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 $c_{u,i}$. This can be sparse where confidence is 0

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

*dlyubimov at apache dot org

throw away rows for which there's no observation, 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).

Spark mapping notes.

Precompute steps.

- $\mathbf{U}$ and $\mathbf{V}$ initialized with random small numbers and presented as Spark-based DRM.
- Next, every row in $\mathbf{U}$ ($\mathbf{V}$) is connected with corresponded sparse vectors $\mathbf{c}^{(u)} = \mathbf{C}^*_{u,*}$ ($\mathbf{c}^{(i)} = \mathbf{C}^*_{*,i}$) and $\mathbf{p}^{(u)} = \mathbf{P}_{u,*}$ ($\mathbf{p}^{(i)} = \mathbf{P}_{*,i}$). So what we have in the end are two spark RDDs of Bagel vertices containing (id=$u|i$, $\mathbf{U}_{u,*}|\mathbf{V}_{*,i}$, $\mathbf{c}^{(u)}|\mathbf{c}^{(i)}$, $\mathbf{p}^{(u)}|\mathbf{p}^{(i)}$).

Iteration steps.

- $\mathbf{V}^\top\mathbf{V}$ precomputed in distributed way for each iteration and broadcast. Since this matrix has tiny $k \times k$ geometry, collecting and broadcasting it should not be a problem.
- Form Bagel messages from current $\mathbf{V}$ side : each vector $\mathbf{V}_i$ is sent to two sets of user vertices: (1) users for which $c_{u,i} \neq 0$; (2) users for which $p_{u,i} \neq 0$. (obviously if group 1 and group 2 share a vertex, no need to send row of $\mathbf{V}$ twice to that vertex).
- During Bagel's compute() it can be seen that rows sent as group (1) is essential to compute $\mathbf{V}^\top\mathbf{D}^{*(u)}\mathbf{V}$; group (2) is essential to compute $\mathbf{V}^\top\mathbf{p}^{(u)}$. Update $\mathbf{U}_{u,*}$ per (6).
  - Use Cholesky to solve $\mathbf{AX} = \mathbf{B}$ by applying $\mathbf{A} = \mathbf{LL}^\top \Rightarrow \mathbf{X} = \left(\mathbf{L}^\top\right)^{-1}\mathbf{L}^{-1}\mathbf{B}$ which is `chol(A).solveRight(I) %*% chol(A).solveRight(B)`.

Iteration updating $\mathbf{V}$ is symmetrical.

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