ALS-WR for preferences with confidence

Dmitriy Lyubimov*

Method

Notation. element of vector vs. i-th vector in a vector set:

$\mathbf{u}^{(k)}$ k-th vector in implied set of vectors

$\mathbf{u}_i$ i-th element of vector $\mathbf{u}$

Similarly for matrices.

User subscript is denoted by u and item's subscript is denoted by i.

Problem 1. (User-Item preferences): Given:

- user-item preference $p_{u,i} \in \{0, 1\}$, total matrix $\mathbf{P} \in \mathbb{R}^{m \times n}$ (m is number of users, n is number of items)
- user-item confidence matrix $\mathbf{C}$ consists of some weights $c_{u,i}$.

We are solving matrix completion for $\mathbf{P} \approx \mathbf{U}^\top \mathbf{V}$ by finding $\mathbf{U} \in \mathbb{R}^{m \times k}$ and $\mathbf{V} \in \mathbb{R}^{n \times k}$.■

Solution. Denote row-vector $\mathbf{u}^{(u)} = \mathbf{U}^\top_{u,*}$, $\mathbf{v}^{(i)} = \mathbf{V}^\top_{i,*}$, $\mathbf{p}^{(u)} = \mathbf{P}^\top_{u,*}$, $\mathbf{p}'^{(i)} = \mathbf{P}_{*,i}$

Preference predictor

$$\hat{p}^{(u,i)} = \mathbf{u}^{(u)\top}\mathbf{v}^{(i)}, \tag{1}$$

or simply the whole prediction matrix

$$\hat{\mathbf{P}} = \mathbf{U}\mathbf{V}^\top \tag{2}$$

Minimizing cost with L2 reguralization

$$\sum_{u,i}\left[c_{u,i}\left(p_{ui} - \mathbf{u}^{(u)\top}\mathbf{v}^{(i)}\right)^2\right] + \\ + \lambda\left(\sum_u n_u \left\|\mathbf{u}^{(u)}\right\|^2 + \sum_i n'_i \left\|\mathbf{v}^{(i)}\right\|^2\right) \tag{3}$$

via alternating updates:

$\forall u \in 1..m \Rightarrow$

$$\mathbf{u}_u = \left(\mathbf{V}^\top \mathbf{D}^{(u)} \mathbf{V} + n\lambda \mathbf{I}\right)^{-1} \mathbf{V}^\top \mathbf{D}^{(u)} \mathbf{p}^{(u)} \tag{4}$$

where $\mathbf{D}^{(u)}$ is $n \times n$ diagonal matrix such that $\mathbf{D}^{(u)}_{i,i} = c_{u,i}$ and n_u is the number of measurements for user u such that $c_{ui} \neq 0$;

and $\forall i \in 1..n \Rightarrow$

$$\mathbf{v}_i = \left(\mathbf{U}^\top \mathbf{D}^{(i)} \mathbf{U} + n_i\lambda \mathbf{I}\right)^{-1} \mathbf{U}^\top \mathbf{D}^{(i)} \mathbf{p}^{(i)} \tag{5}$$

where $\mathbf{D}^{(i)}$ is $m \times m$ diagonal matrix such that $\mathbf{D}^{(i)}_{u,u} = c_{u,i}$ and n_i is the number of measurements for item i such that $c_{ui} \neq 0$.

The algorithm seeds $\mathbf{U}$ and $\mathbf{V}$ with small random numbers out of say $U(0, 0.01) - 0.005$ and then sequentially applies updates (4) and (5) to them until convergence. ■

Another thing to notice is that perhaps $n_u \ll n$ and $n'_i \ll m$ for many u and i and and therefore the equations can be efficiently blockified to throw away rows for which there's no observation,

*dlyubimov at apache dot org

i.e. $c_{u,i} = 0$. Hence, dimensionality of these equations will typically be significantly reduced (as defined in [Zhou-1])

Minimum confidence in unobserved preference. Variation presented in [Koren-1] mostly differs in that it never assumes $c_{u,i} = 0$ (in fact, it assumes $c_{u,i} = 1$ and $p_{u,i} = 0$ whenever there's no observation at all to reflect the fact that there's a standardized tiny belief that no observation may mean lack of interest of the user in an item). This approach can be made computationally equivalent to the sparse observations (see details in the [Koren-1]).

To generalize, we will assume minimum confidence for any directly unobserved preference p_0 being some constant c_0 and increment any actual observed confidence with it:

$$c_{u,i} = c_0 + c^*_{u,i}.$$

Thus, we can rewrite original problem as having input of $\mathbf{C}^*$ and c_0 to keep confidence input sparse (implying actual confidence matrix by $\mathbf{C}_{u,i} = c_0 + \mathbf{C}^*_{u,i}$).

We construct diagonal matrix $\mathbf{D}^{*(u)}$ similarly to construction of $\mathbf{D}^{(u)}$, but using $\mathbf{C}^*$ as a source. Obviously, since users of $\mathbf{C}^*$ will have very few nonzero elements (in fact, only those items that were ever attempted to be consumed will have any, which in real world normally is just a fraction of all available items), the diagonal of $\mathbf{D}^*$ can be expected to be super sparse.

By construction, $\mathbf{D}^{(u)} = \mathbf{D}^{*(u)} + c_0\mathbf{I}$. Subsituting $\mathbf{D}^{(u)}$ into (4) and (5), we get

$$\mathbf{u}_u = \left(c_0\mathbf{V}^\top\mathbf{V} + \mathbf{V}^\top\mathbf{D}^{*(u)}\mathbf{V} + n_u\lambda\mathbf{I}\right)^{-1} \times$$

$$\times\, \mathbf{V}^\top \left(c_0\mathbf{I} + \mathbf{D}^{*(u)}\right)\mathbf{p}^{(u)} \qquad (6)$$

$$\forall u \in 1..m; \text{ and}$$

$$\mathbf{v}_i = \left(c_0\mathbf{U}^\top\mathbf{U} + \mathbf{U}^\top\mathbf{D}^{*(i)}\mathbf{U} + n_i\lambda\mathbf{I}\right)^{-1} \times$$

$$\times\, \mathbf{U}^\top \left(c_0\mathbf{I} + \mathbf{D}^{*(i)}\right)\mathbf{p}^{(i)} \qquad (7)$$

$$\forall i \in 1..n.$$

Here, $c_0\mathbf{V}^\top\mathbf{V}$ is precomputed once for all u per iteration; product $\mathbf{V}^\top\mathbf{D}^{*(u)}\mathbf{V}$ is sparse because diagonal of $\mathbf{D}^{*(u)}$ is sparse; and $\mathbf{p}^{(u)}$ keeps the part $\mathbf{V}^\top\left(c_0\mathbf{I} + \mathbf{D}^{*(u)}\right)\mathbf{p}^{(u)}$ still sparse (as we usually construct our problem by assuming preference $\mathbf{P}_{u,i} = 0$ for any interaction we don't have an observation for).

Note 1. c_0 usually does not require crossvalidation; instead, parameters that consruct $c^*_{u,i} \neq 0$, do (e.g. see [Koren-1] for an example of confidence encoding scheme for a fraction of a consumed item). In reality it is quite possible that more than one parameter constructing confidence would require search by crossvalidation. Therefore, it is totally ok to assume $c_0 = 1$ in majority of cases and focus on search for other confidence encoding parameters instead.

Note 2. And corollary to note 1, λ is the only parameter of the training api itself that should be subject to crossvalidation.

Note 3. The equations (6) and (7) represent weighted regularization example. I failed to fiind any theoretical justification for weighted regularization in [Zhou-1] , only a statement of better results based on empirical evidence obtained by the authors. Anyhow, non-WR option of the training is easily achieved by dropping n_i and n_u factors from (6), (7).

Implementation working notes.

Overview

Input encoding. We don't consider any particular feature encoding scheme from ratings r into (p, c) (preference/confidence) pairs as given in [Koren-1], directly. Instead, we leave it to the user. Crossvalidation of preference and confidence encoding scheme parameters is left outside of MAHOUT-1365 scope as well.

Formally, we consider input to be two matrices – preferences $\mathbf{P}$ and confidence of observations $\mathbf{C}$.

$\mathbf{P} \in \{0, 1\}^{m \times n}$ is already sparse.

$\mathbf{C} \in \mathbb{R}^{m \times n}$ in its pristine form is dense, since we assume that entries corresponding to no observations contain some baseline non-negative confidence c_0. We also assume that any matrix entry where we did observe an interaction, we have confidence greater than baseline (i.e. $\mathbf{C}_{i,j} \geq c_o \, \forall i, j$).

We assume the following encoding of input $\mathbf{A}$ elements:

$$a_{ij} = \begin{cases} c_{ij} - c_0, & p_{ij} = 1.0; \\ c_0 - c_{ij}, & p_{ij} = 0. \end{cases}$$

Thus, input $\mathbf{A} \in \mathbb{R}^{m \times n}$ captures both matrices, $\mathbf{P}$ and $\mathbf{C}$, in one matrix; and that matrix is also sparse, since all non-observation entries ($p = 0$, $c = c_0$) will evaluate to 0. Here, we assume that input ($p = 1$, $c = c_0$) is never occuring (i.e we always consider no-observation to correspond to no-preference by default, albeit with least confidence).

Commonalities. In both in-core and distributed versions, we first preallocate dense matrices $\mathbf{U}$ and $\mathbf{V}$ and initialize $\mathbf{V}$ with small numbers.

Subsequently, we use alternating updates to $\mathbf{U}$ and $\mathbf{V}$ per formulas 6 and 7. Note that there, component $c_0 \mathbf{V}^\top \mathbf{V}$ is the same for all updates of rows of $\mathbf{U}$. Consequently, it is computed once for all updates.

We proceed doing alternating updates until we either achieve maximum iterations parameter, or convergence test (if requested) doesn't yield any significant improvement. Convergence test condition looks like

$$\frac{\text{rmse}^{(i-1)} - \text{rmse}^{(i)}}{\text{rmse}^{(i-1)}} \leq \text{convergenceThreshold},$$

i.e. parameter 'convergenceThreshold' is minimum acceptable improvement fraction in rmse so that we can consider further iterations to be a bona-fide investment.

In-core implicit ALS details

Alternating row updates of $\mathbf{U}$ and $\mathbf{V}$ are symmetrical and are verbatim encoding of the formulas (6) and (7) which in Mahout's DSL looks like the following (in case of updating $\mathbf{U}_{i,*}$):

```
inCoreU(i, ::) = solve(
        a = c0vtv + (inCoreV.t %*%: diagv(inCoreD(i, ::))) %*%
            inCoreV + diag(n_u * lambda, k),
        b = (inCoreV.t %*%: diagv(d_i + c0)) %*% p_i
      )
```

Distributed implicit ALS details

We are constructing two matrices in the form$(\mathbf{U}|\mathbf{A}) \triangleq \text{cbind}\,(\mathbf{U}, \mathbf{A})$ and $\left(\mathbf{V}|\mathbf{A}^\top\right) \triangleq \text{cbind}\left(\mathbf{V}, \mathbf{A}^\top\right)$. This approach requires us to impose Int-keyed restriction on input A. Additionally, $\mathbf{V}$ is initialized with small random numbers just like in the in-core version while it is being built side-by-side.

The update (of say $\mathbf{U}$ part of $(\mathbf{U}|\mathbf{A})$) proceeds as following.

Component $c_0\mathbf{V}^\top\mathbf{V}$ is computed and broadcasted trivially using Mahout DRM DSL:

```
val c0vtvBcast = drmBroadcast((drmV.t %*% drmV).collect * c0)
```

Next, we start sending data from $\left(\mathbf{V}|\mathbf{A}^\top\right)$ to $(\mathbf{U}|\mathbf{A})$ and apply updates to the $\mathbf{U}$ part of the $(\mathbf{U}|\mathbf{A})$.

The idea is to send every row $\mathbf{V}_{i,*}$ to rows u of $(\mathbf{U}|\mathbf{A})$ for which either $\mathbf{P}_{u,i} = 1.0$ or $\mathbf{C}_{u,i} > c_o$. By the way of encoding of $\mathbf{A}$, it is always true iff $a_{u,i} \neq 0$. Since rows of $\mathbf{V}$ and $\mathbf{A}^\top$ are collocated in $\left(\mathbf{V}|\mathbf{A}^\top\right)$, this provides enough information to generate destination indices u for any $\mathbf{v}_i$ payload. Payload of the messages to rows of $(\mathbf{U}|\mathbf{A})$ is formatted as $(i \to \mathbf{v}_i)$ pairs. We will call a set of i's arriving at u-th row of $(\mathbf{U}|\mathbf{A})$ as $S^{(u)}$. By construction, therefore, $S^{(u)}$ are indices i such that $a_{u,i} \neq 0$.

Once messages with $(i \to \mathbf{v}_i)$ payloads arrive at u-th row of $(\mathbf{U}|\mathbf{A})$, they are used to compile two most difficult terms in 6: (a) $\mathbf{V}^\top\mathbf{D}^{*(u)}\mathbf{V}$ and (b) $\mathbf{V}^\top\left(c_0\mathbf{I} + \mathbf{D}^{*(u)}\right)\mathbf{p}^{(u)}$.

• For computing term (a), we notice that diagonal of $\mathbf{D}^{*(u)}$ is readily available as $|a_{u,i}|$ which we already have collocated in the destination row of $(\mathbf{U}|\mathbf{A})$. In other words, we iteratively compute the term $\mathbf{V}^\top\mathbf{D}^{*(u)}\mathbf{V}$ as

$$\mathbf{V}^\top\mathbf{D}^{*(u)}\mathbf{V} = \sum_{i\in S^{(u)}} a_{u,i}\mathbf{v}_i\mathbf{v}_i^\top. \tag{8}$$

Since all necessary $\mathbf{v}_i$ such that $i \in S^{(u)}$are in the message stream sent to u-th row of $(\mathbf{U}|\mathbf{A})$, we have all necessary information to compute (8). We preallocate dense $k \times k$ accumulator to add $(a_{u,i}\mathbf{v}_i)\,\mathbf{v}_i^\top$ for every new message $\mathbf{v}_i$ seen in the incoming message sequence. At the end of message processing loop the accumulator will contain the result sought.■

• For computing term (b), we notice that $\mathbf{p}_i^{(u)} = 1.0$ whenever $\mathbf{A}_{u,i} > 0$. Again, row $\mathbf{A}_{u,*}$ is conveniently collocated in the destination matrix $(\mathbf{U}|\mathbf{A})$. Let denote set of indices i such that $a_{u,i} > 0$ as $S'^{(u)}$.

Therefore, the term $\mathbf{V}^\top\left(c_0\mathbf{I} + \mathbf{D}^{*(u)}\right)\mathbf{p}^{(u)}$ can be computed simply as

$$\mathbf{V}^\top\left(c_0\mathbf{I} + \mathbf{D}^{*(u)}\right)\mathbf{p}^{(u)} = \sum_{i\in S'^{(u)}} (c_0 + a_{u,i})\,\mathbf{v}_i. \tag{9}$$

Since, by construction, $S'^{(u)} \subseteq S^{(u)}$, all necessary $\mathbf{v}_i$ to compute term (9) are also in the incoming stream of $\mathbf{v}_i$ (we just need to apply $i \in S'^{(u)}$ filter while accumulating the term from incoming messages).■

Updating $\mathbf{V}$ part of the $\left(\mathbf{V}|\mathbf{A}^\top\right)$ matrix is a symmetric reflection of the process above.

References

[Koren-1] Y. Koren, et.al. Collaborative Filtering for Implicit Feedback Datasets

[Zhou-1] Y. Zhou, et. al. Large-scale Parallel Collaborative Filtering for the Netflix Prize