

Stochastic Singular Value Decomposition

Stochastic SVD method in Mahout produces reduced rank Singular Value Decomposition output in its strict mathematical definition: $A=USV'$.

The benefits over other methods are:

- reduced flops required compared to Krylov subspace methods
- In map-reduce world, a fixed number of MR iterations required regardless of rank requested
- Tweak precision/speed balance with options.
- A is a Distributed Row Matrix where rows may be identified by any Writable (such as a document path). As such, it would work directly on the output of seq2sparse.
- As of 0.7 trunk, includes PCA and dimensionality reduction workflow (EXPERIMENTAL! Feedback on performance/other PCA related issues/ blogs is greatly appreciated.)

map-reduce characteristics:

SSVD uses at most 3 MR *sequential* steps (map-only + map-reduce + 2 optional parallel map-reduce jobs) to produce reduced rank approximation of U, V and S matrices. Additionally, two more map-reduce steps are added for each power iteration step if requested.

Potential drawbacks:

- potentially less precise (but adding even one power iteration seems to fix that quite a bit).

Documentation

[Overview and Usage](#)

Note: Please use 0.6 or later! for PCA workflow, please use 0.7 or later.

Publications

[Nathan Halko's dissertation](#) "Randomized methods for computing low-rank approximations of matrices" contains comprehensive definition of parallelization strategy taken in Mahout SSVD implementation and also some precision/scalability benchmarks, esp. w.r.t. Mahout Lanczos implementation on a typical corpus data set.

R simulation

[Non-parallel SSVD simulation in R with power iterations and PCA options](#). Note that this implementation is not most optimal for sequential flow solver, but it is for demonstration purposes only.

However, try this R code to simulate a meaningful input:

tests.R

```
n<-1000
m<-2000
k<-10

qi<-1

#simulated input
svalsim<-diag(k:1)

usim<- qr.Q(qr(matrix(rnorm(m*k, mean=3), nrow=m,ncol=k)))
vsim<- qr.Q(qr( matrix(rnorm(n*k,mean=5), nrow=n,ncol=k)))

x<- usim %*% svalsim %*% t(vsim)
```

and try to compare `ssvd.svd(x)` and stock `svd(x)` performance for the same rank k , notice the difference in the running time. Also play with power iterations (`qlter`) and compare accuracies of standard `svd` and `SSVD`.

Note: numerical stability of R algorithms may differ from that of Mahout's distributed version. We haven't studied accuracy of the R simulation. For study of accuracy of Mahout's version, please refer to Nathan's dissertation as referenced above.