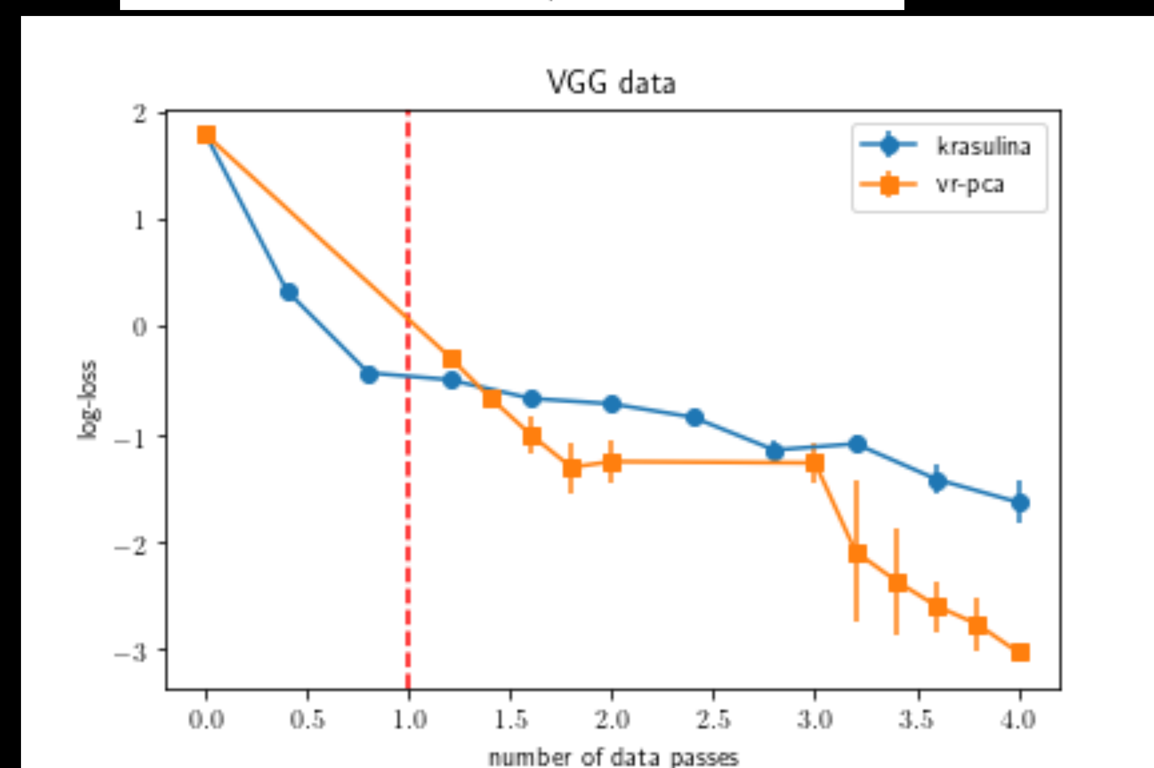

k=1, d=100
log loss
fraction of epoch
noise-over-signal=0
noise-over-signal=0.01
noise-over-signal=0.1
noise-over-signal=0.5


k=1, d=500
log loss
fraction of epoch
noise-over-signal=0
noise-over-signal=0.01
noise-over-signal=0.1
noise-over-signal=0.5


k=10, d=100
log loss
fraction of epoch
noise-over-signal=0
noise-over-signal=0.01
noise-over-signal=0.1
noise-over-signal=0.5


k=10, d=500
log loss
fraction of epoch
noise-over-signal=0
noise-over-signal=0.01
noise-over-signal=0.1
noise-over-signal=0.5


MNIST data
log-loss
number of data passes
krasulina
vr-pca


VGG data
log-loss
number of data passes
krasulina
vr-pca

Open problem

Can we characterize convergence rate when data is *nearly low-rank ?*