

PCA options for SSVD

working notes

November 28, 2011

1 Mean of rows

1.1 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

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

\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 $\Xi\mathbf{\Omega}$ will need more work.

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

⇐Outstanding issue!!!

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

$$\mathbf{B} = \mathbf{Q}^\top \tilde{\mathbf{A}} \tag{1}$$

$$= \mathbf{Q}^\top \mathbf{A} - \mathbf{Q}^\top \Xi. \tag{2}$$

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

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

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

*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.

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

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 that (2) can be computed column-wise as well with computation seeded by the $\mathbf{s}_Q \odot \boldsymbol{\xi}$ vector.

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

$$\mathbf{b}_i = \tilde{\mathbf{b}}_i - \mathbf{w}. \quad (3)$$

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

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

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

$$\mathbf{B}\mathbf{B}^\top = \sum_i^n \tilde{\mathbf{b}}_i \tilde{\mathbf{b}}_i^\top \quad (4)$$

$$- \sum_i^n \left[\tilde{\mathbf{b}}_i \mathbf{w}^\top + (\tilde{\mathbf{b}}_i \mathbf{w}^\top)^\top \right] \quad (5)$$

$$+ n \cdot \mathbf{w} \mathbf{w}^\top. \quad (6)$$

Let $k \times k$ matrix $\mathbf{C} = \sum_i^n \tilde{\mathbf{b}}_i \mathbf{w}^\top$, and then we can rewrite equation (5) 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 $n \cdot \mathbf{w}\mathbf{w}^\top$ in front end before we do eigendecomposition since it is a tiny matrix and at that 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 the front end as well. Note that

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

In this case, $\mathbf{s}_{\tilde{\mathbf{B}}} = \sum_i^n \tilde{\mathbf{b}}_i$ 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}_σ 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 (3).

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

Power iterations pipeline produces $\mathbf{B}_i^\top = \mathbf{A}^\top \text{qr}(\mathbf{A}\mathbf{B}_{i-1}^\top) \cdot \mathbf{Q}$. 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} per (3).

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