PCA options for SSVD
working notes

November 28, 2011

1 Mean of rows

1.1 $\mathbf{B}_0$ 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

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

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

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

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

> The term $\Xi\mathbf{\Omega}$ will have identical rows $\boldsymbol{\xi}\mathbf{\Omega}$ so we need to precompute just one dense n-vector $\boldsymbol{\xi}\mathbf{\Omega}$. 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* $\mathbb{E}\left(\boldsymbol{\xi}\mathbf{\Omega}\right) = 0$.

$\Leftarrow$Outstanding issue!!!

Alternatively, we could just brute-force it by creating a separate distributed computation of this over n.

Moving onto $\mathbf{B}$ and $\mathbf{B}\mathbf{B}^\top$:

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

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

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

Let also $\mathbf{a} \odot \mathbf{b} = \begin{pmatrix} a_1 b_1 \\ a_2 b_2 \\ \vdots \\ a_k b_k \end{pmatrix}$ to be a notation for element-wise vector product.

We see that all columns of $\mathbf{W}$ are identical, and, more specifically,

$$
\begin{aligned}
\mathbf{W}_{*,i} &= \mathbf{w} \\
&= \left(\mathbf{Q}^\top \boldsymbol{\Xi}\right)_{*,i} \\
&= \left[\sum_{i=1}^{m} \mathbf{Q}_{i,*}\right] \odot \boldsymbol{\xi} \\
&= \mathbf{s}_Q \odot \boldsymbol{\xi}
\end{aligned}
$$

where $\mathbf{s}_Q = \sum_{i=1}^{m} \mathbf{Q}_{i,*}$ 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 thatequation (2) can be computed column-wise as well with computation seeded by the $\mathbf{s}_Q \odot \boldsymbol{\xi}$ vector.

Flow problem with existing SSVD is the $\mathbf{s}$ term, which is not yet known at the time of formation of $\mathbf{B}$ columns because formation of $\mathbf{Q}$ blocks happens at the same pipeline that further on produces $\mathbf{B}$ columns. But we probably can produce it by the time of final summations of $\mathbf{B}\mathbf{B}^\top$.

Let $\mathbf{b}_i = \mathbf{B}_{*,i}$, $\tilde{\mathbf{b}}_i = \left(\mathbf{Q}^\top \mathbf{A}\right)_{*,i}$. Then $\mathbf{b}_i = \tilde{\mathbf{b}}_i - \mathbf{w}$

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

$$
\begin{aligned}
\mathbf{b}_i \mathbf{b}_i^\top &= \left(\tilde{\mathbf{b}}_i - \mathbf{w}\right)\left(\tilde{\mathbf{b}}_i - \mathbf{w}\right)^\top \\
&= \tilde{\mathbf{b}}_i \tilde{\mathbf{b}}_i^\top - \tilde{\mathbf{b}}_i \mathbf{w}^\top - \mathbf{w}\tilde{\mathbf{b}}_i^\top - \mathbf{w}\mathbf{w}^\top \\
&= \tilde{\mathbf{b}}_i \tilde{\mathbf{b}}_i^\top - \tilde{\mathbf{b}}_i \mathbf{w}^\top - \left(\tilde{\mathbf{b}}_i \mathbf{w}^\top\right)^\top + \mathbf{w}\mathbf{w}^\top .
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{B}\mathbf{B}^\top &= \sum_{i}^{n} \tilde{\mathbf{b}}_i \tilde{\mathbf{b}}_i^\top && (3) \\
&- \sum \left[\tilde{\mathbf{b}}_i \mathbf{w}^\top + \left(\tilde{\mathbf{b}}_i \mathbf{w}^\top\right)^\top\right] && (4) \\
&+ n \cdot \mathbf{w}\mathbf{w}^\top . && (5)
\end{aligned}
$$

Let $k \times k$ matrix $\mathbf{C} = \sum_{i}^{n} \tilde{\mathbf{b}}_i \mathbf{w}^\top$, and then we can rewrite equation (4) as

$$
\mathbf{B}\mathbf{B}^\top = \sum_{i}^{n} \tilde{\mathbf{b}}_i \tilde{\mathbf{b}}_i^\top - \mathbf{C} - \mathbf{C}^\top + n \cdot \mathbf{w}\mathbf{w}^\top .
$$

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 $n \cdot \mathbf{w}_i \mathbf{w}_i^\top$in front end since it is a tiny matrix and at this point $\mathbf{w}$ is already known. The task boils down to computing small $(k+p) \times (k+p)$matrix $\mathbf{C}$ and then subtracting $\mathbf{C} + \mathbf{C}^\top$ in front end as well. Note that

$$\begin{aligned}
\mathbf{C} &= \sum_i^n \tilde{\mathbf{b}}_i \mathbf{w}^\top \\
&= \left(\sum_i^n \tilde{\mathbf{b}}_i \right) \mathbf{w}^\top \\
&= \mathbf{s}_{\tilde{B}} \mathbf{w}^\top .
\end{aligned}$$

In this case, $\mathbf{s}_{\tilde{B}}$ can be output by Bt job as well. Hence $\mathbf{C}$ can be computed as an outer product of two small k-vectors in the front end as well.

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.

1.2 Power Iterations (aka B_i pipeline) additions

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

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

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.