Scaling Factorization Machines to Relational Data

Steffen Rendle
University of Konstanz
78457 Konstanz, Germany

Introduction

- Nowadays, feature engineering based ML is the dominant technique in predictive analytics. However, if it is applied to relational data, especially involving relations of high cardinality, the feature vectors can grow very large which can make learning and prediction very slow or even infeasible.

- To follow the example from above, the friends of a customer might be predictive for his/her taste. Using the variable ”friends of a customer” in the feature vector (e.g. for a SVM, LR, etc.) can result in a very long feature vector because all friends (i.e. their IDs) are included in the feature vector.

- In this paper, it is shown how prediction and learning algorithms for linear regression and factorization machines can be scaled to predictor variables generated from relational data involving relations of high cardinality.

- The idea is to make use of repeating patterns over a set of feature vectors.

- While no change is made on the predictive modeling approach and also not on the underlying statistical model. Thus the proposed algorithms learn the same parameters and make the same predictions but with a much lower runtime complexity.

Introduction (2)

- From a practical point of view, the proposed algorithms allow to handle predictive modeling as usual: defining predictor variables (also variables from relations of high cardinality) by feature engineering and applying a feature-vectorbased ML algorithm.

- Internally, the algorithms make use of the repeating patterns stemming from the relational structure of the data to largely speed up computation.

Rating Events

UserID	*MovieID*	*Score*	*Date*
Alice	TI	5	2012-09-01
Alice	NH	3	2012-09-12
Alice	SW	1	2012-09-15
Bob	SW	4	2012-09-02
Bob	ST	5	2012-10-07
Charlie	TI	1	2012-09-05
Charlie	SW	5	2012-09-05
...	...	...	...

User

ID	*Gender*	*Age*
Alice	F	30
Bob	M	25
Charlie	M	28
…	...	

Friends

ID1	*ID2*
Alice	Eve
Alice	Charlie
Bob	Charlie
Bob	Dave
Charlie	Alice
Charlie	Bob
Charlie	Dave
…	...

Movie

ID	*Genre*
TI	Action
TI	Romance
SW	Science Fiction
…	...

Figure 1: Example database from a movie community.

Type of Modeling

- Predictive Modeling: the n cases each represented by a feature vector $x \in R_p$ can be seen as a matrix $X \in R_{n^\circ p}$ which is typically called design matrix. The n prediction targets can be represented as an n dimensional vector $y \in R_n$ for regression (or $y \in \{-,+\}^n$ for binary classification).

- Often the data is sparse, i.e. contains many 0 values. Let $N_z(X)$ denote the number of non-zeros in a matrix X. Sparse learning algorithms can make use of the non-zeros in the training data and a desirable property is a runtime complexity linear in $N_z(X)$.

Type of Modeling (2)

- Relational Predictive Modeling: The standard approach of predictive modeling to select predictor variables and to learn the dependency from features. However, relational data can lead to very large feature vectors/ design matrices with redundant information.

- Example: If we are trying to find liking of Alice and she rated 5 to titanic,all of this information will be present redundantly in Alice's final features and data. Whenever a case uses predictor variables describing Alice, the feature vector will include all of Alice's descriptors including age, gender, friends, etc. (Data on next slide)

- Thus, relational datasets can result in very large design matrices that are infeasible for standard algorithms, even if the algorithm has a linear runtime complexity in $N_z(X)$.

Type of Modeling (3)

(a) Predictive function

$$\text{score} : \text{UserID} \times \text{MovieID} \times \text{Date} \times \text{UserGender} \times \text{UserAge} \times \text{MovieGenres} \times \text{UserFriends} \times \text{ItemsWatched} \rightarrow \mathbb{R}$$

(b) Training Data for Predictive Function

UserID	*MovieID*	*Date*	*Gender*	*Age*	*Genres*	*Friends*	*ItemsWatched*	*Score*
Alice	TI	2012-09-01	F	30	{A,R}	{E,C}	{TI,NH,SW}	5
Alice	NH	2012-09-12	F	30	{C,R}	{E,C}	{TI,NH,SW}	3
Alice	SW	2012-09-15	F	30	{S,A}	{E,C}	{TI,NH,SW}	1
Bob	SW	2012-09-02	M	25	{S,A}	{C,D}	{SW,ST}	4
Bob	ST	2012-10-07	M	25	{S}	{C,D}	{SW,ST}	5
Charlie	TI	2012-09-05	M	28	{A,R}	{A,B,D}	{TI,SW}	1
Charlie	SW	2012-09-05	M	28	{S,A}	{A,B,D}	{TI,SW}	5
...	...	...	...	...	...	...	...	...

(c) Training Data in Numeric Format (*Design Matrix*)

UserID	*MovieID*	*Date*	*Gender*	*Age*	*Genres*	*Friends*	*ItemsWatched*	*Score*
1 0 0	1 0 0 0	1	1 0	30	.5 .5 0 0	0 0 .5 0 .5	.3 .3 .3 0	5
1 0 0	0 1 0 0	12	1 0	30	0 .5 .5 0	0 0 .5 0 .5	.3 .3 .3 0	3
1 0 0	0 0 1 0	15	1 0	30	.5 0 0 .5	0 0 .5 0 .5	.3 .3 .3 0	1
0 1 0	0 0 1 0	2	0 1	25	.5 0 0 .5	0 0 .5 .5 0	0 0 .5 .5	4
0 1 0	0 0 0 1	37	0 1	25	0 0 0 1	0 0 .5 .5 0	0 0 .5 .5	5
0 0 1	1 0 0 0	5	0 1	28	.5 .5 0 0	.3 .3 0 .3 0	.5 0 .5 0	1
0 0 1	0 0 1 0	5	0 1	28	.5 0 0 .5	.3 .3 0 .3 0	.5 0 .5 0	5
...	...	...	...	...	...	...	...	...
A B C	*TI NH SW ST*		*F M*		*A R C S*	*A B C D E*	*TI NH SW ST*	

← *corresponding levels of categorical variables*

Block Structure

- Repeating patterns in X can be formalized by a condensed block structure representation B.

- Let B = {B_1,B_2, . . .} be a set of blocks, where each block B = (X_B,) consists of a design matrix $X_B \in R^{nB \times pB}$ and a mapping B: {1, . . . , n} → {1, . . . , n_B} from rows in the original design matrix X to rows within X_B. B is a block structure representation of X iff for all rows i:

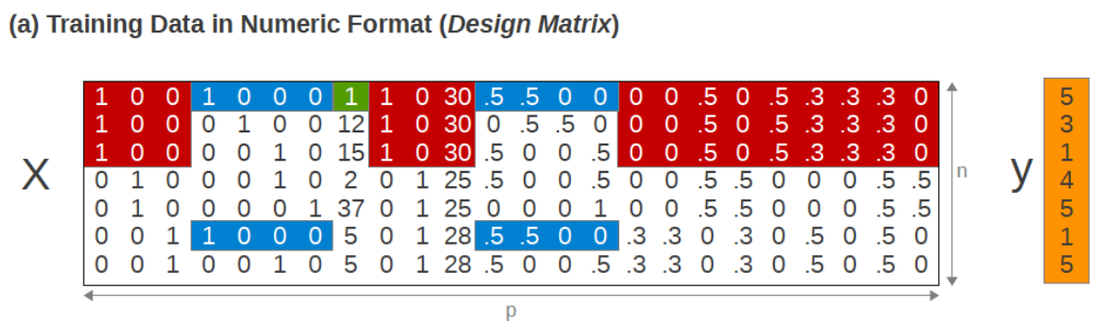
$$\mathbf{x}_i \equiv (x^{B_1}_{\phi^{B_1}(i),1}, x^{B_1}_{\phi^{B_1}(i),2}, \ldots, x^{B_2}_{\phi^{B_2}(i),1}, x^{B_2}_{\phi^{B_2}(i),2}, \ldots) \quad (1)$$

- That is the X can be reconstructed by reverse mapping B, i.e applying eq. (1) to each of the n rows

Block Structure (2)

- The design matrix can be grouped in three blocks B_1,B_2,B_3 where the first block represents the user, the second the movie and the third the time. Each block consists of a design matrix and a mapping.

- E.g. X_{B1} is the design matrix with all the predictor variables of the user and Φ_{B1} the mapping from the case to the row of the design matrix of the block. E.g. the fourth case of the original design matrix X can be obtained by concatenating the second row of X_{B1}, the third row of X_{B2} and the fourth row of X_{B3}.



Complexity Analysis

- The complexity of a block structure representation is the sum of the complexity of all blocks plus the size of the mappings, i.e:

$$N_Z(\mathcal{B}) := |\mathcal{B}|\, n + \sum_{B \in \mathcal{B}} N_Z(X^B)$$

- $N_z(B) \ll N_z(X)$. Where z is the number of rows of training data.

Scaling Linear Regression

- Standard Linear Regression:

    1. Method of least least square (p x p) $O(p^3)$

    2. Iterative approaches where large number of predictor variables p and coordinate descent (CD). The CD algorithm starts with an initial (random) guess of , then iterates over each model parameter $w_l \in$ and performs an update:

$$w_l \leftarrow \frac{w_l \sum_{i=1}^{n} x_{i,l}^2 + \sum_{i=1}^{n} x_{i,l}\, e_i}{\sum_{i=1}^{n} x_{i,l}^2 + \lambda_l}$$

    Where $\lambda_l \in R+$ is a predefined regularization constant and $e_i = y_i - \hat{y}(x_i)$ which is residual change and is updated in constant time.

3. The runtime of CD is dominated by computing the two quantities

$$\sum_{i=1}^{n} x_{i,l}^2, \qquad \sum_{i=1}^{n} x_{i,l}\, e_i.$$

4. Bayesian inference can include uncertainty into the model. Bayesian inference typically improves the prediction quality and also allows to infer regularization values automatically. (Gibbs Sampling is used generally)

Scaling Linear Regression (2)

- For scaling LR, both learning and prediction has to make use of the block structure representation. For achieving a linear runtime in $N_Z(B)$.

- Predictions can be made like:

$$\hat{y}(\mathbf{x}_i) = w_0 + \sum_{B \in \mathcal{B}} \sum_{j=1}^{p^B} w_j^B \, x_{\phi(i),j}^B = w_0 + \sum_{B \in \mathcal{B}} q_{\phi(i)}^B$$

where

$$q_i^B = \sum_{j=1}^{p^B} w_j^B \, x_{i,j}^B, \quad \forall i \in \{1, \ldots, n^B\} \qquad ($$

- Where prediction = Bias + sum of product of weights and block matrixes.

- Thus, This directly shows how n cases can be efficiently predicted: (i) compute q_B in $O(N_Z(X_B))$ for each block B.

Scaling Linear Regression (3)

- Learning: The goal is to rearrange the sum to iterate over n^B element of the block instead of the n elements in the original design matrix. The idea is that for any function f_i The sum can be rewritten as :

$$\sum_{i=1}^{n} f(i) = \sum_{i=1}^{n^B} \sum_{j=1}^{n} \delta(\phi^B(j) = i) f(j).$$

- And a decomposition of f has to be found that can replace the inner sum with a term dependent only on i, with constant computation time. We minimise the two dependent quantities as :

$$\sum_{i=1}^{n} x_{i,l}^2 = \sum_{i=1}^{n^B} \sum_{j=1}^{n} \delta(\phi^B(j) = i) x_{j,l}^2 = \sum_{i=1}^{n^B} (x_{i,l}^B)^2 \#_i^B$$

with the constant

$$\#_i^B = \sum_{j=1}^{n} \delta(\phi^B(j) = i)$$

which counts how many mappings go to row i of block B

$$\sum_{i=1}^{n} x_{i,l}\, e_i = \sum_{i=1}^{n^B} x_{i,l}^B e_i^B, \quad e_i^B := \sum_{j=1}^{n} \delta(\phi^B(j) = i)\, e_j. \quad (10)$$

Here $e^B{}_i$ is not a constant but a block depending sum of those residuals that are mapped by from X to the i-th row of the block.

Hence the parameters are updated after each step of CD. $e_i^B \leftarrow e_i^B + \Delta_l\, x_{i,l}^{B'} \#_i^B$.

SCALING FACTORIZATION MACHINES

- Standard Factorization Machines: Factorization machines (FM) include interactions between predictor variables. This allows to learn more complex functions (esp. non-linear ones) than with linear regression.

- A second-order FM for the i-th feature vector xof the n x p design matrix X is

$$\hat{y}(\mathbf{x}_i) = w_0 + \sum_{j=1}^{p} w_j \, x_{i,j} + \frac{1}{2} \sum_{f=1}^{k} \left[\left(\sum_{j=1}^{p} v_{j,f} \, x_{i,j} \right)^2 - \sum_{j=1}^{p} v_{j,f}^2 \, x_{i,j}^2 \right].$$

where last term is the effect of the variable interaction x (i,j) with the dot product of two k-dimensional latent.

- Learning: For CD, a model parameter $\boldsymbol{\theta} \in \Theta$ is updated by the equation:

$$\theta \leftarrow \frac{\theta \sum_{i=1}^{n} h_\theta(\mathbf{x}_i)^2 + \sum_{i=1}^{n} h_\theta(\mathbf{x}_i) \, e_i}{\sum_{i=1}^{n} h_\theta(\mathbf{x}_i)^2 + \lambda_\theta}.$$

- Similarly lambda is the regularization constant and ei is the ith residual.

- Again two important values that determine the runtime are

$$\sum_{i=1}^{n} h_\theta(\mathbf{x}_i)^2, \qquad \sum_{i=1}^{n} h_\theta(\mathbf{x}_i) \, e_i.$$

where h is the descent function of variable xi

SCALING FACTORIZATION MACHINES (2)

- Scaling Factorization Machines to Block Structures:This is more complicated than for LR because FMs contain all pairwise interactions between all predictor variables.

- Prediction:

$$\hat{y}(\mathbf{x}_i) = w_0 + \sum_{B \in \mathcal{B}} q^B_{\phi(i)} + \frac{1}{2} \sum_{f=1}^{k} \left[\left(\sum_{B \in \mathcal{B}} q^B_{\phi(i),f} \right)^2 - \sum_{B \in \mathcal{B}} q^{B,S}_{\phi(i),f} \right] \tag{19}$$

with the caches

$$q^B_{i,f} := \sum_{j=1}^{p^B} v^B_{j,f}\, x^B_{i,j}, \qquad q^{B,S}_{i,f} := \sum_{j=1}^{p^B} (v^B_{j,f}\, x^B_{i,j})^2.$$

- Computing the caches are $O(kN_z(X^B))$, Computing the n predictions is in $O(k\ n\ |B|)$.The total runtime for predicting n cases is $O(kN_z(B))$.

- q, caches can be isolated and precomputed which results in an efficient computation.

SCALING FACTORIZATION MACHINES (3)

- Learning: To make use of the repeating patterns in X, these sums have to be rewritten such that computing the whole sum is in $O(n^B)$ instead of O(n). The basic idea is similar to LR to apply factorization of f in a way that it is dependents only on Blocks.

- This will require several caches for which efficient updates have to be derived.

$$\sum_{i=1}^{n} h_{v_{l,f}}(\mathbf{x}_i)^2 = \sum_{i=1}^{n^B} \Big[\#_i^B (h_{v_{l,f}}^B(\mathbf{x}_i^B))^2 + 2c_{i,f}^B x_{i,l}^B h_{v_{l,f}}^B(\mathbf{x}_i^B) + (x_{i,l}^B)^2 c_{i,f}^{B,S} \Big] \quad (21)$$

where

$$h_{v_{l,f}}^B(\mathbf{x}_i^B) := x_{i,l}^B \left(q_{i,f}^B - v_{l,f}^B x_{i,l}^B \right),$$

and the new caches

$$c_{i,f}^B := \sum_{j=1}^{n} \delta(\phi^B(j) = i) \sum_{B' \in (\mathcal{B} \setminus \{B\})} q_{\phi(j),f}^{B'}$$

$$c_{i,f}^{B,S} := \sum_{j=1}^{n} \delta(\phi^B(j) = i) \left(\sum_{B' \in (\mathcal{B} \setminus \{B\})} q_{\phi(j),f}^{B'} \right)^2 .$$

- If cachs are present the equation 21 can be computed in $O(n^B)$. And final parameters are updated

$$\sum_{i=1}^{n} h_{v_{l,f}}(\mathbf{x}_i^B) e_i = \sum_{i=1}^{n^B} \left(h_{v_{l,f}}^B(\mathbf{x}_i^B) e_i^B + x_{i,l}^B e_{i,f}^{B,q} \right) \quad (22)$$

where e_i^B is the same as for LR and

$$e_{i,f}^{B,q} = \sum_{j=1}^{n} \delta(\phi^B(j) = i) e_j \sum_{B' \in (\mathcal{B} \setminus \{B\})} q_{\phi(j),f}^{B'}.$$

$$e_i^B \leftarrow e_i^B + \Delta_{l,f}^B \left(h_{v_{l,f}}^B(\mathbf{x}_i^B) \#_i^B + x_{i,l}^B c_{i,f}^B \right)$$

$$e_{i,f}^{B,q} \leftarrow e_{i,f}^{B,q} + \Delta_{l,f}^B \left(h_{v_{l,f}}^B(\mathbf{x}_i^B) c_{i,f}^B + x_{i,l}^B c_{i,f}^{B,S} \right)$$

Both caches have to be kept in sync while model parameters are learned.

Evaluation and Dataset

- (1) relational predictor variables improve the quality and (2) the scaled FM model can achieve a high prediction quality in very competitive tasks.

- Netflix Prize and KDDCup 2012 dataset were used

Conclusion

- In this work, it was shown that design matrices from relational data can get very large which makes learning and prediction slow or even infeasible for standard machine learning algorithm.

- In a study on two very competitive ML tasks, it was shown that the proposed algorithms have a large speedup compared to standard feature-based learning algorithms and a prediction quality as good as the best specialized models develop for the respective tasks.

Thank You

–Shivam Mehta