# MAHOUT-817
# PCA options for SSVD
# working notes*

February 24, 2012

## Contents




*Dmitriy Lyubimov, dlyubimov@apache.org

# 1 Approach

## 1.1 Outline: mean of rows

We approach general PCA and dimensionality reduction problem with respect two input expressed as Mahout's distributed row matrix format. We also assume data points are row vectors in such a matrix. We denote such $m \times n$ input matrix as $\tilde{\mathbf{A}}$*:

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

Mean of rows is n-vector

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

We denote $m \times n$ mean matrix as

$$\boldsymbol{\Xi} = \begin{pmatrix} \boldsymbol{\xi}^\top \\ \boldsymbol{\xi}^\top \\ \vdots \\ \boldsymbol{\xi}^\top \end{pmatrix} \in \mathbb{R}^{m \times n}$$

Traditional approach under these settings is to find a mean of all rows and subtract it from all data points (row vectors) of the input

$$\mathbf{A} = \tilde{\mathbf{A}} - \boldsymbol{\Xi}$$

and then run reduced $k$-rank SVD:

$$\mathbf{A} \simeq \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top.$$

At this point rows of $\mathbf{U}$ correspond to original data points (rows of $\tilde{\mathbf{A}}$) converted to PCA space (and we are done).

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

## 1.2 Transformations to/from for new observation data points

Transformation of any new data point observation of$n$-vector$\tilde{\mathbf{a}}$ *into* reduced dimentionality PCA space $k$-vector $\mathbf{u}$ will look like

$$\tilde{\mathbf{u}} = \boldsymbol{\Sigma}^{-\mathbf{1}}\mathbf{V}^\top (\tilde{\mathbf{a}} - \boldsymbol{\xi}) \tag{1}$$

(this operation is essentially an SVD fold-in operation corrected for the mean subtraction).

Inverse transformation (from reduced PCA space into original space) looks like

$$\tilde{\mathbf{a}} = \boldsymbol{\xi} + \mathbf{V}\boldsymbol{\Sigma}\tilde{\mathbf{u}}. \tag{2}$$

## 1.3 MAHOUT-817 goals: why brute force approach is hard in context of big data computations

In context of massive computations, input $\tilde{\mathbf{A}}$ is often rather sparse. Sparse matrices are packed and their subsequent operations within Mahout framework are optimized to account for degenerate nature of zero elements computation. Sometimes such reduction of need for flops and space may approach several orders of magnitude. However, mean subtraction step would turn such sparse inputs into a dense matrix $\left(\tilde{\mathbf{A}} - \boldsymbol{\Xi}\right)$. Such intermediate input will take a lot of space and subsequent SVD will require a lot of flops. Fortunately, this can be addressed in context of SSVD method in a way that will be (almost) cost-equivalent to a regular SSVD computation on original sparse input $\tilde{\mathbf{A}}$.

MAHOUT-817 addresses two goals:

- Provide column-wise mean computation step in the whole pipeline (or use outside mean vector if already available)
- Lift the dense matrix data concerns per above.

The sparser the original input is, the more efficiency gain is to be had by using SSVD PCA options compared to brute-force approach.

If original input is 100% dense, SSVD PCA options will have roughly the same cost as brute-force approach.

## 2 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 $\mathbf{Q} = \mathrm{qr}\left(\mathbf{Y}\right).\mathbf{Q}$.
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 $\mathbf{V} = \mathbf{A}^\top\mathbf{U}\mathbf{\Sigma}^{-1}$.

## 3 $\mathbf{B}_0$ pipeline mods

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

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

**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

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

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.

The term $\boldsymbol{\Xi}\boldsymbol{\Omega}$ 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 $\Omega$ 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 $\Omega$.[†]

Formally, fixing rows of Y will therefore look like

$$\mathbf{Y}_{l,*} = \left(\tilde{\mathbf{A}}\boldsymbol{\Omega}\right)_{l,*} - \mathbf{s}_\Omega. \tag{3}$$

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

$$\begin{aligned} \mathbf{B} &= \mathbf{Q}^\top \mathbf{A} && (4)\\ &= \mathbf{Q}^\top \tilde{\mathbf{A}} - \mathbf{Q}^\top \boldsymbol{\Xi}. && (5) \end{aligned}$$

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

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

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

where

$$\mathbf{s}_Q = \sum_{i=1}^{m} \mathbf{Q}_{i,*} \tag{6}$$

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 (5) can be computed column-wise as well with computation seeded by the $\mathbf{s}_Q$ and $\boldsymbol{\xi}$ vectors.

[†] *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 $\boldsymbol{\Omega}$.

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.

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 (6) 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

$$\mathbf{b}_i = \tilde{\mathbf{b}}_i - \xi_i\mathbf{s}_Q. \tag{7}$$

Getting a little ahead, we will use (7) 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,

$$\begin{aligned}
\mathbf{BB}^\top &= \tilde{\mathbf{B}}\tilde{\mathbf{B}}^\top - && (8)\\
&- \left[\mathbf{C} + \mathbf{C}^\top\right] + && (9)\\
&+ \|\boldsymbol{\xi}\|_2^2\mathbf{s}_Q\mathbf{s}_Q^\top. && (10)
\end{aligned}$$

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

$$\mathbf{s}_{\tilde{B}} = \tilde{\mathbf{B}}\boldsymbol{\xi} = \sum_i^n \xi_i\tilde{\mathbf{b}}_i. \tag{11}$$

Then we also can produce

$$\mathbf{C} = \mathbf{s}_{\tilde{B}}\mathbf{s}_Q^\top \tag{12}$$

in the front end as well.

Bottom line, Bt job needs to be augmented to produce (6) and (11) 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 (7) which we must seed with $\boldsymbol{\xi}$ and $\mathbf{s}_Q$.

# 4 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.

## 4.1 Fixing $\mathbf{B}_{i-1}$.

First, the on-the fly correction for $\mathbf{B}$ columns must be applied per (7) 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.

## 4.2 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

$$(\mathbf{Y}_i)_{l,*} = \left(\tilde{\mathbf{A}}\mathbf{B}_{i-1}^\top\right)_{l,*} - \mathbf{s}_{\tilde{B},i-1}. \tag{13}$$

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 (13) and (3).

### 4.3 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 (6). 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 (10) to fix columns of B the same way as was discussed there.

## 5 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 $\tilde{\mathbf{B}} \equiv \tilde{\mathbf{B}}_q$,$\mathbf{s}_Q \equiv \mathbf{s}_{Q,q}$, $\mathbf{s}_{\tilde{B}} \equiv \mathbf{s}_{\tilde{B},q}$.

## 6 R simulation code

See Mahout-817.