MAHOUT-817
PCA options for SSVD
working notes

November 30, 2011

1 Mean of rows

1.1 Recap of SSVD flow.

Modified SSVD Algorithm. Given an $m \times n$ matrix $\mathbf{A}$, a target rank $k \in \mathbb{N}_1$, an oversampling parameter $p \in \mathbb{N}_1$, and the number of additional power iterations $q \in \mathbb{N}_0$, this procedure computes an $m \times (k+p)$ SVD $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$(some notations are adjusted):

1. Create seed for random $n \times (k+p)$ matrix $\mathbf{\Omega}$. The seed defines matrix $\mathbf{\Omega}$ using Gaussian unit vectors per one of suggestions in [**?**].

2. $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$, $\mathbf{Y} \in \mathbb{R}^{m \times (k+p)}$.

3. Column-orthonormalize $\mathbf{Y} \to \mathbf{Q}$ by computing thin decomposition $\mathbf{Y} = \mathbf{Q}\mathbf{R}$. Also, $\mathbf{Q} \in \mathbb{R}^{m \times (k+p)}$, $\mathbf{R} \in \mathbb{R}^{(k+p) \times (k+p)}$. I denote this as $\mathbf{Q} = \operatorname{qr}(\mathbf{Y}).\mathbf{Q}$.

4. $\mathbf{B}_0 = \mathbf{Q}^\top \mathbf{A} : \ \mathbf{B} \in \mathbb{R}^{(k+p) \times n}$. (Another way is $\mathbf{R}^{-1}\mathbf{Y}^\top \mathbf{A}$, depending on whether we beleive if size of A less than size of Q).

5. If $q > 0$ repeat: for $i = 1..q$: $\mathbf{B}_i^\top = \mathbf{A}^\top \operatorname{qr}\left(\mathbf{A}\mathbf{B}_{i-1}^\top\right).\mathbf{Q}$ (power iterations step)

6. Compute Eigensolution of a small Hermitian $\mathbf{B}_q\mathbf{B}_q^\top = \hat{\mathbf{U}}\mathbf{\Lambda}\hat{\mathbf{U}}^\top$. $\mathbf{B}_q\mathbf{B}_q^\top \in \mathbb{R}^{(k+p) \times (k+p)}$.

7. Singular values $\mathbf{\Sigma} = \mathbf{\Lambda}^{0.5}$, or, in other words, $s_i = \sqrt{\sigma_i}$.

8. If needed, compute $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

9. If needed, compute $\mathbf{V} = \mathbf{B}_q^\top \hat{\mathbf{U}}\mathbf{\Sigma}^{-1}$. Another way is $\mathbf{V} = \mathbf{A}^\top \mathbf{U}\mathbf{\Sigma}^{-1}$.

1.2 $\mathbf{B}_0$ pipeline mods

This option considers that data points are rows in the $m \times n$ input matrix

$$\mathbf{A} = \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_m \end{pmatrix}$$

Mean of rows is n-vector

$$\begin{aligned} \boldsymbol{\xi} &= \begin{pmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_n \end{pmatrix} \\ &= \frac{1}{m} \sum_i^m \mathbf{a}_i. \end{aligned}$$

Let$\tilde{\mathbf{A}}$ be $\mathbf{A}$ with the mean subtracted.

$$\tilde{\mathbf{A}} = \begin{pmatrix} \mathbf{a_1} - \boldsymbol{\xi} \\ \mathbf{a_2} - \boldsymbol{\xi} \\ \vdots \\ \mathbf{a}_m - \boldsymbol{\xi} \end{pmatrix}.$$

We denote $m \times n$ mean matrix

$$\boldsymbol{\Xi} = \begin{pmatrix} \boldsymbol{\xi} \\ \boldsymbol{\xi} \\ \vdots \\ \boldsymbol{\xi} \end{pmatrix}$$

$\mathbf{B}_0$ pipeline starts with notion that since $\tilde{\mathbf{A}}$ is dense, its mutliplications are very expensive. Hence, we factorize $\mathbf{Y}$ as

$$\begin{aligned} \mathbf{Y} &= \tilde{\mathbf{A}}\boldsymbol{\Omega} \\ &= \mathbf{A}\boldsymbol{\Omega} - \Xi\boldsymbol{\Omega} \end{aligned}$$

Current $\mathbf{B}_0$ pipeline already takes care of $\mathbf{A}\boldsymbol{\Omega}$, but the term $\boldsymbol{\Xi}\boldsymbol{\Omega}$ will need more work.

The term $\Xi\boldsymbol{\Omega}$ will have identical rows $\boldsymbol{\xi}\boldsymbol{\Omega}$ so we need to precompute just one dense n-vector $\boldsymbol{\xi}\boldsymbol{\Omega}$. This computation is very expensive since matrix Ω is dense (potentially several orders of magnitude bigger than input $\mathbf{A}$) and the median $\boldsymbol{\xi}$ is dense as well, even that we don't actually have to materialize any of Ω.
Question is whether we could just ignore it since $\mathbb{E}\left(\boldsymbol{\xi}\boldsymbol{\Omega}\right)=0$.
Alternatively, we could just brute-force it by creating a separate distributed computation of this over n.

⇐Outstanding issue!!!

Moving onto $\mathbf{B}$ and $\mathbf{B}\mathbf{B}^{\top}$. Here and on we assume $\mathbf{B}\equiv\mathbf{B}_0$ and omit the index for compactness.

$$\begin{aligned}\mathbf{B} &= \mathbf{Q}^{\top}\tilde{\mathbf{A}} & (1)\\ &= \mathbf{Q}^{\top}\mathbf{A}-\mathbf{Q}^{\top}\boldsymbol{\Xi}. & (2)\end{aligned}$$

Again, current pipeline takes care of $\mathbf{Q}^{\top}\mathbf{A}$ but product $\mathbf{Q}^{\top}\Xi$ would need more work.

Let $\mathbf{W}=\mathbf{Q}^{\top}\boldsymbol{\Xi}$.

$$\begin{aligned}\mathbf{W} &= \sum^{m}\mathbf{Q}_{i,*}\boldsymbol{\Xi}_{i,*}^{\top}\\ &= \mathbf{s}_Q\boldsymbol{\xi}^{\top}\end{aligned}$$

where

$$\mathbf{s}_Q=\sum_{i=1}^{m}\mathbf{Q}_{i,*} \qquad (3)$$

is sum of all rows of $\mathbf{Q}$.

Since B_0 pipeline computes $\mathbf{Q}^{\top}\mathbf{A}$ column-wise over columns of $\mathbf{Q}$ and $\mathbf{A}$, the first thought is that (2) can be computed column-wise as well with computation seeded by the $\mathbf{s}_Q$ and $\boldsymbol{\xi}$ vectors.

One problem with our first thought is that the $\mathbf{s}_Q$ term is not yet known at the time of formation of $\mathbf{B}$ columns because formation of final $\mathbf{Q}$ blocks happens in the same distributed map task that produces initial $\mathbf{Q}^{\top}\mathbf{A}$ blocks. Hence, the sum of Q rows at that moment would not be available. But we probably can fix our output later at the time when $\mathbf{s}_Q$ would already have been known.

Let $\mathbf{b}_i=\mathbf{B}_{*,i}$, $\tilde{\mathbf{b}}_i=\left(\mathbf{Q}^{\top}\mathbf{A}\right)_{*,i}$. Then correction for $\mathbf{B}$ output would be

$$\mathbf{b}_i=\tilde{\mathbf{b}}_i-\xi_i\mathbf{s}_Q. \qquad (4)$$

Moving on to $\mathbf{B}\mathbf{B}^{\top}$:

$$\mathbf{B}\mathbf{B}^\top = \sum_i^n \mathbf{b}_i\mathbf{b}_i^\top$$

$$\begin{aligned}
\mathbf{b}_i\mathbf{b}_i^\top &= \left(\tilde{\mathbf{b}}_i - \xi_\mathbf{i}\mathbf{s}_\mathbf{Q}\right)\left(\tilde{\mathbf{b}}_i - \xi_i\mathbf{s}_Q\right)^\top \\
&= \tilde{\mathbf{b}}_i\tilde{\mathbf{b}}_i^\top - \xi_i\tilde{\mathbf{b}}_i\mathbf{s}_Q^\top - \xi_\mathbf{i}\mathbf{s}_Q\tilde{\mathbf{b}}_i^\top - \xi_\mathbf{i}^\mathbf{2}\mathbf{s}_Q\mathbf{s}_Q^\top \\
&= \tilde{\mathbf{b}}_i\tilde{\mathbf{b}}_i^\top - \left[\xi_i\tilde{\mathbf{b}}_i\mathbf{s}_Q^\top + \left(\xi_i\tilde{\mathbf{b}}_i\mathbf{s}_Q^\top\right)^\top\right] + \xi_\mathbf{i}^\mathbf{2}\mathbf{s}_Q\mathbf{s}_Q^\top.
\end{aligned}$$

Let $\mathbf{C} = \left[\sum_i^n \xi_i\tilde{\mathbf{b}}_i\right]\mathbf{s}_Q^\top$, then

$$\begin{aligned}
\mathbf{B}\mathbf{B}^\top &= \sum_i^n \tilde{\mathbf{b}}_i\tilde{\mathbf{b}}_i^\top && (5)\\
&- \left[\mathbf{C} + \mathbf{C}^\top\right] && (6)\\
&+ n\|\boldsymbol{\xi}\|_2^2\mathbf{s}_Q\mathbf{s}_Q^\top. && (7)
\end{aligned}$$

So we can compute $\tilde{\mathbf{B}} = \sum_i \tilde{\mathbf{b}}_i\tilde{\mathbf{b}}_i^\top$ right away, that's what Bt-job does.

We also can add the term $n\|\boldsymbol{\xi}\|_2^2\mathbf{s}_Q\mathbf{s}_Q^\top$ in front end before we do eigendecomposition since it is a tiny matrix and at that point $\mathbf{s}_Q$ is already known.

Bt-job can also aggregate and collect vector

$$\mathbf{s}_{\tilde{B}} = \sum_i^n \xi_i\tilde{\mathbf{b}}_i. \tag{8}$$

Then we also can produce

$$\mathbf{C} = \mathbf{s}_{\tilde{B}}\mathbf{s}_Q^\top \tag{9}$$

in the front end.

So, Bt job needs to be amended to produce (3) and 8.

PCA would be primarily interested in $\mathbf{V}$ or $\mathbf{V}_\sigma$ output of the decomposition in order to fold in new items back into PCA space, so we need to correct $\mathbf{V}$ job as well in this case to fix output of Bt-job per (4) which we must seed with $\boldsymbol{\xi}$ and $\mathbf{s}_Q$.

1.3 Power Iterations (aka B_i pipeline) additions

Power iterations pipeline produces $\mathbf{B}_i^\top = \tilde{\mathbf{A}}^\top\mathrm{qr}\left(\tilde{\mathbf{A}}\mathbf{B}_{i-1}^\top\right).\mathbf{Q}$. Similarly to versions of $\mathbf{B}$, each iteration would produce corrective vector $\mathbf{w}_{i-1}$.

First, we need to amend power iteration work flow to fix output of previous Bt-job on the fly with to reconstruct correct $\mathbf{B}_{i-1}$similarly to what is done in the $\mathbf{V}$ job per (4):

$$\mathbf{B}_{i-1} = \tilde{\mathbf{B}}_{i-1} - \mathbf{W}_{i-1}.$$

Second, again, $\tilde{\mathbf{A}}$ multipliers are a problem because they would be dense and perhaps should be decomposed in a way similar to $\mathbf{B}_0$ pipeline.

==============> to be ctd. <==============

~~Another note is that we run eigendecomposition only after the last iteration so the term $\mathbf{s}_{\tilde{B}}$ needs to be computed only during the last iteration.~~