Item 111
git_comments:

1. Evaluate the model. TODO: Create an evaluator to compute RMSE.
2. * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
3. Inspect false positives.
4. * * An example app for ALS on MovieLens data (http://grouplens.org/datasets/movielens/). * Run with * {{{ * bin/run-example ml.MovieLensALS * }}}

git_commits:

1. **summary:** [SPARK-3541][MLLIB] New ALS implementation with improved storage
   message: [SPARK-3541][MLLIB] New ALS implementation with improved storage This PR adds a new ALS implementation to `spark.ml` using the pipeline API, which should be able to scale to billions of ratings. Compared with the ALS under `spark.mllib`, the new implementation 1. uses the same algorithm, 2. uses float type for ratings, 3. uses primitive arrays to avoid GC, 4. sorts and compresses ratings on each block so that we can solve least squares subproblems one by one using only one normal equation instance. The following figure shows performance comparison on copies of the Amazon Reviews dataset using a 16-node (m3.2xlarge) EC2 cluster (the same setup as in http://databricks.com/blog/2014/07/23/scalable-collaborative-filtering-with-spark-mllib.html): ![als-wip](https://cloud.githubusercontent.com/assets/829644/5659447/4c4ff8e0-96c7-11e4-87a9-73c1c63d07f3.png) I keep the `spark.mllib`'s ALS untouched for easy comparison. If the new implementation works well, I'm going to match the features of the ALS under `spark.mllib` and then make it a wrapper of the new implementation, in a separate PR. TODO: - [X] Add unit tests for implicit preferences. Author: Xiangrui Meng <meng@databricks.com> Closes #3720 from mengxr/SPARK-3541 and squashes the following commits: 1b9e852 [Xiangrui Meng] fix compile 5129be9 [Xiangrui Meng] Merge remote-tracking branch 'apache/master' into SPARK-3541 dd0d0e8 [Xiangrui Meng] simplify test code c627de3 [Xiangrui Meng] add tests for implicit feedback b84f41c [Xiangrui Meng] address comments a76da7b [Xiangrui Meng] update ALS tests 2a8deb3 [Xiangrui Meng] add some ALS tests 857e876 [Xiangrui Meng] add tests for rating block and encoded block d3c1ac4 [Xiangrui Meng] rename some classes for better code readability add more doc and comments 213d163 [Xiangrui Meng] org imports 771baf3 [Xiangrui Meng] chol doc update ca9ad9d [Xiangrui Meng] add unit tests for chol b4fd17c [Xiangrui Meng] add unit tests for NormalEquation d0f99d3 [Xiangrui Meng] add tests for LocalIndexEncoder 80b8e61 [Xiangrui Meng] fix imports 4937fd4 [Xiangrui Meng] update ALS example 56c253c [Xiangrui Meng] rename product to item bce8692 [Xiangrui Meng] doc for parameters and project the output columns 3f2d81a [Xiangrui Meng] add doc 1efaecf [Xiangrui Meng] add example code 8ae86b5 [Xiangrui Meng] add a working copy of the new ALS implementation

github_issues:

github_issues_comments:

github_pulls:

1. **title:** [SPARK-3541][MLLIB] New ALS implementation with improved storage
   body: This PR adds a new ALS implementation to `spark.ml` using the pipeline API, which should be able to scale to billions of ratings. Compared with the ALS under `spark.mllib`, the new implementation 1. uses the same algorithm, 2. uses float type for ratings, 3. uses primitive arrays to avoid GC, 4. sorts and compresses ratings on each block so that we can solve least squares subproblems one by one using only one normal equation instance. The following figure shows performance comparison on copies of the Amazon Reviews dataset using a 16-node (m3.2xlarge) EC2 cluster (the same setup as in http://databricks.com/blog/2014/07/23/scalable-collaborative-filtering-with-spark-mllib.html): ![als-wip]

(https://cloud.githubusercontent.com/assets/829644/5659447/4c4ff8e0-96c7-11e4-87a9-73c1c63d07f3.png) I keep the `spark.mllib`'s ALS untouched for easy comparison. If the new implementation works well, I'm going to match the features of the ALS under `spark.mllib` and then make it a wrapper of the new implementation, in a separate PR. TODO: - [X] Add unit tests for implicit preferences.

github_pulls_comments:

1. [Test build #24537 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/24537/consoleFull) for PR 3720 at commit [`3f2d81a`] (https://github.com/apache/spark/commit/3f2d81aae68e4a9dd095183c6fed4622a0fb0015). - This patch **passes all tests**. - This patch merges cleanly. - This patch adds the following public classes _(experimental)_: - `case class Rating(user: Int, product: Int, rating: Float)` - `case class Params(` - `class ALS extends Estimator[ALSModel] with ALSParams` - `trait ParquetTest`
2. [Test build #24653 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/24653/consoleFull) for PR 3720 at commit [`4937fd4`] (https://github.com/apache/spark/commit/4937fd45c07253c181371ca8d3ab4a582c3ce852). - This patch **passes all tests**. - This patch merges cleanly. - This patch adds the following public classes _(experimental)_: - `case class Rating(userId: Int, movieId: Int, rating: Float, timestamp: Long)` - `case class Movie(movieId: Int, title: String, genres: Seq[String])` - `case class Params(` - `class ALS extends Estimator[ALSModel] with ALSParams`
3. [Test build #24911 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/24911/consoleFull) for PR 3720 at commit [`213d163`] (https://github.com/apache/spark/commit/213d1636558051e454a748b6f9bab6f4bf0a9d2d). - This patch **passes all tests**. - This patch merges cleanly. - This patch adds the following public classes _(experimental)_: - `case class Rating(userId: Int, movieId: Int, rating: Float, timestamp: Long)` - `case class Movie(movieId: Int, title: String, genres: Seq[String])` - `case class Params(` - `class ALS extends Estimator[ALSModel] with ALSParams`
4. [Test build #25195 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/25195/consoleFull) for PR 3720 at commit [`2a8deb3`] (https://github.com/apache/spark/commit/2a8deb34e926908ddfa528e8e59c278eced3cf5a). - This patch **passes all tests**. - This patch merges cleanly. - This patch adds the following public classes _(experimental)_: - `case class Rating(userId: Int, movieId: Int, rating: Float, timestamp: Long)` - `case class Movie(movieId: Int, title: String, genres: Seq[String])` - `case class Params(` - `class ALS extends Estimator[ALSModel] with ALSParams` - `case class RatingBlock(srcIds: Array[Int], dstIds: Array[Int], ratings: Array[Float])`
5. @srowen @coderxiang This PR is almost ready, pending few unit tests. Would you be interested in reviewing the code?
6. [Test build #25207 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/25207/consoleFull) for PR 3720 at commit [`a76da7b`] (https://github.com/apache/spark/commit/a76da7bb4d3a3073999725f8c10c5c2443fd7839). - This patch **fails Spark unit tests**. - This patch merges cleanly. - This patch adds the following public classes _(experimental)_: - `case class Rating(userId: Int, movieId: Int, rating: Float, timestamp: Long)` - `case class Movie(movieId: Int, title: String, genres: Seq[String])` - `case class Params(` - `class ALS extends Estimator[ALSModel] with ALSParams` - `case class RatingBlock(srcIds: Array[Int], dstIds: Array[Int], ratings: Array[Float])`
7. [Test build #25337 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/25337/consoleFull) for PR 3720 at commit [`b84f41c`] (https://github.com/apache/spark/commit/b84f41cf9e4d9ac87f819cfa9222575b6bd44d8d). - This patch **passes all tests**. - This patch merges cleanly. - This patch adds the following public classes _(experimental)_: - `case class Rating(userId: Int, movieId: Int, rating: Float, timestamp: Long)` - `case class Movie(movieId: Int, title: String, genres: Seq[String])` - `case class Params(` - `class ALS extends

Estimator[ALSModel] with ALSParams` - `case class RatingBlock(srcIds: Array[Int], dstIds: Array[Int], ratings: Array[Float])`

8. [Test build #25353 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/25353/consoleFull) for PR 3720 at commit [`dd0d0e8`] (https://github.com/apache/spark/commit/dd0d0e8ecd36b9e607306dd170d1e22437180389). - This patch **fails Spark unit tests**. - This patch merges cleanly. - This patch adds the following public classes _(experimental)_: - `case class Rating(userId: Int, movieId: Int, rating: Float, timestamp: Long)` - `case class Movie(movieId: Int, title: String, genres: Seq[String])` - `case class Params(` - `class ALS extends Estimator[ALSModel] with ALSParams` - `case class RatingBlock(srcIds: Array[Int], dstIds: Array[Int], ratings: Array[Float])`
9. test this please
10. [Test build #25485 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/25485/consoleFull) for PR 3720 at commit [`dd0d0e8`] (https://github.com/apache/spark/commit/dd0d0e8ecd36b9e607306dd170d1e22437180389). - This patch **fails Spark unit tests**. - This patch merges cleanly. - This patch adds the following public classes _(experimental)_: - `case class Rating(userId: Int, movieId: Int, rating: Float, timestamp: Long)` - `case class Movie(movieId: Int, title: String, genres: Seq[String])` - `case class Params(` - `class ALS extends Estimator[ALSModel] with ALSParams` - `case class RatingBlock(srcIds: Array[Int], dstIds: Array[Int], ratings: Array[Float])`
11. test this please
12. @srowen @coderxiang T on the implementation?
13. [Test build #25765 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/25765/consoleFull) for PR 3720 at commit [`dd0d0e8`] (https://github.com/apache/spark/commit/dd0d0e8ecd36b9e607306dd170d1e22437180389). - This patch **fails to build**. - This patch merges cleanly. - This patch adds the following public classes _(experimental)_: - `case class Rating(userId: Int, movieId: Int, rating: Float, timestamp: Long)` - `case class Movie(movieId: Int, title: String, genres: Seq[String])` - `case class Params(` - `class ALS extends Estimator[ALSModel] with ALSParams` - `case class RatingBlock(srcIds: Array[Int], dstIds: Array[Int], ratings: Array[Float])`
14. [Test build #25767 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/25767/consoleFull) for PR 3720 at commit [`1b9e852`] (https://github.com/apache/spark/commit/1b9e852748b4e6bcee415a1e9317c218e7a823b9). - This patch **passes all tests**. - This patch merges cleanly. - This patch adds the following public classes _(experimental)_: - `case class Rating(userId: Int, movieId: Int, rating: Float, timestamp: Long)` - `case class Movie(movieId: Int, title: String, genres: Seq[String])` - `case class Params(` - `class ALS extends Estimator[ALSModel] with ALSParams` - `case class RatingBlock(srcIds: Array[Int], dstIds: Array[Int], ratings: Array[Float])`
15. @srowen @coderxiang Do you have more comments? I'm thinking about merging this and then port nonnegative support. After that, we can replace the ALS implementation under "spark.mllib".
16. @mengxr Given how familiar you are with this implementation, and the tests, I can only be pretty sure it works. I didn't see any style issues, and thought through some of the loops for speed/correctness but every spot check was fine. It is looking good to me.
17. @mengxr the logic and style also look good to me.
18. Thanks! I've merged this into master. I'll send follow-up PRs soon.
19. hi, we use als example, but we found same data, same input arguments, but output different result, spark 1.2.0 can result data we want, but spark 1.3.0 cannot return right data, it all return zero matrix in userFeature.

github_pulls_reviews:

1. Tiny: would it be clearer to return `Some` and `None`? This works too of course.
2. Tiny: I see that the Scala source usually renames this to `ju`. Shorter, not sure if it's clearer, but I'm used to reading `ju.Map`, etc.
3. I think you're still implementing the 'extension' here that supports `r < 0` so you could say "0 if r <= 0".
4. can this and the previous line be merged?

5. What is the benefit of using `view` here?
6. I assume it's to materialize the result lazily. Otherwise it creates the entire result in memory at once.
7. Done.
8. Done.
9. Done. I also copied the comments to `NormalEquation.addImplicit`.
10. `size` is a field of `InBlock`.
11. not sure if it matters much here, but zipWithIndex is probably much slower than using a while loop.
12. It doesn't matter much here, because this is not performance critical. It is inside a `flatMap`, using a while loop may need some extra lines of code.

jira_issues:

jira_issues_comments: