logitSVD

Combines the advantages of classic logit models with the SVD approach for building product recommendation systems

- Predicts the probability to buy a product or to rate the product e.g. with a certain number of stars
- Strong predictive power with a minimal number of parameter to avoid overfitting
- Avoids cold start problem through the use of user features
- Selection of features is supported by a feature importance measure (Wald test)
- Highly transparent model with interpretable parameters
- Building separate models for each product is a special case of logitSVD

A feature based SVD for binary user-item-matrices

Definition[1]

Let $R \in \{0,1,NaN\}^{n_u \times n_i}$ be a binary matrix with entries 0, 1 or missing value for n_u users and n_i items and let n_f features $X \in \mathbb{R}^{n_u \times n_f}$ for each user u be given. The probability $p_{u,i} = P(r_{u,i} = 1)$ that $r_{u,i} = 1$ can then be approximated by the binary logit SVD:

$$S = X\ E\ Q, \qquad p_{u,i} = l(s_{u,i}) = \frac{1}{1+e^{-s_{u,i}}}$$

with parameter matrices $E \in \mathbb{R}^{n_f \times n_d}$ and $\mathrm{Q} \in \mathbb{R}^{n_d \times n_i}$. n_d is a design parameter, which can be interpreted as the number of estimated models.

Illustration

X

	Constant	User Feature 1	User Feature 2
User 1	1.0	2.5	-1.0
User 2	1.0	2.0	0.5
User 3	1.0	1.0	-0.5
...	...	...	...
User n_u	1.0	1.5	1.0

*

E

	Model 1	Model 2
Intercept	2.1	-2.0
Parameter 1	1.0	1.4
Parameter 2	0.2	1.6

=

XE

4.4	-0.1
4.2	1.6
3.0	-1.4
...	...
3.8	1.7

Q

0	1.5	1	...	-0.2
1	-0.5	0	...	0.5

l(.) =

	Item 1	Item 2	Item 3	...	Item n_i
User 1	0.525	0.001	0.012	...	0.717
User 2	0.168	0.004	0.014	...	0.510
User 3	0.802	0.005	0.047	...	0.786
...	...	...	...	...	...
User n_u	0.154	0.008	0.022	...	0.478

1. We are using the matrix notation $A = (a_{i,i})_{i \le n, j \le m}$

Newton's method boosted maximum likelihood parameter estimation for binary user-item-matrices

Maximum likelihood

The parameter matrixes E and Q are estimated such the log-likelihood to observe $R \in \{0,1,NaN\}^{n_u \times n_i}$ is maximized

$$\log \mathrm{L}(\mathrm{E},\mathrm{Q}) = \sum_{u \le n_u, i \le n_i, r_{u,i} \ne NaN} \left[r_{u,i} \log\left(l(s_{u,i})\right) + (1 - r_{u,i}) \log\left(1 - l(s_{u,i})\right) \right]$$

with $\mathrm{S} = X\ E\ Q\ \in \mathbb{R}^{n_u \times n_i}$ and $l(x) = \frac{1}{1+e^{-x}}$.

Note that only known product usages, i.e. $r_{u,i} \ne NaN$, are considered in the likelihood.

Full Newton's and alternating Newton's method

The logitVD package offers two options for solving the non-linear likelihood optimization problem:

- straight Newton's method with line search to simultaneously calculate E and Q and
- alternating Newton's methods with line search to iteratively optimize E with fixed Q and subsequently optimize Q with fixed E

For most problems the alternating approach is quicker and more stable. It can be shown that each of the alternating optimization problems (i.e. for E with fixed Q and $Q$with fixed E) is convex with s.p.d. Hessian.

A feature based SVD for user-item-matrices with several ordered classes

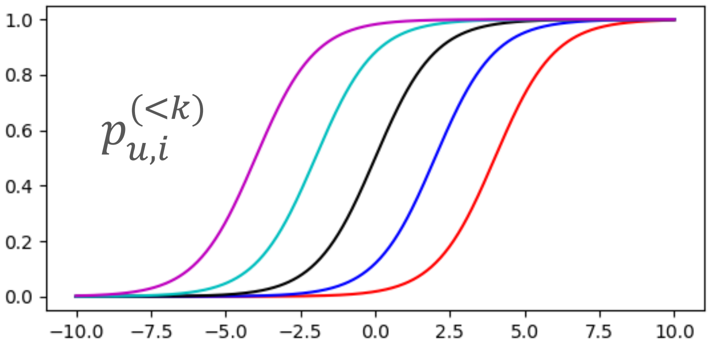
Definition[1]

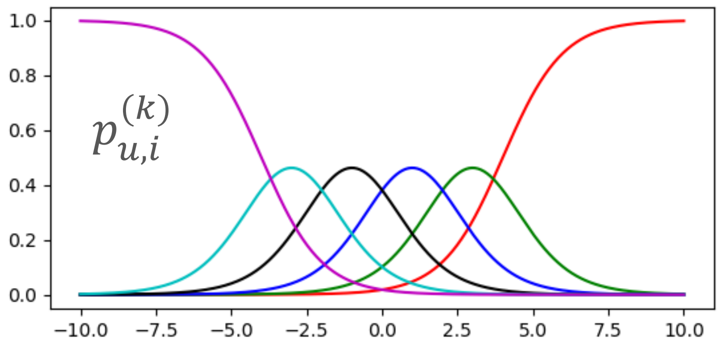
Let $R \in \{0,1,..,n_k, NaN\}^{n_u \times n_i}$ be the user-item-matrix describing the rating given by user u to item i .The ratings $0 \le r_{u,i} \le n_k$ are assumed to be ordered, i.e. all ratings $r_{u,i}$ with $r_{u,i} < r_{v,j}$ are better/worse than the ratings $r_{v,j}$. The probability $p_{u,i}^{(<k)} = P(r_{u,i} < k)$ that the rating $r_{u,i}$ is smaller than k can then be approximated by the feature based SVD:

$$S = X\ E\ Q, \qquad p_{u,i}^{(<k)} = l(s_{u,i} - t_k) = \frac{1}{1+e^{-s_{u,i}-t_k}}$$

with parameter matrices $E \in \mathbb{R}^{n_f \times n_d}$ and $\mathrm{Q} \in \mathbb{R}^{n_d \times n_i}$ and intercepts $t_k \in \mathbb{R}$ for $1 \le k \le n_k$ and $t_0 = -\infty$, $t_{n_k+1} = \infty$. The probability $p_{u,i}^{(k)}$ that item i is rated k by user u rating $r_{u,i}$is then $p_{u,i}^{(k)} = P(r_{u,i} = k) = p_{u,i}^{(<k+1)} - p_{u,i}^{(<k)}$

Illustration





1. We are using the matrix notation $A = (a_{i,i})_{i\le n, j\le m}$

Newton's method boosted maximum likelihood parameter estimation for ordered target classes

Maximum likelihood

The parameter matrixes E, Q and the intercepts t are estimated such the log-likelihood to observe $R \in \{0,1,..,n_k, NaN\}^{n_u \times n_i}$ is maximized

$$\log \mathrm{L}(\mathrm{E}, \mathrm{Q}, t) = \sum_{k \leq n_k} \left\{ \sum_{u \leq n_u, i \leq n_i, r_{u,i}=k} \left[\log\left(l\left(s_{u,i} + t_{k+1}\right)\right) - \log\left(l(s_{u,i} + t_k)\right)\right]\right\}$$

with $\mathrm{S} = X\ E\ Q\ \in \mathbb{R}^{n_u \times n_i}$ and $l(x) = \frac{1}{1+e^{-x}}$.

Note that only known product usages, i.e. $r_{u,i} \neq NaN$, are considered in the likelihood.

Two different alternating Newton's method

The logitSVD package offers two options for solving the non-linear likelihood optimization problem:

- an alternating Newton's method with line search to iteratively optimize simultaneously t and E for fixed Q and then t and Q for fixed E
- An alternating Newton's methods with line search to iteratively optimize all 3 elements separately, i.e. first E with fixed t and Q, then t for fixed E and Q, and subsequently Q with fixed t and E.

For most problems, both approaches are successful. Typically, the first approach is quicker but less stable, i.e. one of the Hessians might not be positive definite.

logitSVD is implemented in a Python package (1/2)

Function call

```
P, C, Z, E, Q, t, z_score, p_value  = logitSVD(X, R, depth, la, E = None, Q = None, t=None, method
="alternating", tol = 1e-4, maxit = 20, tolNewton = None, maxitNewton = 100, verbose = "warn")
```

Parameter

```
X      : ndarray[nuser,nfeature], user feature vectors
R      : ndarray[nuser,nitem], user-item-matrix (target)
depth  : int, model parameter, depth of the embeddings = number of different models
la     : float, regularization paramter
E      : ndarray[nfeature,depth], initial solution for the feature weights (embeddings)
Q      : ndarray[depth,nitem], initial solution for the item embeddings (model combination parameter)
t      : ndarray[max(R)], initial solution for intercepts (only for multinomial case)
method : string, binary: alternating [alter], fullNewton [full], alter_full, i.e. first alter, then
                         fullNewton
             multinomial: alternating2 [alter2], 2 alternating steps  1. Q,t and 2. E,t
                          alternating3 [alter3], 3 alternating steps  1. Q, 2. t, 3. E
tol    : float, alternating methods stop if the reduction of the log-likelihood is smaller than tol
maxit  : int, maximum number of iterations of the alternating methods
tolNewton: float, Newton's method stops if the 2-norm of the gradient becomes smaller than tolNewton
maxitNewton: int, maximum number of iterations of Newton's method
verbose: string, ("none" | "warn" | "all"), print warnings and convergence progress. Default is "warn"
```

logitSVD is implemented in a Python package (2/2)

Function call

```
P, C, Z, E, Q, t, z_score, p_value  = logitSVD(X, R, depth, la, E = None, Q = None, t=None, method
="alternating", tol = 1e-4, maxit = 20, tolNewton = None, maxitNewton = 100, verbose = "warn")
```

Output

```
P       : binary: ndarray[nuser,nitem], P[u,i] is the probability that R[u,i] = 1
          multinomial: ndarray[max(R),nuser,nitem], P[k,u,i] is the probability that R[u,i] = k
C       : ndarray[nuser,nitem], most likely class - None for binary
Z       : ndarray[nuser,nitem], Z = X E Q
E       : ndarray[nfeature,depth], solution for the parameter E
Q       : ndarray[depth,nitem], solution for the parameter Q
t       : ndarray[max(R)], solution for the intercepts t - None for binary
z-score: ndarray[nfeature,depth], z-score from Wald test for the parameter E
p-value: ndarray[nfeature,depth], p-values from Wald test for the parameter E
```