Scaling SSVD method
and
Command Line Interface notes

Dmitriy Lyubimov
dlieu.7@gmail.com

January 2, 2011

Contents

1 Base algoirthm

Branch: ***givens-ssvd.*** This branch contains more or less stable code optimized for tall but relatively thin (n=30,000...50,000 non-zero elements). With 1G memory in child processes and right combination of block height, min split size and k,p parameters it is expected to be able to process 1 billion rows or more with 1 million dense (i.e. non-zero) data elements. However, network IO deficiencies will start to occur much sooner as discussed in the p. 6.1 of the working notes (this is one of TODOs).

Usage.

```
mahout ssvd <options>
```

Options.

`-k, --rank <int-value>` the requested SVD rank (minimum number of singular values and dimensions in U, V matrices)

`-p, --oversampling <int-value>` stochastic SVD oversampling. (k+p=500 is probably more than reasonable).

`-r, --blockHeight <int-value>` the number of rows of source matrix for block computations. Taller blocking causes more memory use but produces less blocks and therefore somewhat better running times. The most optimal mode from the running time point of view should be 1 block per 1 mapper. For 64Mb blocks this is probably somewhere around 30,000. However, Q-job mapper preallocates the memory for the entire Q block $(k+p) \times r$ even if it doesn't encounter that many rows. So it's ok to set it somewhat higher than 1 block per mapper would take to ensure exactly 1 block is formed, but setting it too high may cause memory management problems.

`-s, --minSplitSize <int-value>` the minimum split size to use in mappers. Since in this branch block formation happens in mappers, for significantly large -r and width of the input matrix the algorithm may not be able to read minimum $k+p$ rows and form a block of minimum height, so the job would bail out at the very first mapping step. If that is the case, then one of the recourses available is to force increase in the MapReduce split size using SequenceFileInputFormat.setMinSplitSize() property. Increasing this significantly over HDFS size will result in network IO overhead as discussed in p.6.2 of the working notes).

`--computeU <true|false>` Request computation of the U matrix (default true)

`--computeV <true|false>` Request computation of the V matrix (default true)

`--vHalfSigma <true|false>` compute $\mathbf{V}_\sigma = \mathbf{V}\mathbf{\Sigma}^{0.5}$ instead of $\mathbf{V}$ (affords for easier similarity computations between row-wise and column-wise items). Default false.

`--uHalfSigma <true|false>` compute $\mathbf{U}_\sigma = \mathbf{U}\mathbf{\Sigma}^{0.5}$ instead of $\mathbf{U}$. default false.

`--reduceTasks <int-value>` The number of reducers to use (where applicable): depends on size of the hadoop cluster

Standard Options.

`--input <glob>` HDFS glob specification where the DistributedRowMatrix input to be found.

`--output <hdfs-dir>` non-existent hdfs directory where to output $\mathbf{U}, \mathbf{V}$ and $\mathbf{\Sigma}$ (singular values) files.

`--tempDir <temp-dir>` temporary dir where to store intermediate files (cleaned up upon normal completion). This is a standard Mahout optional parameter.

Output. Singular values are output as a single-record Sequence File containing a dense vector with k singular values. U and V matrices are k-wide dense DistributedRowMatrix files. U matrix inherits row labels of the input matrix.

2 Matrix preprocessing

Branch: ssvd-preprocessing. This branch resolves issue defined in § 6.2 of the "working notes" document.

Ssvd-preprocessing branch equips class *VectorWritable* with the following preprocessing interface.

```
/**
 * Hooks into {@link VectorWritable} to provide matrix data
     preprocessing
 * capabilities to save on allocating long vectors .
 *
 * @author Dmitriy
 *
 */
public interface VectorPreprocessor extends Configurable {
    /**
     * called before starting processing a new vector in the writable.
     *
     * @param sequential true if vector is either dense or sequential
         sparse.
     * false if vector formation is non-sequential.
     *
     * @return true if preprocessing of this vector type is supported.
     * if preprocessing is not supported, {@link VectorWritable}
         accumulates
     * the vector in the usual way instead of feeding to the
         preprocessor.
     */
    boolean beginVector ( boolean sequential ) throws IOException;

    /**
     * called when next vector element becomes available.
     * @param index
     * @param value
     */
    void onElement ( int index, double value ) throws IOException;

    /**
     * called for named vectors if vector name is available.
     *
     * @param name
     */
    void onVectorName ( String name ) throws IOException ;
    /**
     * called after last element is processed.
     */
    void endVector () throws IOException;

}
```

The algorithmic change in Q-Job:

1. A row of $\mathbf{A}$ is never formed. Instead, vector preprocessor is set up in the mapper and is accumulating $k+p$ dot products of incoming elements of $\mathbf{A}$. Thus, as soon as all elements of row $\mathbf{A}$ are consumed, the preprocessor holds row of $\mathbf{Y}$. Preprocessor supports both sequential and non-sequential preprocessing.

The algorithmic change in $\mathbf{B}^\top$-Job:

1. A row of $\mathbf{A}$ is never formed. Instead, vector preprocessor is set up in the mapper.
2. Partial $\mathbf{B}^\top$ products $1 \times k + p$ are formed and output during preprocessing.
3. $\mathbf{Q}_{i,*}$ row is requested from preprocessor and $\hat{\mathbf{Q}}_j$ blocks are computed on the fly when current block is exhausted as a part of such request from preprocessor.

3 Enabling wider matrices I/O optimizations.

Branch: ssvd-wide. This branch enables processing wider matrices without having to increase `--minSplitSize` parameter. This is achieved by adding a reducer to Q-Job. The first Q pass is pushed down to reducers and mappers now only preprocess **Y** rows. What it acheives is that mappers now will not fail if they can't aquire at least $k + p$ rows of **A**. Most importantly, it defers "supersplits" problem until up to ~8 (16) mln dense items in incoming A vectors assuming 64M(128M) HDFS blocks.

It alleviates problem defined in § 6.1 of the "working notes" document.

`-w, --wide <true|false>` enable first **QR** pass pushdown to reducers. **This will not be more efficient** unless matrix vectors significantly exceed ~10-50k non-zero elements. Moreover, it will be less efficient then.

4 Enabling taller matrices I/O optimizations

It solves problem defined in § 6.3 of the "working notes" document.