MAHOUT-817
PCA options for SSVD
working notes

December 6, 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 $\mathbf{Y} \rightarrow \mathbf{Q}$ 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 $\mathbf{Q} = \mathrm{qr}\left(\mathbf{Y}\right).\mathbf{Q}$.
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 $i = 1..q$: $\mathbf{B}_i^\top = \mathbf{A}^\top \mathrm{qr}\left(\mathbf{A}\mathbf{B}_{i-1}^\top\right).\mathbf{Q}$ (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*

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

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

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

We denote $m \times n$ mean matrix

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

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

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

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

*Here and on I use tilde notation to denote something that hasn't yet been adjusted for the effects of mean subtraction.

The term $\Xi\boldsymbol{\Omega}$ will have identical rows

$$\mathbf{s}_\Omega = \boldsymbol{\Omega}^\top \boldsymbol{\xi}.$$

so we need to precompute just one dense $(k+p)$-vector $\boldsymbol{\Omega}^\top\boldsymbol{\xi}$ before we enter Q-job. This computation *may* turn out fairly expensive since matrix Ω is dense (potentially several orders of magnitude bigger than input $\tilde{\mathbf{A}}$) and the median $\boldsymbol{\xi}$ is dense as well, even that we don't actually have to materialize any of Ω.[†]

Formally, fixing rows of Y will therefore look like

$$\mathbf{Y}_{l,*} = \left(\tilde{\mathbf{A}}\boldsymbol{\Omega}\right)_{l,*} - \mathbf{s}_\Omega. \tag{1}$$

Fixing B. Here and on we assume $\mathbf{B} \equiv \mathbf{B}_0$ and omit the index for compactness.

$$\begin{aligned} \mathbf{B} &= \mathbf{Q}^\top\mathbf{A} && (2)\\ &= \mathbf{Q}^\top\tilde{\mathbf{A}} - \mathbf{Q}^\top\Xi. && (3)\end{aligned}$$

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

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

$$\begin{aligned} \mathbf{W} &= \sum_{i=1}^{m} \mathbf{Q}_{i,*}\Xi_{i,*}^\top \\ &= \mathbf{s}_Q\boldsymbol{\xi}^\top \end{aligned}$$

where

$$\mathbf{s}_Q = \sum_{i=1}^{m} \mathbf{Q}_{i,*} \tag{4}$$

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

Since B_0 pipeline computes $\mathbf{Q}^\top\tilde{\mathbf{A}}$ column-wise over columns of $\mathbf{Q}$ and $\tilde{\mathbf{A}}$, the first thought is that (3) 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

[†] *Question is whether we could just ignore it since* $\mathbb{E}\left(\boldsymbol{\Omega}^\top\boldsymbol{\xi}.\right) = 0$. This question is more or less answered, it looks the answer is no, we should not be ignoring this product by default.

One question for experimentation is whether it is worth to create a separate distributed job for this. its $n \times k$ iterations. maybe it is not, assuming $n \approx 10^7$ and $k \approx 10^2$. it would also require artificial map splits for this since there's no input for $\boldsymbol{\Omega}$.

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.

Hence, we won't actually ever form $\mathbf{B}$ in this pipeline. Instead, we will produce $\tilde{\mathbf{B}} = \mathbf{Q}^\top\tilde{\mathbf{A}}$ and vector (4) and provide a way to produce columns of $\mathbf{B}$ on the fly given columns of $\tilde{\mathbf{B}}$ and $\mathbf{s}_Q$ in the future jobs.

Let denote *columns* $\mathbf{b}_i = \mathbf{B}_{*,i}$, and $\tilde{\mathbf{b}}_i = \tilde{\mathbf{B}}_{*,i}$. Then correction for $\mathbf{B}$ columns will look like

$$\mathbf{b}_i = \tilde{\mathbf{b}}_i - \xi_i\mathbf{s}_Q. \tag{5}$$

Getting a little ahead, we will use (5) to fix columns of B in V-job and ABt-job. Those two jobs are the primary customers of this equation.

Fixing $\mathbf{BB}^\top$ on the fly (in front end). Moving on to $\mathbf{BB}^\top$:

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

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

Let's denote $(k+p)\times(k+p)$ matrix $\mathbf{C} = \left[\sum_i^n \xi_i\tilde{\mathbf{b}}_i\right]\mathbf{s}_Q^\top$. Taking sum,

$$\begin{aligned}
\mathbf{BB}^\top &= \tilde{\mathbf{B}}\tilde{\mathbf{B}}^\top - && (6)\\
&- \left[\mathbf{C} + \mathbf{C}^\top\right] + && (7)\\
&+ \|\boldsymbol{\xi}\|_2^2\mathbf{s}_Q\mathbf{s}_Q^\top. && (8)
\end{aligned}$$

Bt-job already produces $\tilde{\mathbf{B}}\tilde{\mathbf{B}}^\top$ partial products (1 per reducer) and front end finishes the summation. So no change here.

We also can add the term $\|\boldsymbol{\xi}\|_2^2\mathbf{s}_Q\mathbf{s}_Q^\top$ 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

$$\mathbf{s}_{\tilde{B}} = \tilde{\mathbf{B}}\boldsymbol{\xi} = \sum_i^n \xi_i\tilde{\mathbf{b}}_i. \tag{9}$$

Then we also can produce

$$\mathbf{C} = \mathbf{s}_{\tilde{B}}\mathbf{s}_Q^\top \tag{10}$$

in the front end as well.

Bottom line, Bt job needs to be augmented to produce (4) and (9) if PCA computation is desired.

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 (5) which we must seed with $\boldsymbol{\xi}$ and $\mathbf{s}_Q$.

1.3 Power Iterations (aka $\mathbf{B}_i$ pipeline) additions

Power iterations pipeline produces $\mathbf{B}_i^\top = \mathbf{A}^\top \mathrm{qr}\left(\mathbf{A}\mathbf{B}_{i-1}^\top\right).\mathbf{Q}$. Here and on in power iteration-related sections I use index i to denote the iteration number.

Here, we assume that $\mathbf{s}_{Q,0} \equiv \mathbf{s}_Q$, $\mathbf{s}_{\tilde{B},0} \equiv \mathbf{s}_{\tilde{B}}$ and $\boldsymbol{\xi}$ are known and are results of $\mathbf{B}_0$ pipeline above.

We also assume outer loop $\forall i \in [1, q]$ for all the following steps.

Fixing $\mathbf{B}_{i-1}$. First, the on-the fly correction for $\mathbf{B}$ columns must be applied per (5) using $\mathbf{s}_{Q,i-1}$ produced by previous iteration or $\mathbf{B}_0$ iteration as at this point we would have access to $\tilde{\mathbf{B}}_{i-1}$ and $\mathbf{s}_{Q,i-1}$. So, the mapper of ABt-job must be preloaded with $(k+p)$-long $\mathbf{s}_{Q,i-1}$ vector known from previous iteration.

Fixing $\mathbf{Y}_i$. Next, we consider

$$\begin{aligned}\mathbf{Y}_i &= \mathbf{A}\mathbf{B}_{i-1}^\top \\ &= \tilde{\mathbf{A}}\mathbf{B}_{i-1}^\top - \boldsymbol{\Xi}\mathbf{B}_{i-1}^\top.\end{aligned}$$

Again, $\boldsymbol{\Xi}\mathbf{B}_{i-1}^\top$ term would have identical rows $\mathbf{B}_{i-1}\boldsymbol{\xi}$ but the problem is that it needs to be known before ABt job. But we alrady know it from the previous iteration as $\mathbf{s}_{\tilde{B},i-1}$. The rows of $\mathbf{Y}_i$, l-th of which I denote as $(\mathbf{Y}_i)_{l,*}$, must be corrected per

$$(\mathbf{Y}_i)_{l,*} = \left(\tilde{\mathbf{A}}\mathbf{B}_{i-1}^\top\right)_{l,*} - \mathbf{s}_{\tilde{B},i-1}. \tag{11}$$

The row $\left(\tilde{\mathbf{A}}\mathbf{B}_{i-1}^\top\right)_{l,*}$ is known at the time of the reducing of ABt-job. So reducers of ABt-job must be preloaded with $\mathbf{s}_{\tilde{B},i-1}$ $(k+p)$-long vector in order to finish correction of $\mathbf{Y}_i$ rows before handing them off to first step of QR. So, practically no additional effort here. Notice similarity between (11) and (1).

Fixing $\mathbf{B}_i$. Similarly to $\mathbf{B}_0$ pipeline, we don't produce $\mathbf{B}_i$ output directly in this iteration. Instead, we produce $\tilde{\mathbf{B}}_i = \mathbf{Q}_i^\top \tilde{\mathbf{A}}$ and $(k+p)$-long vector $\mathbf{s}_{Q,i}$ exactly per (4). Nothing new here. (in fact, code producing $\mathbf{s}_Q$ and $\mathbf{s}_{Q,i}$ would be exactly the same and shares Bt job with $\mathbf{B}_0$ pipeline). Subsequent jobs will use (8) to fix columns of B the same way as was discussed there.

1.4 Fixing $\mathbf{B}\mathbf{B}^\top$ and the rest of pipeline with power iterations.

In case q power iterations are enabled, fixing $\mathbf{B}\mathbf{B}^\top$steps and on run exactly as defined in $\mathbf{B}_0$ pipeline, except they must assume $\tilde{\mathbf{B}} \equiv \tilde{\mathbf{B}}_q$,$\mathbf{s}_Q \equiv \mathbf{s}_{Q,q}$, $\mathbf{s}_{\tilde{B}} \equiv \mathbf{s}_{\tilde{B},q}$.