libFM - Manual for the BPR extension.

Fabio Petroni*

2015

Contents

*http://www.fabiopetroni.com

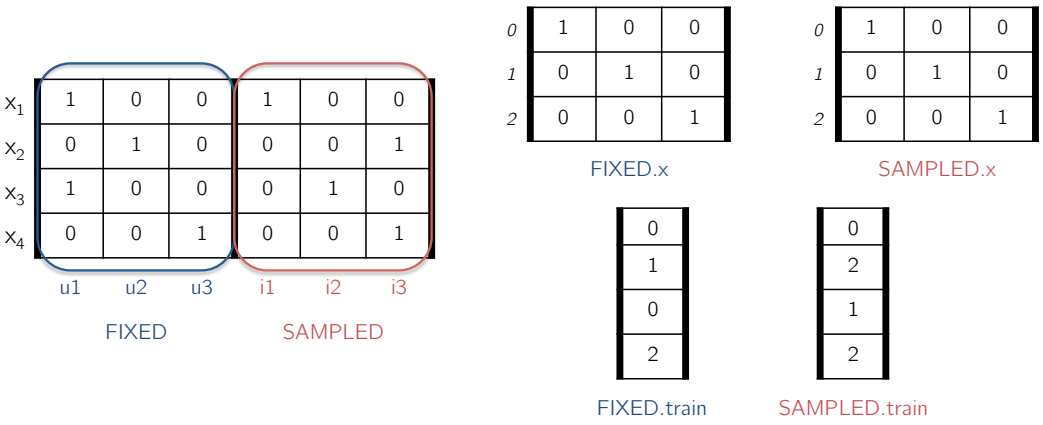

Figure 1: Example of data representation.

1 Introduction

This manual is an extension of the main libFM 1.4.2 manual[1]. In particular it aims at describing the new functionality we added to the original libFM library, that is the support for the BPR optimization criterion. Moreover, we implement the training procedure for BPR in a parallel fashion, using a parallel lock-free version of stochastic gradient ascent. This allows to handle (reasonably) large datasets.

You can find more information, as well as a complete application scenario, in the following publication:

Fabio Petroni, Luciano Del Corro and Rainer Gemulla (2015): CORE: Context–Aware Open Relation Extraction with Factorization Machines. In Empirical Methods in Natural Language Processing (EMNLP 2015)

If you use this software please cite the paper.

2 BPR Data Format

This extension uses the Block Structure data format (extensively described in Chapter 4 of the main libFM 1.4.2 manual). In particular, the software requires in input two blocks, respectively referred to as FIXED block and SAMPLED block.

[1] http://www.libfm.org/libfm-1.42.manual.pdf

Figure 1 shows a small example, where the two blocks contain just a single group of column. For instance, in the Recommender Systems field, the FIXED block can be associated with the users and the SAMPLED block with the items. As already described in the original libFM manual, these blocks can be represented using several files:

<**blockname**>**.x** The design matrix of the block. Differently from libFM 1.4.2 now it can be expressed both in binary format and in text format.

<**blockname**>**.groups** Optional file for grouping predictor variables.

<**blockname**>**.train** The mapping from train rows to block rows. The expected file is in text format with as many lines as the training dataset (in the –train parameter).

<**blockname**>**.test** The mapping from test rows to block rows. The expected file is in text format with as many lines as the testing dataset (in the –test parameter).

<**blockname**>**.validation** The mapping from validation rows to block rows. The expected file is in text format with as many lines as the validation dataset (in the –validation parameter).

BPR is a pairwise approach. In each stochastic update, a positive x^+ (i.e. observed) and a negative x^- (i.e. not observed) row are simultaneously analyzed by the algorithm, whose scope is make the score achieved by x^+ bigger than the score achieved by x^-. This reflect the idea that an observed fact is more probable than an unobserved one. For example, let's assume that the algorithm is analyzing the first row in Figure 1, that become the current positive row (e.g. it can be interpreted as saying that user u_1 bought item i_1). The negative row is sampled by keeping fixed the FIXED block (i.e. user u_1), and sampling a row for the SAMPLED block that is never observed in correspondence with the FIXED row. In this example the algorithm will sample the third row of SAMPLED.x, since both the first and the second have been observed with the first row of FIXED.x (e.g. user u_1 bought both items i_1 and i_2, but not i_3; the combination u_1 i_3 is sampled as negative evidence).

3 LibFM with BPR

We added (or modified) the following options to libFM:

```
--method                learning method (SGD, SGDA, ALS, MCMC,
                        BPR, BPRA); default=MCMC
--neg_sample            number of the negative pair samples
                        drawn for each training observation,
                        default 1 (only for bpr or bpra)
--out_conv              filename for output the convergence info
                         (only for bpr or bpra)
--threads               number of threads (only for bpr or bpra)
--top_k                 number of recommendation for bpr or bpra
                        , default 100
--list_id_output        list of target ids (in FIXED block),
                        comma separated without spaces, to
                        compute output ranked list for, default
                        all (only for bpr or bpra)
--out_ranked_list_dir   directory where to store the output
                        ranked list, one file for each target id
                         (only for bpr or bpra)
```

In order to execute libFM with the BPR procedure is mandatory to specify the two blocks (i.e. FIXED and SAMPLED, in this order) using the -relation option.

3.1 Eaxmples

In this section several examples are provided, using as input data the example in Figure 1. You can find all the files in a zip archive together with this manual (or, alternatively, here `http://www.fabiopetroni.com/Download/example_libFM_with_BPR_extension.zip`). Notice that the content of the files *train.libfm* and *test.libfm* is ignored by the software, but it is mandatory that they contain a number of rows equal to the files *FIXED.train* and *SAMPLED.train* (for *train.libfm*), respectively FIXED.test and SAMPLED.test (for *test.libfm*).

The following command executes libFM with BPR on 8 threads:

```
./libFM -task r -train train.libfm -test test.libfm -dim
    '1,1,100' -method bpr -relation FIXED,SAMPLED -threads 8
```

It is possible to specify a file where the software will write some convergence information, as well as how many negative rows will be sampled for a single positive row, for each stochastic update.

```
./libFM -task r -train train.libfm -test test.libfm -dim
    '1,1,100' -method bpr -relation FIXED,SAMPLED -threads 8
    -neg_sample 3 -out_conv out_convergence.dat
```

Finally, it is possible to write the output ranked list (of size k, in this example k=1) for the elements in the FIXED block. In this example the top-k recommendations are computed only for elements 0 and 2, and the software store a file for each of them in the directory specified with -out_ranked_list_dir.

```
./libFM -task r -train train.libfm -test test.libfm -dim
    '1,1,100' -method bpr -relation FIXED,SAMPLED -threads 8
    -neg_sample 3 -out_conv out_convergence.dat -top_k 1
    -out_ranked_list_dir output -list_id_output 0,2
```

4 Other features added

We added the following options to libFM:

```
--out_vectors          filename for output the latent vectors
--resume_state         files with the vectors to resume the
    state of the FM
```

These options allow to store the final state of a libFM execution (with -out_vectors) in a text file, and successively resume the computation using as initial values the state stored in the file (with -resume_state).