Multi-relational Weighted Tensor Decomposition

Ben London, Theodoros Rekatsinas, Bert Huang, and Lise Getoor
Department of Computer Science
University of Maryland, College Park
College Park, MD 20742
{blondon,thodrek,bert,getoor}@cs.umd.edu

1 Introduction

There has recently been a growing interest in tensor methods within the machine learning community [1, 2, 3, 4, 5, 6, 7, 9], partially due to their natural representation of *multi-relational* data. Multi-relational data appears in applications such as social network analysis, where links between individuals may be personal, familial, or professional. In this paper, we examine a multi-relational learning scenario in which the learner is given a small training set, sampled from the set of all potential pairwise relationships, and aims to perform *transductive* inference on the remaining, unknown relationships. The target relations we consider may be binary-, discrete ordinal- or real-valued functions of the object pairs. To model this data, we propose a tensor decomposition method that is natural for multi-relational data, and produces more accurate predictions using minimal training data.

Our proposed method, *multi-relational weighted tensor decomposition* (MrWTD), assumes that the relationships are determined by a linear combination of latent factors associated with each object. Learning these factors, and their interactions in each relation, thus becomes analogous to a tensor decomposition (see Figure 1). Though our factorization is similar to that of Nickel et al. [7], unlike theirs and other approaches, we do not decompose the input tensor directly; instead, we explicitly model a mapping from the low-rank representation to the observed tensor, which is often better suited to prediction. For example, a binary relationship can be modeled as the sign of a latent representation; this gives the latent representation more freedom to increase the prediction margin, rather than reproduce $\{\pm 1\}$ exactly.

We formulate the decomposition as a nonlinear optimization problem, using a combination of task-specific, weighted loss functions to enable simultaneous learning of multiple relations. By weighting the loss function, we are able to leverage limited observed (training) relationships without fitting the unobserved (testing) ones. Further, due to our decomposition, we are able to transfer information across the various types of relations during learning, thus better exploiting the structure of multi-relational data. We demonstrate the efficacy of this approach using two synthetic data experiments.[1] The results show significant improvements in accuracy over competing factorization models.

2 Multi-relational Weighted Tensor Decomposition

Fix a set of objects $\{o_1, \ldots, o_m\}$ and a set of relations $\{\mathcal{R}_1, \ldots, \mathcal{R}_n\}$.[2] We are given a partially observed tensor $\mathbf{Y} \in \mathbb{R}^{m \times m \times n}$, in which each observed entry is a (possibly noisy) measurement of a relationship $y_{i,j,k} \approx \mathcal{R}_k(o_i, o_j)$ and each unobserved entry is set to a null value. We are additionally given a nonnegative weighting tensor $\mathbf{W} \in \mathbb{R}^{+^{m \times m \times n}}$, where each entry $w_{i,j,k} \in [0, 1]$ corresponds to a user-defined confidence, or certainty, in the value of $y_{i,j,k}$; if $y_{i,j,k}$ is unobserved,

[1] While we also have promising results using real-world data, we omit them here due to space restrictions.

[2] Here, we use the term *relation* loosely to include not only strict relations, for which relationships are either present or not, but also real-valued functions. To simplify our analysis, we assume that all relations are symmetric, though one can obtain an analogous derivation for asymmetric relations with slightly more work.

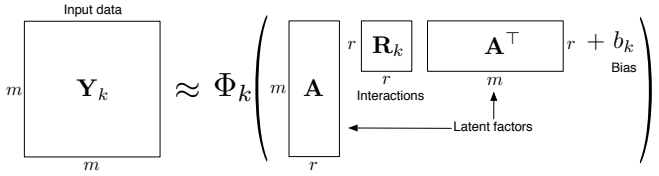


Figure 1: Each slice $\mathbf{Y}_k$ of the input tensor is approximated by a function Φ_k of a low-rank decomposition $\mathbf{A}\mathbf{R}_k\mathbf{A}^\top + b_k$. The latent factors $\mathbf{A}$ are common to all slices. Each $\mathbf{R}_k$ determines the interactions of $\mathbf{A}$ in the k^{th} relation, while b_k accounts for distributional bias.

then $w_{i,j,k}$ is necessarily zero. The goal of multi-relational transduction in this tensor formulation is to infer the unobserved entries in $\mathbf{Y}$.

Our fundamental assumption is that each relationship $\mathcal{R}_k(o_i, o_j)$ is equal to a mapping Φ_k applied to an element $x_{i,j,k}$ in an underlying low-rank tensor $\mathbf{X} \in \mathbb{R}^{m \times m \times n}$. Each Φ_k depends on the nature of the relation $\mathcal{R}_k$, and may differ across relations. For example, for binary relations in $\{\pm 1\}$, Φ_k is the *sign* function. We further assume that each frontal slice $\mathbf{X}_k$ can be factored as a rank-r decomposition:

$$\mathbf{X}_k = \mathbf{A}\mathbf{R}_k\mathbf{A}^\top + b_k, \tag{1}$$

where $\mathbf{A} \in \mathbb{R}^{m \times r}$, $\mathbf{R}_k \in \mathbb{R}^{r \times r}$ and $b_k \in \mathbb{R}$ (see Figure 1). Note that we place no constraints on $\mathbf{A}$ or $\mathbf{R}_k$; the columns of $\mathbf{A}$ need not be linearly independent, and $\mathbf{R}_k$ need not be positive-semidefinite. To infer the values of the missing (or uncertain) entries, we predict each $y_{i,j,k}$ by computing $x_{i,j,k} = \mathbf{a}_i^\top \mathbf{R}_k \mathbf{a}_j + b_k$ and applying the appropriate mapping.

The columns of $\mathbf{A}$ can be interpreted as the *global latent factors* of the objects, where the i^{th} row $\mathbf{a}_i$ corresponds to the latent factors of object o_i. Each $\mathbf{R}_k$ determines the interactions of $\mathbf{A}$ in the k^{th} relation. Thus, each predicted relationship comes from a linear combination of the objects' latent factors. Because the latent factors are global, information propagates between relations during the decomposition, thus enabling collective learning. The addition of b_k accounts for distributional bias within each relation.

The key distinction between our approach and previous tensor decompositions [1, 5, 7], for multi-relational learning is that, instead of directly decomposing the input tensor, we model the mapping from $\mathbf{X}$ to $\mathbf{Y}$. Moreover, we explicitly model the potential sparsity and uncertainty in the observations, producing more accurate predictions even when observed (training) data is limited (see Section 3).

To compute the decomposition in Equation 1, we minimize the following regularized objective:

$$f(\mathbf{A}, \mathbf{R}, \mathbf{b}) \triangleq \frac{\lambda}{2} ||\mathbf{A}||_{\text{F}}^2 + \sum_{k=1}^{n} \frac{\lambda}{2} ||\mathbf{R}_k||_{\text{F}}^2 + \text{tr}\left(\mathbf{W}_k (\ell_k(\mathbf{Y}_k, \mathbf{X}_k))^\top\right), \tag{2}$$

where $\lambda \geq 0$ is a regularization parameter, $\mathbf{X}_k$ is computed by Equation 1, and ℓ_k is a loss function that is applied element-wise to the k^{th} slice. This ability to combine multiple loss functions is central to our approach, as the appropriate penalty depends on the mapping for each $\mathbf{X}_k$ to $\mathbf{Y}_k$. Though most matrix and tensor decompositions focus on minimizing the quadratic loss, this criterion may not be optimal for certain prediction tasks, such as binary prediction; by explicitly making the loss function for each slice task-specific, our framework offers more flexibility than related techniques.

To minimize Equation 2, we use *quasi-Newton* optimization; in particular, we implement our framework using the *limited-memory Broyden-Fletcher-Goldfarb-Shanno* (L-BFGS) algorithm. This requires the gradients of $f(\mathbf{A}, \mathbf{R}, \mathbf{b})$ w.r.t. $\mathbf{A}$, $\mathbf{R}_k$ and b_k. Leveraging the symmetry of $\mathbf{R}_k$, we derive these as

$$\nabla_{\mathbf{A}} f(\mathbf{A}, \mathbf{R}, \mathbf{b}) = \lambda\mathbf{A} + \sum_{k=1}^{n} 2\left(\mathbf{W}_k \odot \nabla_{\mathbf{X}_k} \ell_k(\mathbf{Y}_k, \mathbf{X}_k)\right) \mathbf{A}\mathbf{R}_k^\top \tag{3}$$

$$\nabla_{\mathbf{R}_k} f(\mathbf{A}, \mathbf{R}, \mathbf{b}) = \lambda\mathbf{R}_k + \mathbf{A}^\top \left(\mathbf{W}_k \odot \nabla_{\mathbf{X}_k} \ell_k(\mathbf{Y}_k, \mathbf{X}_k)\right) \mathbf{A}, \tag{4}$$

$$\nabla_{b_k} f(\mathbf{A}, \mathbf{R}, \mathbf{b}) = \text{tr}\left(\mathbf{W}_k (\nabla_{\mathbf{X}_k} \ell_k(\mathbf{Y}_k, \mathbf{X}_k))^\top\right), \tag{5}$$

where $\odot$ denotes the *Hadamard* (i.e., element-wise) product, and $\nabla_{\mathbf{X}_k}\ell_k(\mathbf{Y}_k, \mathbf{X}_k)$ is the gradient of ℓ_k w.r.t. $\mathbf{X}_k$. Though this accommodates any differentiable loss function, we now present two that are applicable to many relational problems, and derive their corresponding gradients.

Quadratic Loss: The most common loss function used in matrix and tensor factorization is the *quadratic loss*, which we define as $\ell^q(y, x) \triangleq \frac{1}{2}(y-x)^2$. Minimizing the quadratic loss corresponds to the setting in which each relationship is directly approximated by a linear combination of latent factors, i.e., Φ_k is the identity and $\mathbf{Y}_k \approx \mathbf{X}_k$. For this loss function, the loss gradient is simply $\nabla_{\mathbf{X}_k}\ell_k(\mathbf{Y}_k, \mathbf{X}_k) \triangleq (\mathbf{X}_k - \mathbf{Y}_k)$.

Smooth Hinge Loss: While the quadratic loss may be appropriate for learning real-valued functions, it is sometimes ill-suited for learning binary relations. For binary classification, the goal is to complete a partially observed slice $\mathbf{Y}_k \in \{\pm 1\}^{m\times m}$. Recall that the mapping Φ_k is the sign function, and so $y_{i,j,k} \approx \text{sgn}(x_{i,j,k})$. Approximating $\{\pm 1\}$ with a quadratic penalty may yield a "small-margin" solution, since high-confidence predictions will push low-confidence predictions closer to the decision boundary. To get a "large-margin" solution, we use the *smooth hinge loss* [8],

$$\ell^h(y, x) \triangleq h(yx), \quad \text{where} \quad h(z) \triangleq \begin{cases} 1/2 - z & \text{if } z \leq 0, \\ (1-z)^2/2 & \text{if } 0 < z < 1, \\ 0 & \text{if } z \geq 1. \end{cases}$$

Unlike the standard hinge loss, the smooth hinge is differentiable everywhere. To obtain closed-form gradients, we define tensors $\mathbf{P}, \mathbf{Q} \in \mathbb{R}^{m\times m\times n}$, where

$$p_{i,j,k} \triangleq \begin{cases} 1 & \text{if } 0 < y_{i,j,k}x_{i,j,k} < 1, \\ 0 & \text{otherwise,} \end{cases} \quad \text{and} \quad q_{i,j,k} \triangleq \begin{cases} 1 & \text{if } y_{i,j,k}x_{i,j,k} < 1, \\ 0 & \text{otherwise.} \end{cases}$$

We can therefore express the smooth hinge as

$$h(y_{i,j,k}x_{i,j,k}) = (p_{i,j,k}x_{i,j,k}^2 - 2q_{i,j,k}y_{i,j,k}x_{i,j,k} + q_{i,j,k})/2,$$

which we can differentiate w.r.t. $\mathbf{X}_k$ to obtain $\nabla_{\mathbf{X}_k}\ell_k(\mathbf{Y}_k, \mathbf{X}_k) \triangleq (\mathbf{P}_k \odot \mathbf{X}_k - \mathbf{Q}_k \odot \mathbf{Y}_k)$.

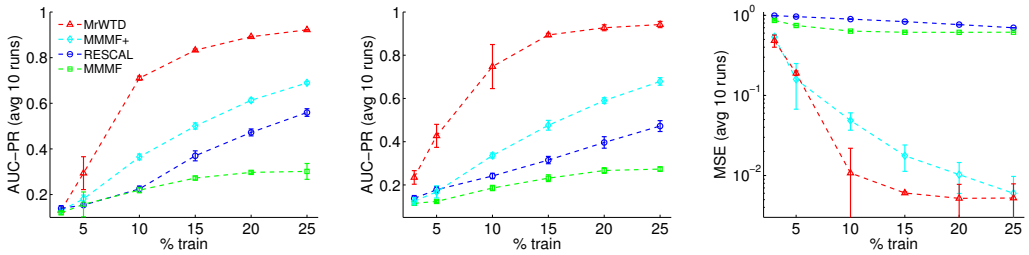
3 Experiments

In this section, we compare MrWTD with related tensor and matrix decompositions [7, 8] in two experiments using synthetic data. (We omit our real-world data experiments due to space limitations.)

The first method we compare against MrWTD is the RESCAL model [7]. In RESCAL, unobserved relationships are treated as negative examples. To distinguish between (un)observed relationships and negative examples, we modified the representation of binary relationships from $\{0, 1\}$ to $\{\pm 1\}$ for observed data and zeros elsewhere.[3] The second technique is *maximum-margin matrix factorization* (MMMF) [8], a model for matrix reconstruction. As such, it is not designed for multi-relational data; however, we can use it to reconstruct each slice of the tensor individually. Since the decomposition of MMMF differs significantly from MrWTD, to ensure a fair comparison, we run a variant of MrWTD that decomposes each slice separately instead of jointly, using a separate $\mathbf{A}_k$ for $k = 1, \ldots, n$. This is meant to equalize the discrepancy in decompositions, while isolating the deficiencies of non-collective learning. We refer to this model as MMMF+.

To generate the synthetic data, we start by computing a low-rank tensor $\hat{\mathbf{X}} \in \mathbb{R}^{m\times m\times n}$ as $\hat{\mathbf{X}}_k \leftarrow \hat{\mathbf{A}}\hat{\mathbf{R}}_k\hat{\mathbf{A}}^\top + \mathbf{E}_k$, for $k = 1, \ldots, n$, where $\hat{\mathbf{A}} \in \mathbb{R}^{m\times r}$ and $\hat{\mathbf{R}}_k \in \mathbb{R}^{r\times r}$ are sampled from a normal distribution, and $\mathbf{E}_k \in \mathbb{R}^{m\times m}$ is low-level, normally-distributed noise. For the first experiment, we construct $n = 3$ binary relations (i.e., slices), over $m = 500$ objects, using rank $r = 10$. We refer to this dataset as BinarySynthetic. To generate a binary $\mathbf{Y} \in \{\pm 1\}^{m\times m\times n}$, we round the values of $\hat{\mathbf{X}}_k$ using the 90^{th} percentile of $\hat{\mathbf{X}}$ as a threshold. This produces a heavy skew towards the negative class, as is typical in real multi-relational data. For the second experiment, we construct one binary relation and one real-valued relation, again over 500 objects, with rank 10. We refer to this dataset as MixedSynthetic.

[3] We find that this modification improves RESCAL's performance over the original method.



(a) BinarySynthetic (b) MixedSynthetic (AUC-PR) (c) MixedSynthetic (MSE)

Figure 2: Results of the synthetic experiments, with standard deviations. We plot the AUC-PR for BinarySynthetic (a) and MixedSynthetic (b) datasets. We also plot the MSE for MixedSynthetic (c); note that the MSE is an error measure, and thus lower numbers are better.

We evaluate over training sizes $t \in \{3, 5, 10, 15, 20, 25\}$ percent, using the remaining entries for testing. From each training set, we withold a random 25% of entries as a validation set for finding a good regularization parameter λ^*. Once identified, we then retrain on the full training set using λ^* and evaluate on the test set. The results we report are averaged over 10 runs per training size.

On BinarySynthetic, MrWTD achieves a statistically significant[4] lift over the competing methods for training sizes 5% and up. We attribute these results to two primary advantages: the weighted objective function, with its mixture of task-specific loss functions, and the global latent factors. The weighted objective allows exploiting small amounts of observed data, without fitting the unobserved entries. Since RESCAL treats all entries as observed, it tends to fit the unobserved entries in sparsely populated tensors. Furthermore, we observed improved performance over MMMF, since the latent factors are specific to each slice, while in MrWTD, information from one slice is propagated to the others via the global latent factors.

On MixedSynthetic, MrWTD's improvement over RESCAL and MMMF, in both AUC-PR and MSE, is statistically significantly for all training sizes. Compared to MMMF+, MrWTD's reduction in MSE is significant for sizes 10% and above. Yet, though MMMF+ is competitive with MrWTD in MSE for sizes 3-5%, since MMMF+ is unable to transfer information between slices, it still lags significantly behind MrWTD in AUC-PR for all training sizes.

We considered examining the benefit of large-margin learning in isolation (without weighting); however, this is not especially meaningful for transduction, as the hinge loss does not work with unobserved entries. In experiments not shown, we found that weighting alone (i.e., with quadratic loss) showed improvement over RESCAL in BinarySynthetic, but not as much as with the (large-margin) smooth hinge loss.

We also experimented with varying the rank of the decomposition from 5 to 40, but because the methods we test use L2 regularization on the latent factors, the results are nearly identical across ranks. Even with rank cross-validation per method, MrWTD still dominated the other methods. We believe this is because the L2 regularizer is the primary complexity control, so varying the rank has little effect on the predictions.

Finally, the running time for MrWTD is somewhat slower than that of RESCAL, in part because of the more complex objective function; yet it remains efficient because the sparsity of the observed tensor. On BinarySynthetic, using the optimal λ^* on 25% training data, learning executes in approximately ten seconds using our MATLAB implementation. This running time is comparable to RESCAL, which takes approximately five seconds. We are currently working on an algorithm to replace L-BFGS that will drastically improve efficiency.

References

[1] B. Bader, R. Harshman, and T. Kolda. Temporal analysis of semantic graphs using ASALSAN. In *Proc. of the 7th IEEE International Conf. on Data Mining (ICDM)*, 2007.

[4] We measure statistical significance in all experiments using a 2-sample t-test with rejection threshold 0.05.

[2] D. Dunlavy, T. Kolda, and W. Kegelmeyer. Multilinear algebra for analyzing data with multiple linkages. Technical Report, 2006.

[3] D. Dunlavy, T. Kolda, and E. Acar. Temporal link prediction using matrix and tensor factorizations. *ACM Trans. on Knowledge Discovery from Data*, 5(2), 2011.

[4] S. Gao, L. Denoyer, and P. Gallinari. Link pattern prediction with tensor decomposition in multi-relational networks. In *IEEE Symposium on Comp. Intell. and Data Mining*, 2011.

[5] R. Harshman. Models for analysis of asymmetrical relationships. In *First Joint Meeting of the Psychometric Society and the Society for Mathematical Psychology*, 1978.

[6] H. Kashima, T. Kato, Y. Yamanishi, M. Sugiyama, and K. Tsuda. Link propagation: A fast semi-supervised learning algorithm for link prediction. In *SIAM International Conference on Data Mining (SDM)*, 2009.

[7] M. Nickel, V. Tresp, and H. Kriegel. A three-way model for collective learning on multi-relational data. In *Proc. of the 28th International Conf. on Machine Learning (ICML)*, 2011.

[8] J. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *In Proc. of the 22nd International Conf. on Machine Learning (ICML)*, 2005.

[9] L. Xiong, X. Chen, T. Huang, J. Schneider, and J. Carbonell. Temporal collaborative filtering with Bayesian probabilistic tensor factorization. In *SIAM International Conference on Data Mining (SDM)*, 2010.