Journal of Artificial Intelligence Research 55 (2016) 1059-1090 Submitted 10/15; published 04/16

# Learning Concept Graphs from Online Educational Data

**Hanxiao Liu** HANXIAOL@CS.CMU.EDU
**Wanli Ma** MAWANLI@CS.CMU.EDU
**Yiming Yang** YIMING@CS.CMU.EDU
**Jaime Carbonell** JGC@CS.CMU.EDU
*School of Computer Science*
*Carnegie Mellon University*
*5000 Forbes Avenue*
*Pittsburgh, PA 15213 USA*

## Abstract

This paper addresses an open challenge in educational data mining, i.e., the problem of automatically mapping online courses from different providers (universities, MOOCs, etc.) onto a universal space of concepts, and predicting latent prerequisite dependencies (directed links) among both concepts and courses. We propose a novel approach for inference within and across course-level and concept-level directed graphs. In the training phase, our system projects partially observed course-level prerequisite links onto directed concept-level links; in the testing phase, the induced concept-level links are used to infer the unknown course-level prerequisite links. Whereas courses may be specific to one institution, concepts are shared across different providers. The bi-directional mappings enable our system to perform interlingua-style transfer learning, e.g. treating the concept graph as the interlingua and transferring the prerequisite relations across universities via the interlingua. Experiments on our newly collected datasets of courses from MIT, Caltech, Princeton and CMU show promising results.

## 1. Introduction

The large and growing amounts of online education data present both open challenges and significant opportunities for machine learning research to enrich educational offerings. One of the most important challenges is to automatically detect the prerequisite dependencies among massive quantities of online courses, and to support decision making such as curricula planning for students, and to support course and curriculum design by teachers based on existing course offerings. One example is to find a coherent sequence of courses among MOOC offerings from different providers that respect implicit prerequisite relations. A more specific example would be a new student who just enters a university for a MS or PhD degree. She is interested in machine learning and data mining courses, but finds it difficult to choose among many courses which look similar or with ambiguous course titles to her, such as *Machine Learning, Statistical Machine Learning, Applied Machine Learning, Machine Learning with Large Datasets, Scalable Analytics, Advanced Data Analysis, Statistics: Data Mining, Intermediate Statistics, Statistical Computing*, and so on. Completing all the courses would imply taking forever to graduate, and possibly waste a big portion of her time due to the overlapping content. Alternately, if she wants to choose a small subset, which courses should she include? How should she order the included courses

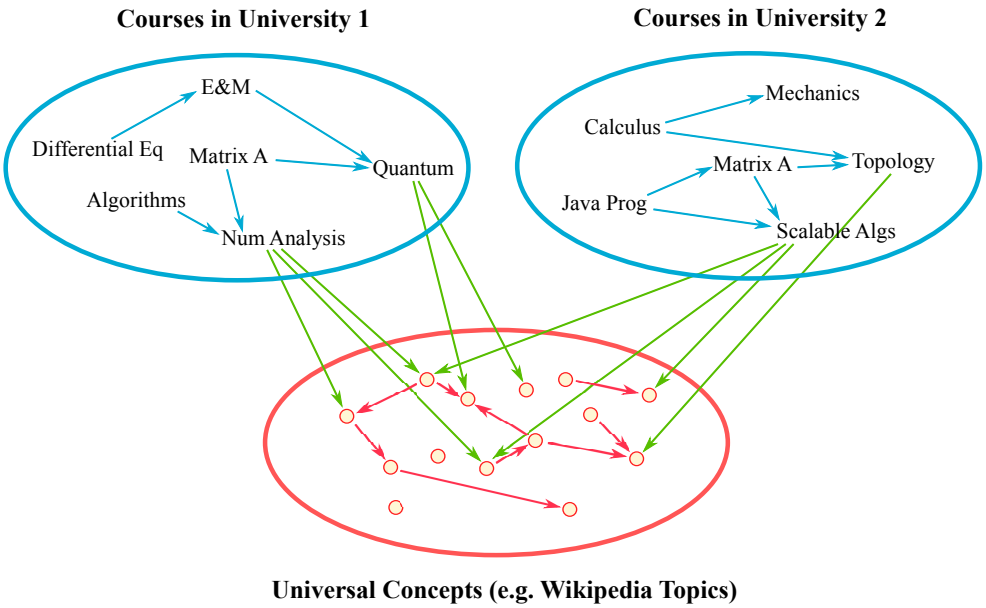


Figure 1: The framework of two-level directed graphs: The higher-level graphs have courses (nodes) with prerequisite relations (links). The lower-level graph consists of universal concepts (nodes) and pairwise preference in learning or teaching concepts. The links between the two levels are system-assigned weights of concepts to each course.

without sufficient understanding about the prerequisite dependencies? Often prerequisites are explicit within an academic department but implicit across departments. Moreover, if she already took several courses in machine learning or data mining through `Coursera` or in her undergraduate education, how much do those courses overlap with the new ones? Without an accurate representation of content overlap between courses and how the overlapped content reflects the prerequisite relations, it is difficult to help her find the most suitable courses in a correct order. Universities solve this problem in the old-fashioned way, via academic advisors, but it is not clear how to address this problem in the context of MOOCs or cross-university offerings where courses do not have unique IDs and are not described in a universally controlled or consistent vocabulary.

Ideally, we would like to have a universal graph whose nodes are canonical and discriminant concepts (e.g. "*convexity*" or "*eigenvalues*") being taught in a broad range of courses, and whose links indicate pairwise preferences in sequencing the teaching of these concepts. For example, to learn the concepts of *PageRank* and *HITS*, students should have already learned the concepts of *eigenvectors, Markov matrices* and *irreducibility of matrices*. This means directed links from *eigenvectors, Markov matrices* and *irreducibility* to *PageRank* and *HITS* in the concept graph. To generalize this further, if there are many directed links from the concepts in one course (say *Matrix Algebra*) to the concepts in another course (say *Web Mining* with *Link Analysis* as a sub-topic), we may infer a prerequisite relation between the two courses. Clearly, having a directed graph with a broad coverage of universal

concepts is crucial for reasoning about course content overlap and prerequisite relationships, and hence important for educational decision making, such as curriculum planning by students and modularization in course syllabus design by instructors.

How can we obtain such a knowledge-rich concept graph? Manual specification is obviously not scalable when the number of concepts reaches tens of thousands or larger. Using machine learning to automatically induce such a graph based on massive online course materials is an attractive alternative; however, no statistical learning techniques have been developed for this problem, to our knowledge. Addressing this open challenge with principled algorithmic solutions is the novel contribution we aim to accomplish in this paper. We call our new method Concept Graph Learning (CGL). Specifically, we propose a multi-level inference framework as illustrated in Figure 1, which consists of two levels of graphs and cross-level links. Generally, a course would cover multiple concepts, and a concept may be covered by more than one course. Notice that the course-level graphs do not overlap because different universities do not have universal course IDs. However, the semantic concepts taught in different universities do overlap, and we want to learn the mappings between the non-universal courses and the universal concept space based on online course materials.

In this paper we investigate the problem of concept graph learning (CGL) with our new collections of course syllabi (including course names, descriptions, listed lectures, prerequisite relations, etc.) from Massachusetts Institute of Technology (MIT), California Institute of Technology (Caltech), Carnegie Mellon University (CMU) and Princeton. The syllabus data allow us to construct an initial course-level graph for each university, which may be further enriched by discovering latent prerequisite links. As for representing the universal concept space, we study four representation schemes (Section 2.1), including 1) using the English words in course descriptions, 2) using sparse coding of English words, 3) using distributed word embedding of English words, and 4) using a large subset of Wikipedia categories. For each of these representation schemes, we provide algorithmic solutions to establish a mapping from courses to concepts, and to learn the concept-level dependencies based on observed prerequisite relations at the course level. The second part, i.e., the explicit learning of the directed graph for universal concepts, is the most unique part of our proposed framework. Once the concept graph is learned, we can predict *unobserved* prerequisite relations among any courses, including those not in the training set and by different universities. In other words, CGL enables an interlingua-style transfer learning as to train the models on the course materials of some universities and to predict the prerequisite relations for the courses in other universities. The universal transferability is particularly desirable in MOOC environments where courses are offered by different instructors in many universities. As we mentioned before, the course-level sub-graphs of different universities do not overlap with each other, and the prerequisite links are only local within each sub-graph. Thus to enable cross-university transfer, it is crucial to project course-level prerequisite links in different universities onto the directed links among universal concepts.

The bi-directional inference between the two directed graphs makes our CGL framework fundamentally different from existing approaches in graph-based link detection (Kunegis & Lommatzsch, 2009; Liben-Nowell & Kleinberg, 2007; Lichtenwalter, Lussier, & Chawla, 2010), matrix completion (Candès & Recht, 2009; Fazel, 2002; Johnson, 1990) and collaborative filtering (Su & Khoshgoftaar, 2009). That is, our approach requires explicit learning

of the concept-level directed graph and the optimal mapping between the two levels of links while other methods do not (see Section 7 for more discussion).

Our main contributions in this paper[1] can be summarized as:

1. A novel framework for within- and cross-level inference of prerequisite relations at the course-level and the concept-level directed graphs;

2. New algorithmic solutions for scalable concept graph learning under various (dense, sparse and transductive) settings;

3. New data collections from multiple universities with syllabus descriptions, prerequisite links and lecture materials;

4. The first evaluation for prerequisite link prediction in within- and cross-university settings.

The rest of the paper is organized as follows: Section 2 introduces the formal definitions of our framework and optimization objectives; Section 3 provides scalable algorithms for learning large concept graphs; Section 4 extends our new method to learn a sparse, parsimonious concept graph for better interpretability; Section 5 explores how unlabeled course pairs can be leveraged to significantly improve the prediction performance of the learned concept graph; Section 6 describes the new datasets we collected for this study and future benchmark evaluations, and reports our empirical findings; Section 7 discusses related work and how concept graphs can be deployed to benefit future educational applications; and Section 8 summarizes the main findings in this study.

## 2. Framework & Algorithms

Let us formally define our methods with the following notation.

- $n$ is the number of courses in a training set;
- $p$ is the dimension of the universal concept space (Section 2.1);
- $X = [x_1, x_2, \ldots, x_n]^\top \in \mathbb{R}^{n \times p}$ is a collection of $n$ courses, where $x_i \in \mathbb{R}^p$ is the bag-of-concepts representation of the $i$-th course;
- $Y \in \{-1, +1\}^{n \times n}$ is a collection of $n^2$ binary indicators of the observed prerequisite relations between courses, i.e., $y_{ij} = 1$ means that course $j$ is a prerequisite of course $i$, and $y_{ij} = -1$ otherwise.
- $A \in \mathbb{R}^{p \times p}$ is the adjacency matrix of the concept graph, whose elements are the weights of directed links among concepts. That is, $A$ is the matrix of model parameters we want to optimize given the training data in $X$ and $Y$.

1. This journal paper is a substantially extended version of a previous paper (Yang, Liu, Carbonell, & Ma, 2015).

### 2.1 Representation Schemes

What is the best way to represent the contents of courses to learn the universal concept space? We explore different answers with four alternate choices as follows:

1. ***Word-based Representation (Word)***: This method uses the vocabulary of course descriptions plus any listed keywords by the course providers (MIT, Caltech, CMU and Princeton) as the entire concept (feature) space. We applied standard procedures for text preprocessing, including stop-word removal, term-frequency (TF) based term weighting, and the removal of the rare words whose training-set frequency is one. We did not use TF-IDF weighting because the relative small number of "documents" (courses) in our datasets do not allow reliable estimates of the IDF part.

2. ***Sparse Coding of Words (SCW)***: This method projects the original n-dimensional vector representations of words (the columns in the course-by-word matrix $X$) onto sparse vectors in a smaller $k$-dimensional space using Non-negative Matrix Factorization (Lee & Seung, 1999), where $k$ is much smaller than $n$. One can view the lower dimensional components as the system-discovered latent concepts. Intrigued by the successful application of sparse coding in image processing (Hoyer, 2004), we explored its application to our graph-based inference problem. By applying an existing sparse coding algorithm (Kim & Park, 2008) to our training sets we obtained a $k$-dimensional vector for each word; by taking the average of word vectors in each course we obtained the bag-of-concepts representation of the course. This resulted in an $n$-by-$k$ matrix $X$, representing all the training-set courses in the $k$-dimensional space of latent concepts. We set $k = 100$ in our experiments based on cross validation.

3. ***Distributed Word Embedding (DWE)***: This method also uses dimension-reduced vectors to represent words in the courses, similar to SCW. However, the lower dimensional vectors (continuous vector representations) for words are discovered by neural networks based on word usage w.r.t. contextual, syntactic and semantic information (Le & Mikolov, 2014). Intrigued by the popularity of DWE in recent research in Natural Language Processing and other domains (Collobert, Weston, Bottou, Karlen, Kavukcuoglu, & Kuksa, 2011; Chen, Perozzi, Al-Rfou, & Skiena, 2013), we explored its application to our graph-based inference problem. Specifically, we deploy English word embeddings trained on Wikipedia articles (Al-Rfou, Perozzi, & Skiena, 2013), a domain which is believed to be semantically close to that of academic courses. The vector representation for each course is obtained by aggregating the vector representations of words it contains.

4. ***Category-based Representation (Cat)***: This method used a large subset of Wikipedia categories as the concept space. We selected the subset via a pooling strategy as follows: We used the words in our training-set courses to form 3509 queries (one query per course), and retrieved the top 100 documents per query based on cosine similarity. We then took the union of the Wikipedia category labels of these retrieved documents, and removed the categories which were retrieved by only three queries or less. This process resulted in a total of 10,051 categories in the concept space. The categorization of courses was based on an earlier highly scalable very-large category

space work (Gopal & Yang, 2013): the classifiers were trained on labeled Wikipedia articles and then applied to the word-based vector representation of each course for (weighted) category assignments.

Each of the above representation schemes may have its own strengths and weaknesses. *Word* is simple and natural but rather noisy, because semantically equivalent lexical variants are not unified into canonical concepts and there could be systematic vocabulary variation across universities. Also, this scheme will not work in cross-language settings, e.g., if course descriptions are in English and Chinese. *Cat* would be less noisy and better in cross-language settings, but the automated classification step will unavoidably introduce errors in category assignments. *SCW* (sparse coding of words) reduces the total number of model parameters via dimensionality reduction, which may lead to robust training (avoiding overfitting) and efficient computation, but at the risk of losing useful information in the projection from the original high-dimensional space to a lower dimensional space. *DWE* (distributed word embedding) deploys recent advances in deep learning of word meanings in context. However, reliable word embedding requires the availability of large volumes of training text (e.g., Wikipedia articles); the potential mismatch between the training domain (for which large volumes of data can be obtained easily) and the test domain (for which large volumes of data are hard or costly to obtain) could be a serious issue. Yet another distinction among these representation schemes is that *Word* and *Cat* produce human-understandable concepts and links, while *SCW* and *DWE* produce latent factors which are harder to interpret by humans, although methods such as L1 regularization help with interpretability (sec 6.5).

By exploring all the four representation schemes in our unified framework for two-level graph based inference, and by examining their effectiveness in the task of link prediction of prerequisite relations among courses, we aim to obtain a deeper understanding of the strengths and weaknesses of those representational choices.

### 2.2 The Optimization Methods

We define the problem of concept graph learning as a key part of learning-to-predict prerequisite relations among courses, i.e., for the two-level statistical inference we introduced in Section 1 with Figure 1. Given a training set of courses with a bag-of-concepts representation per course as a row in matrix X, and a list of known prerequisite links per course as a row in matrix Y, we optimize matrix A whose elements specify both the direction (sign) and the strength (magnitudes) of each link between concepts. We propose two new approaches to this problem: a classification approach and a learning-to-rank approach. Both approaches deploy the same extended versions of SVM algorithms with squared hinge loss, but the objective functions for optimization are different. We also propose a nearest-neighbor approach for comparison, which predicts the course-level links (prerequisites) without learning the concept-level links.

#### 2.2.1 The Classification Approach (CGL.Class)

In this method, we predict the score of the prerequisite link from course $i$ to course $j$ as:

$$F_{ij} = x_i^\top A x_j \tag{1}$$

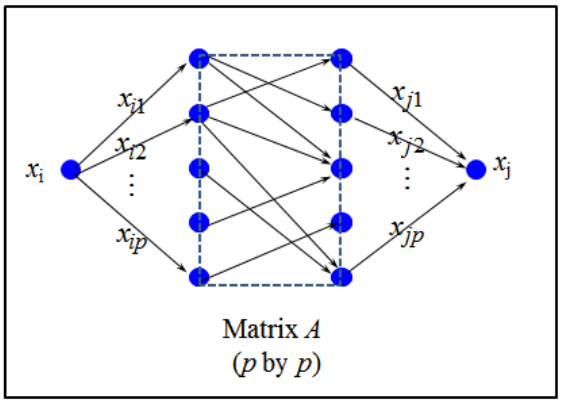


Figure 2: The weighted connections from course $i$ to course $j$ via matrix $A$ which encodes the directed links between concepts

The intuition behind this formula is shown in Figure 2. It can be easily verified that the quantity $x_i^\top A x_j$ is the summation of the weights of all the paths from node $i$ to node $j$ in this graph, where each path is weighted using the product of the corresponding $x_{ik}$, $A_{kk'}$ and $x_{jk'}$. In other words, we assume the prerequisite strength between two courses is a cumulative effect of the prerequisite strengths of all concept pairs.

The criterion for optimizing matrix $A$ given training data $x_i$ for $i = 1, 2, \ldots, n$ and true labels $y_{ij}$ for all course pairs is defined as:

$$\min_{A \in \mathbb{R}^{p \times p}} \sum_{i,j} \left[1 - y_{ij}\left(x_i^\top A x_j\right)\right]_+^2 + \frac{\lambda}{2}\|A\|_{\mathrm{F}}^2 \tag{2}$$

where $(1-v)_+ = \max(0, 1-v)$ denotes the hinge function, and $\|\cdot\|_{\mathrm{F}}$ denotes the matrix Frobenius norm. The 1st term in formula (2) is the empirical loss; the 2nd term is the regularization term, controlling the model complexity based on the large margin principle. We choose to use the squared hinge loss $(1-v)_+^2$ as the first term to gain first-order continuity of our objective function, enabling efficient computation using accelerated gradient descent (Nesterov, 1983, 1988) (Section 3). This efficiency improvement is crucial because we operate on pairs of courses, and thus have a much larger space than in normal classification (e.g. classifying individual courses).

#### 2.2.2 The Learning-to-Rank Approach (CGL.Rank)

Inspired by the learning-to-rank literature (Joachims, Li, Liu, & Zhai, 2007), we explored going beyond the binary classifier in the previous approach to one that essentially learns to rank prerequisite preferences. Let $\Omega_i$ be the set of course pairs with the true labels $y_{ij} = 1$ for different $j$'s, and $\bar{\Omega}_i$ the pairs of courses with the true labels $y_{ik} = -1$ for different $k$'s, we want our system to give all the pairs in $\Omega_i$ higher scores than that of any pair in $\bar{\Omega}_i$. We call this the partial-order preference over links conditioned on course $i$.

Let $T$ be the union of tuple sets $\{(i,j,k)|(i,j) \in \Omega_i, (i,k) \in \bar{\Omega}_i\}$ for all $i$ in $1, 2, \ldots, n$. We formulate our optimization problem as:

$$\min_{A \in \mathbb{R}^{p \times p}} \sum_{(i,j,k) \in T} \left[1 - \left(x_i^\top A x_j - x_i^\top A x_k\right)\right]_+^2 + \frac{\lambda}{2}\|A\|_{\mathrm{F}}^2 \tag{3}$$

Or equivalently, the objective can be rewritten as:

$$\min_{A \in \mathbb{R}^{p \times p}} \sum_{(i,j,k) \in T} \left[1 - \left(XAX^\top\right)_{ij} + \left(XAX^\top\right)_{ik}\right]_+^2 + \frac{\lambda}{2}\|A\|_{\mathrm{F}}^2 \tag{4}$$

Solving this optimization problem requires us to extend standard packages of SVM algorithms in order to improve computational efficiency because the number of model parameters ($p^2$) in our formulation is very large. For example, with the vocabulary size of 15,396 words in the MIT dataset, the number of model parameters in $A$ is over 237 million. When $p$ (the number of concepts) is much larger than $n$ (the number of courses), one may consider solving this optimization problem in the dual space instead of the primal space. However, even in the dual space, the number of dual variables is still $O(n^3)$, corresponding to all the triplets in $T$, and the kernel matrix is in the order of $O(n^6)$. We are going to address these computational challenges in Section 3.

#### 2.2.3 The Nearest-Neighbor Approach (kNN)

Different from the two approaches above where matrix $A$ plays a central role, as a different baseline, we propose to predict the prerequisite relationship for any pair of courses based on matrix $X$ and $Y$ without $A$. Let $(i', j')$ be a new pair of courses in the test set, we score each course pair $(i, j)$ in the training set with respect the new test pair as:

$$\left\langle x_{i'}, x_i \right\rangle \times \left\langle x_{j'}, x_j \right\rangle \quad \forall i, j = 1, 2, \ldots, n \tag{5}$$

where $\langle \cdot, \cdot \rangle$ stands for the inner product between two vectors. By taking the top-scored pairs in the training set and by aggregating the corresponding $y_{ij}$'s, we perform the kNN-based prediction of $\hat{y_{i'j'}}$ for the new test pair. If we normalize the vectors, the dot-products in (5) become the cosine similarity. This approach requires nearest-neighbor search on-demand; when the number of course pairs in the test set is large, the online computation would be substantial. Via cross validation, we have found $k = 1$ (1NN) works best for this problem on the current datasets.

#### 2.2.4 The Support Vector Machine (SVM)

As another baseline for comparision we also include an SVM with the following objective:

$$\min_{w \in \mathbb{R}^p} \sum_{i,j} \left[1 - y_{ij} \left(x_i - x_j\right)^\top w\right]_+ + \frac{\lambda}{2}\|w\|_2^2 \tag{6}$$

Similar to the kNN approach above and unlike CGL, the SVM optimization in (6) does not involve learning the matrix $A$ (the directed graph of universal concepts). The feature vector

for each course pair is simply the pairwise difference in the two vector representations (the two bags of words) of these courses. Once the model parameter vector $w$ is optimized on a labeled training set, it can be used to predict the prerequisite relation among any pair of courses by computing $w^\top (x_i - x_j)$ whose sign indicates the direction of the relationship, and whose magnitude indicates the strength of the relationship. By sorting the scores of the course pairs for each fixed $i$ in the test set, a ranked list of candidate prerequisites is obtained for course $i$.

## 3. Scalable Algorithms

Though it appears that any gradient-based method is directly applicable to CGL.Rank according to (4), the optimization is computationally challenging due to the following facts:

(a) The large number of course-level triplets $(i, j, k)$ in $T$, which can be $n^3$ in the worst case. This makes the gradient computation w.r.t. the loss term in (4) expensive.

(b) The large size of matrix $A$ in $\mathbb{R}^{p\times p}$. As an example, the number of entries in $A$ for *Word* representation on MIT dataset goes to 237 million, making the matrix manipulations costly both in time and space.

To tackle challenge (a), it is natural for one to consider Stochastic Gradient Descent (SGD). SGD avoids the expensive summation over all the $O(n^3)$ triplets by taking a noisy (instead of exact) gradient step in each iteration, where each noisy gradient step is computed solely based on one individual triplet randomly sampled from the training data. Stochastic optimization has been recently successfully applied to triplet-based loss functions, such as in collaborative filtering with implicit feedback (Rendle, Freudenthaler, Gantner, & Schmidt-Thieme, 2009). However, the success of SGD crucially relies on the assumption that each noisy gradient step is sufficiently cheap, which is not true in our case due to challenge (b).

To tackle challenge (b), one would consider solving the dual problem of CGL.Rank because the dual space may have a smaller number of coefficients to learn—in our case, the $p^2$ entries in $A$ are folded in the kernel matrix. However, as the number of dual variables for CGL.Rank is equal to the number of triplets, i.e., $n^3$ in the worst case, scalable optimization in the dual space is still hard.

In the following sections, we first address (b) by reformulating the CGL.Rank problem in (4) in the way that the optimization objective remains to be equivalent but the number of variables is substantially reduced (from $p^2$ to $n^2$). Then, we address the problem of (a) with two specific algorithms which have a substantially reduced number of iterations during the optimization.

### 3.1 Reduce the Number of Variables

**Theorem 3.1** (Variable Reduction). *Let the kernel matrix $K = XX^\top$, and let $\langle\cdot,\cdot\rangle$ be the matrix inner product. If $A^*$ is the minimizer for CGL.Rank in optimization* (4) *and $B^*$ is the minimizer for the following optimization problem*

$$\min_{B\in\mathbb{R}^{n\times n}} \quad \sum_{(i,j,k)\in T} \Big[1 - (KBK)_{ij} + (KBK)_{ik}\Big]_+^2 + \frac{\lambda}{2}\big\langle KBK, B\big\rangle \tag{7}$$

*then we have* $A^* = X^\top B^* X$.

*Proof.* First, let us introduce a dummy matrix variable $F = XAX^\top \in \mathbb{R}^{n\times n}$, where each element in $F$, denoted by $F_{ij}$, corresponds to our estimated strength of the prerequisite dependency from course $i$ to course $j$.

With $F$ we rewrite the unconstrained optimization (4) as a constrained optimization

$$\begin{aligned} \min_{A\in\mathbb{R}^{p\times p}, F\in\mathbb{R}^{n\times n}} \quad & \sum_{(i,j,k)\in T} (1 - F_{ij} + F_{ik})_+^2 + \frac{\lambda}{2}\|A\|_{\mathrm{F}}^2 \\ \text{subject to} \quad & F = XAX^\top \end{aligned} \tag{8}$$

Now we are going to show that in optimization (8), the degree of freedom of the optimal $A^*$ is actually much smaller than $p^2$.

To achieve this, we introduce a matrix dual variable $M \in \mathbb{R}^{n\times n}$ corresponding to the $n^2$ equality constraints in $F = XAX^\top$. The Lagrangian of (8) can be written as

$$\mathcal{L}(A, F, M) = \sum_{(i,j,k)\in T} (1 - F_{ij} + F_{ik})_+^2 + \frac{\lambda}{2}\|A\|_{\mathrm{F}}^2 + \left\langle F - XAX^\top, M \right\rangle \tag{9}$$

It is not hard to verity that (8) is a convex optimization with Slater's condition satisfied, hence strong duality holds. According to the stationarity condition, the derivative of the Lagrangian w.r.t. $A$ should vanish to zero in the optimal. That is

$$\begin{aligned} \frac{\partial \mathcal{L}(A, F, M)}{\partial A} &= \frac{\partial}{\partial A}\left(\frac{\lambda}{2}\|A\|_{\mathrm{F}}^2 + \left\langle F - XAX^\top, M \right\rangle\right) \\ &= \lambda A - \frac{\partial}{\partial A}\mathrm{tr}\left(X^\top M^\top X A\right) \\ &= \lambda A - X^\top M X \\ &= 0 \end{aligned} \tag{10}$$

$\implies A^* = \frac{1}{\lambda} X^\top M^* X$. It is worth noticing that while $A \in \mathbb{R}^{p\times p}$ contains $p^2$ variables, $A^*$ is completely determined by $M^* \in \mathbb{R}^{n\times n}$ which only involves $n^2$ variables ($n \ll p$).

Now let us define $B \equiv \frac{1}{\lambda} M \in \mathbb{R}^{n\times n}$. Combining $A^* = \frac{1}{\lambda} X^\top M^* X = X^\top B^* X$ with the constraint $F = XAX^\top$, we have $F^* = KB^*K$ where $K = XX^\top$. Plugging back the expressions of $A^*$ and $F^*$ to (8) yields optimization (7). □

The substantially reduced number of variables in optimization (7) allows us to efficiently compute and store the gradients in concept graph learning.

### 3.2 Reduce the Number of Iterations

In this section we introduce two algorithms for optimization (7) which leads to further speed ups by reducing the total number of iterations.

### 3.2.1 Accelerated Gradient Descent

Although gradient descent is readily applicable to (7), the convergence can become slow as we approach the optimal. The smoothness of our objective function (with the squared hinge loss) enables us to deploy Nestrerov's accelerated gradient descent (Nesterov, 1983, 1988), ensuring a faster convergence rate of $O(t^{-2})$ against the rate of $O(t^{-1})$ for gradient descent, where $t$ is the number of gradient steps.

Recall that in Section 3 we have $F = XAX^\top = KBK$. Denote by $\delta_{ijk} = (1 - F_{ij} + F_{ik})_+$ and by $e_i$ the $i$-th unit vector in $\mathbb{R}^n$.

The gradient of the objective in (7) w.r.t. $B$ is

$$\begin{aligned}
\nabla_B &= \sum_{(i,j,k)\in T} \nabla \left(1 - F_{ij} + F_{ik}\right)_+^2 + \frac{\lambda}{2}\nabla\langle F, B\rangle \\
&= -2 \sum_{(i,j,k)\in T} \delta_{ijk} \nabla \left(F_{ij} - F_{ik}\right) + \frac{\lambda}{2}\nabla \mathrm{tr}\left(BKB^\top K\right) \\
&= -2 \sum_{(i,j,k)\in T} \delta_{ijk} \left[\nabla \mathrm{tr}\left(BKe_j e_i^\top K\right) - \nabla \mathrm{tr}\left(BKe_k e_i^\top K\right)\right] + \lambda KBK \\
&= -2K \left[\sum_{(i,j,k)\in T} \delta_{ijk} \left(e_i e_j^\top - e_i e_k^\top\right)\right] K + \lambda F
\end{aligned} \tag{11}$$

Despite the large number of course-level triplets in $T$, matrix $\sum_{(i,j,k)\in T} \delta_{ijk} \left(e_i e_j^\top - e_i e_k^\top\right)$ of size $n \times n$ can still be computed efficiently since the majority of those triplets are "inactive" (i.e. $\delta_{ijk} = 0$ for a large number of triplet $(i, j, k)$) during the optimization. In fact, the number of operations required for evaluating the $\delta_{ijk}$'s can be substantially reduced by maintaining specialized data structures such as the order statistics tree (Cormen, Leiserson, Rivest, & Stein, 2001), which has recently been exploited to speed up the gradient computation of rankSVM (Lee & Lin, 2014; Airola, Pahikkala, & Salakoski, 2011).

Detailed implementation of CGL.Rank with accelerated gradient descent is summarized in Algorithm 1.

### 3.2.2 Inexact Newton Method

It often turns out that the bottleneck of gradient computation, after variable reduction, is a dense matrix-matrix multiplication in the complexity around $O(n^{2.373})$ (Davie & Stothers, 2013). The multiplication is affordable in our case since the number of courses $n$ we are dealing with is up to few thousands. However, to scale to substantially larger data collections, one may need to either consider further pruning the per-iteration complexity through techniques such as low-rank kernel approximation (Williams & Seeger, 2001), or further reducing the total number of iterations. In this section, we focus on the latter by incorporating second-order information using Newton's method.

The Newton's method we are going to derive is inexact (Dembo, Eisenstat, & Steihaug, 1982). That is, we are about to approximate the Newton direction in each iteration by approximately solving a linear system via preconditioned Conjugate Gradient method (PCG) without inverting the Hessian. In fact, we are going to avoid explicitly writing the Hessian

**Algorithm 1** CGL.Rank with Nestrerov's Accelerated Gradient Descent

**procedure** CGL.Rank.Nestrerov($X, T, \lambda, \eta$)
  $K \leftarrow XX^\top$, $B \leftarrow 0$, $Q \leftarrow 0$
  $t \leftarrow 1$
  **while** not converge **do**
    $\Delta \leftarrow 0$
    $F \leftarrow KBK$
    **for** $(i, j, k)$ **in** $T$ **do**
      $\tilde{\delta}_{ijk} \leftarrow 1 - F_{ij} + F_{ik}$
      **if** $\tilde{\delta}_{ijk} > 0$ **then**
        $\Delta_{ij} \leftarrow \Delta_{ij} + \tilde{\delta}_{ijk}$
        $\Delta_{ik} \leftarrow \Delta_{ik} - \tilde{\delta}_{ijk}$
    $P \leftarrow B - \eta\left(\lambda F - 2K\Delta K\right)$
    $B \leftarrow P + \frac{t-1}{t+2}\left(P - Q\right)$
    $Q \leftarrow P$
    $t \leftarrow t + 1$
  $A \leftarrow X^\top BX$
  **return** $A$

during the entire optimization process, since our Hessian $H \in \mathbb{R}^{n^2 \times n^2}$ (corresponding to the $n^2$ model parameters in $B \in \mathbb{R}^{n \times n}$) is extremely large.

Denote by $\otimes$ the Tensor (Kronecker) product operator between matrices, and by $e_{ij} = e_i \otimes e_j$ the $(i \times n + j)$-th unit vector in $\mathbb{R}^{n^2}$. The Hessian for (7) can be explicitly derived

$$H = 2\left(K \otimes K\right)\Lambda\left(K \otimes K\right) + \lambda K \otimes K \tag{12}$$

where $\Lambda$ is a shorthand for $\sum_{(i,j,k)\in T}\left(e_{ij} - e_{ik}\right)\left(e_{ij} - e_{ik}\right)^\top$.

Denote by "vec" the vectorization operator which concatenates the columns of a matrix into a single vector. Based on (11) and (12) one can verify it is always true that $\mathrm{vec}\left(\nabla_B\right) \equiv H\mathrm{vec}\left(B\right) - 2\left(K \otimes K\right)\sum_{(i,j,k)\in T}\left(e_{ij} - e_{ik}\right)$. Therefore, the Newton update is

$$\begin{aligned}
\mathrm{vec}\left(B\right) &\leftarrow \mathrm{vec}\left(B\right) - H^{-1}\mathrm{vec}\left(\nabla_B\right) \\
&= \mathrm{vec}\left(B\right) - H^{-1}\left[H\mathrm{vec}\left(B\right) - 2\left(K \otimes K\right)\sum_{(i,j,k)\in T}\left(e_{ij} - e_{ik}\right)\right] \\
&= 2H^{-1}\left(K \otimes K\right)\sum_{(i,j,k)\in T}\left(e_{ij} - e_{ik}\right) \\
&= 2\left[\Lambda\left(K \otimes K\right) + \lambda I_{n^2}\right]^{-1}\sum_{(i,j,k)\in T}\left(e_{ij} - e_{ik}\right)
\end{aligned} \tag{13}$$

Though it is even intractable to compute the $n^2 \times n^2$ matrix inside the inverse operation in (13), the updated $\mathrm{vec}(B)$ (or matrix $B$) can be well approximated by solving the following linear system via PCG iterative method

$$2\sum_{(i,j,k)\in T}\left(e_{ij} - e_{ik}\right) = \Lambda\left(K \otimes K\right)\mathrm{vec}\left(B\right) + \lambda\mathrm{vec}\left(B\right) \tag{14}$$

The computation bottleneck of PCG lies in matrix-vector multiplication $\Lambda\left(K \otimes K\right)\mathrm{vec}\left(B\right)$, which seems expensive and requires a huge dense matrix $K \otimes K \in \mathbb{R}^{n^2 \times n^2}$ to be stored in the memory. Interestingly, we can equivalently write the expression as $\Lambda \mathrm{vec}\left(KBK\right)$ by playing the "vec" trick for Tensor (Kronecker) product (Van Loan, 2000). After reformulation, the aforementioned matrix-vector multiplication becomes affordable since $\mathrm{vec}\left(KBK\right)$ can be computed in $O(n^{2.373})$ and $\Lambda$ is highly sparse.

The above suggests the complexity in each iteration of PCG is as cheap as that in each gradient step. We also empirically observed that the inexact Newton method only requires a small constant number (typically 3-5) of Newton updates to reach the optimal, and on average for each Newton update only 10 PCG iterations suffice to yield good results.

## 4. Learning a Sparse Concept Graph

Notice the CGL algorithm we studied so far produces a fully dense concept graph $A$. However, it is commonly believed that dependencies among the knowledge concepts should be highly sparse. A sparse concept graph is also desirable for visualization purpose thus allowing more intuitive user-exploration. For these reasons, in this section we modify our CGL algorithm to produce sparse graphs (sparse-CGL).

It is straightforward to enforce sparsity by replacing the $\ell_2$-norm over concept graph $A$ in the original CGL formulation to be the $\ell_1$-norm defined as $\|A\|_1 := \sum_{i,j} |a_{ij}|$. In this case, the sparse CGL optimization objective can be cast as

$$\min_{A \in \mathbb{R}^{p \times p}} \quad \sum_{(i,j,k) \in \mathcal{T}} \left(1 - x_i^\top A x_j + x_i^\top A x_k\right)_+^2 + \lambda \|A\|_1 \tag{15}$$

Optimization (15) can be viewed as a generalization of LASSO (which is known to produce sparse solutions), except that we are using a pairwise squared hinge loss and that the coefficients are in matrix forms. Unlike in LASSO where one optimizes over a vector coefficient, the parameter space for (15) is a high-dimensional matrix of extremely large size.

Due to the presence of the $\ell_1$-norm in the objective function, our previous parameter reduction techniques for CGL can no longer be applied to sparse CGL. In the following, we are going to focus on directly carrying out optimization w.r.t. $A$. This is feasible because storing a large but highly sparse concept graph is much cheaper than in the dense case.

### 4.1 Efficient Optimization for Sparse-CGL

Although sub-gradient methods can be directly applied to minimizing the non-smooth objective in (15), they suffer from slow convergence rate through both theoretical and empirical perspectives. Commonly used optimization solvers for $\ell_1$ regularization such as the coordinate descent (CD) (Tseng & Yun, 2009; Chang, Hsieh, & Lin, 2008) no longer has the closed-form solution in our case for each sub-step, and needs as many as $p^2$ steps to go over even a single cycle of all the model parameters in $A$.

#### 4.1.1 Proximal Gradient Descent

Among the family of first-order methods, proximal gradient descent (PGD) has been widely applied to objective functions involving non-smooth components. It enjoys several desir-

able computational properties, including having the same order of convergence rate as the gradient descent even when applied to non-smooth objective functions. Each updating step of PGD can be efficiently performed as long as the proximal operation is efficient.

The proximal operator for sparse CGL in (15) is defined as

$$\text{prox}_{t_k}(A) := \underset{Z \in \mathbb{R}^{p \times p}}{\text{argmin}} \quad \frac{1}{2t_k}\|A - Z\|_2^2 + \lambda \|Z\|_1 \tag{16}$$

where $t_k$ is the step size for the $k$-th iteration. The solution for optimization (16) can be expressed concisely in a closed-form:

$$\text{prox}_{t_k}(A) = S_{\lambda t_k}(A) \tag{17}$$

where $S_\tau : \mathbb{R}^{p \times p} \mapsto \mathbb{R}^{p \times p}$ is known as the soft-thresholding operator (parameterized by $\tau$) applied to each element in $A$. That is, $\forall i, j$

$$[S_\tau(A)]_{ij} = \begin{cases} A_{ij} - \tau & A_{ij} > \tau \\ 0 & -\tau \leq A_{ij} \leq \tau \\ A_{ij} + \tau & A_{ij} < -\tau \end{cases} \tag{18}$$

With the proximal operator $\text{prox}_{t_k}(A) \equiv S_{\lambda t_k}(A)$, PGD iteratively applies

$$A^{(k)} = \text{prox}_{t_k}\left(A^{(k-1)} - t_k \nabla g(A^{(k-1)})\right) = S_{\lambda t_k}\left(A^{(k-1)} - t_k \nabla g(A^{(k-1)})\right) \tag{19}$$

until the model converges. In the above expression $\nabla g(A)$ denotes the gradient of the first term in optimization (15), which can be further recast as

$$\nabla g\left(A^{(k-1)}\right) = -2 \sum_{(i,j,k) \in \mathcal{T}} \delta_{ijk} x_i (x_j - x_k)^\top \tag{20}$$

$$= -2X^\top \left( \sum_{(i,j,k) \in \mathcal{T}} \delta_{ijk} e_i (e_j - e_k)^\top \right) X \tag{21}$$

$$= -2X^\top \Delta X \tag{22}$$

For sparse CGL, PGD is guaranteed to reach the global optimal since both the smooth and non-smooth components in the objective function (15) are convex over $A$.

#### 4.1.2 PGD with Nesterov's Acceleration

Similar to the gradient descent for CGL discussed in previous section 3.2.1, the convergence rate of PGD for sparse-CGL can be further accelerated by applying the Nesterov's method. In this case, we replace (19) in PGD with

$$P^{(k)} = S_{\lambda t_k}\left(A^{(k-1)} - t_k \nabla g\left(A^{(k-1)}\right)\right) \tag{23}$$

$$A^{(k)} = P^{(k)} + \frac{k-1}{k+2}\left(P^{(k)} - P^{(k-1)}\right) \tag{24}$$

**Algorithm 2** Sparse CGL.Rank with Accelerated PGD

**procedure** SPARSE.CGL.RANK($X, T, \lambda, \eta$)
  $A \leftarrow 0_{p\times p}$, $P \leftarrow 0_{p\times p}$, $Q \leftarrow 0_{p\times p}$
  $k \leftarrow 1$
  **while** not converge **do**
    $\Delta \leftarrow 0_{n\times n}$
    $F \leftarrow XAX^\top$
    **for** $(i, j, k)$ **in** $T$ **do**
      $\tilde{\delta}_{ijk} \leftarrow 1 - F_{ij} + F_{ik}$
      **if** $\tilde{\delta}_{ijk} > 0$ **then**
        $\Delta_{ij} \leftarrow \Delta_{ij} + \tilde{\delta}_{ijk}$
        $\Delta_{ik} \leftarrow \Delta_{ik} - \tilde{\delta}_{ijk}$
    **for** $j = 1, 2 \dots p$ **do**
      $P_j = S_{\lambda t_k}\left(A_j + 2t_k X^\top \Delta X_j\right)$
    $A \leftarrow P + \frac{k-1}{k+2}(P - Q)$
    $Q \leftarrow P$
    $k \leftarrow k + 1$
  **return** $A$

where we use $P \in \mathbb{R}^{p\times p}$ to capture the "momentum" information from historical iterations. As gradient descent and PGD, the convergence of accelerated PGD is guaranteed for sparse CGL but with a substantially faster rate.

One might be concerned that in both (19) and (23), the gradient $\nabla g\left(A^{(k-1)}\right)$ is a dense matrix by definition (20) and therefore can be expensive in memory consumption throughout the optimization. To address this issue, recall the soft-thresholding $S_{\lambda t_k}$ operator is applied element-wisely to its input. Let $P_i^{(k)}$ be the $i$-th column of $P^{(k)}$, we have

$$P_i^{(k)} = S_{\lambda t_k}\left[A_i^{(k-1)} - t_k \nabla_g \left(A^{(k-1)}\right)_i\right] \tag{25}$$

$$= S_{\lambda t_k}\left(A_i^{(k-1)} + 2t_k X^\top N X_i\right) \tag{26}$$

The above suggests sequentially threshold $A^{(k-1)} - t_k \nabla g\left(A^{(k-1)}\right)$ via $S_{\lambda t_k}$ column-by-column, and only store the resulting sparse column vectors (the $P_i^{(k)}$'s) without storing $\nabla_g\left(A^{(k-1)}\right)$. Details of accelerated PGD for sparse CGL.Rank is summarized in Alg. 2.

## 5. Transductive Concept Graph Learning

In real scenarios, the observed course-level prerequisite links are highly sparse. For example, only 1,173 out of 2,694,681 (0.043%) all possible links has been observed in the MIT data collection. Meanwhile, we notice that the features of unlabeled course-level links are already given, and are massively available during the training phase. We argue that transductive learning should be particular effective in this case for the following reasons:

1. It helps better leverage the unlabeled data by allowing the information to propagate through both the labeled and unlabeled course pairs.

2. It makes weaker assumptions about the unlabeled (missing) links. This is in contrast to our previous CGL formulation where all unobserved links are implicitly treated as negative examples.

To derive transductive CGL, we start with the following equivalent form of CGL, which can be derived by eliminating $A$ in the original CGL optimization via the constraint $F = XAX^\top$.

$$\min_{F \in \mathbb{R}^{n \times n}} \quad \sum_{(i,j,k) \in T} (1 - F_{ij} + F_{ik})_+^2 + \frac{\lambda}{2} vec(F)^\top (K \otimes K)^{-1} vec(F) \tag{27}$$

By viewing the second term in (27) as the negative log-likelihood, we see that the original CGL formulation essentially assumes $vec(F)$ is sampled from a Gaussian prior distribution with covariance matrix $K \otimes K$ where $K = XX^\top$.

To allow transduction over both the labeled and unlabeled course-level links, we propose to replace the inverse pairwise kernel matrix $K \otimes K$ in (27) with its associated graph Laplacian matrix $\mathcal{L}_\otimes$ in $\mathbb{R}^{n^2 \times n^2}$ (Chung, 1997), following the previous work on label propagation (Zhu, 2005) and spectral kernel design (Zhang & Ando, 2006). Formally, define

$$\mathcal{L}_\otimes := D_\otimes^{-\frac{1}{2}} \left( D_\otimes - K \otimes K \right) D_\otimes^{-\frac{1}{2}} \tag{28}$$

where $D_\otimes$ is an $n^2 \times n^2$ diagonal matrix with each of its diagonal element equal to the corresponding row-sum (degree) of $K \otimes K$. When the kernel matrix $K$ has been symmetrically normalized, it is not hard to show that the above definition (28) reduces to

$$\mathcal{L}_\otimes = I - K \otimes K \tag{29}$$

In this case, $vec(F)\mathcal{L}_\otimes vec(F)$ becomes the (normalized) manifold regularizer (Zhu, Ghahramani, & Lafferty, 2003) which enforces our predictions in $F$ to be smooth over the graph of course-level links with adjacency matrix $K \otimes K$. In particular, we have

$$vec(F)^\top \mathcal{L}_\otimes vec(F) \equiv \sum_{(i,j),(i',j')} k_{ii'} k_{jj'} \left( F_{i,j} - F_{i',j'} \right)^2 \tag{30}$$

Given two course-level links $(i, j)$ and $(i', j')$, recall that $k_{ii'}$ defines the similarity between $i$ and $i'$, $k_{jj'}$ denotes the similarity between $j$ and $j'$. Hence $k_{ii'}k_{jj'}$ denotes the similarity between $(i, j)$ and $(i', j')$, and minimizing (30) essentially enforces similar course-level links to share similar prerequisite strength.

With the intuitions above, we cast the optimization for transductive CGL as

$$\min_{F \in \mathbb{R}^{n \times n}} \quad \sum_{(i,j,k) \in T} (1 - F_{ij} + F_{ik})_+^2 + \frac{\lambda}{2} vec(F)^\top \mathcal{L}_\otimes vec(F) \tag{31}$$

### 5.1 Efficient Optimization over the Course-Level Links

It is worth mentioning that though the graph Laplacian matrix $\mathcal{L}_\otimes$ of course-level links is extremely large, it is also highly structured. As a result, we are able to carry out operations involving $\mathcal{L}_\otimes$ fairly efficiently. Moreover, we are going to show that the explicit storage of the full graph Laplacian can be avoided during the entire optimization.

**Algorithm 3** trans-CGL.Rank with accelerated GD

**procedure** CGL.Rank.Nestrerov($X, T, \lambda, \eta$)
  $K \leftarrow XX^\top$, $F \leftarrow \text{rand}(n, n)$, $Q \leftarrow 0_{n\times n}$
  $t \leftarrow 1$
  **while** not converge **do**
    $\Delta \leftarrow 0_{n\times n}$
    **for** $(i, j, k)$ **in** $T$ **do**
      $\tilde{\delta}_{ijk} \leftarrow 1 - F_{ij} + F_{ik}$
      **if** $\tilde{\delta}_{ijk} > 0$ **then**
        $\Delta_{ij} \leftarrow \Delta_{ij} + \tilde{\delta}_{ijk}$
        $\Delta_{ik} \leftarrow \Delta_{ik} - \tilde{\delta}_{ijk}$
    $P \leftarrow F - \eta\left(\lambda F - \lambda KFK - 2\Delta\right)$
    $F \leftarrow P + \frac{t-1}{t+2}\left(P - Q\right)$
    $Q \leftarrow P$
    $t \leftarrow t + 1$
  $A \leftarrow X^\top K^{-1} F K^{-1} X$
  **return** $A$

In the following, we describe how gradient computation can be carried out for trans-CGL. The gradient of the transductive CGL objective in (31) w.r.t. $F$ is

$$\nabla_F = \sum_{(i,j,k)\in T} \nabla\left(1 - F_{ij} + F_{ik}\right)_+^2 + \lambda vec^{-1}\left(\mathcal{L}_\otimes vec(F)\right) \tag{32}$$

$$= -2\sum_{(i,j,k)\in T} \delta_{ijk}\nabla\left(F_{ij} - F_{ik}\right) + \lambda vec^{-1}\left[\left(I - K\otimes K\right)vec(F)\right] \tag{33}$$

$$= -2\left[\sum_{(i,j,k)\in T} \delta_{ijk} e_i\left(e_j - e_k\right)^\top\right] + \lambda vec^{-1}\left(vec(F)\right) - \lambda vec^{-1}\left[\left(K\otimes K\right)vec(F)\right] \tag{34}$$

$$= -2\Delta + \lambda F - \lambda KFK \tag{35}$$

where $\Delta := \left[\sum_{(i,j,k)\in T} \delta_{ijk} e_i\left(e_j - e_k\right)^\top\right]$. In order to obtain the last equality, we have again applied the vectorization trick (Van Loan, 2000) for Tensor (Kronecker) product to the third term. Note that the expression of gradient in (35) doesn't involve any tensor-type operation, despite the huge Laplacian matrix $\mathcal{L}_\otimes \in \mathbb{R}^{n^2\times n^2}$ in the original objective function.

Second-order methods, such as the inexact Newton method are applicable to trans-CGL. We omit the details since the derivations are similar to that for CGL.

### 5.2 Projecting Back to the Concept-Level Links

The objective for transductive CGL (31) only involves the course-level prerequisite strength $F$ instead of the concept-level graph $A$. In order to recover $A^*$ from the optimal solution $F^*$ for (31), we consider solving the following optimization problem

$$\min_{A\in\mathbb{R}^{p\times p}} \quad \|A\|_2^2 \quad \text{subject to} \quad F^* = XAX^\top \tag{36}$$

| University | # Courses | # Prerequisites | # Words |
|---|---|---|---|
| MIT | 2322 | 1173 | 15396 |
| Caltech | 1048 | 761 | 5617 |
| CMU | 83 | 150 | 1955 |
| Princeton | 56 | 90 | 454 |

Table 1: Datasets Statistics

where the bilinear system $F^* = XAX^\top$ is under-determined as the total number of concepts $p$ is assumed to be greater than the number of courses $n$. While there can be multiple feasible concept graphs associated with $F^*$, (36) aims to pick the concept graph with minimum norm (which usually indicates strong generalization ability).

Solution for optimization (36) can be derived from its stationarity condition, and can be written in the closed-form as follows

$$A^* = X^\top K^{-1} F^* K^{-1} X \tag{37}$$

Details of the accelerated gradient descent for trans-CGL.Rank, including the recovery step for the concept graph, are summarized in Alg. 3.

## 6. Experiments

We collected course listings, including course descriptions and available prerequisite structure from `MIT OpenCourseWare`, Caltech, CMU and Princeton[2]. The first two were complete course catalogs, and the latter two required spidering and scraping, and hence we collected only Computer Science and Statistics for CMU, and Mathematics for Princeton. This implies that we can test within-university prerequisite discovery for all four—though MIT and Caltech will be most comprehensive—and cross-university only for university pairs where the training university contains the disciplines in the test university.

Table 1 summarizes the datasets statistics.

To evaluate performance, we use the *Mean Average Precision* (MAP) which has been the preferred metric in information retrieval for evaluating ranked lists, and the Area Under the Curve of ROC (ROC/AUC or simply AUC) which is popular in link detection evaluations.

### 6.1 Within-University Prerequisite Prediction

We tested all the methods on the dataset from each university. We used one third of the data for testing, and the remaining two thirds for training and validation. We conducted 5-fold cross validation on the training two-thirds, i.e., trained the model on 80% of the training/validation dataset, and tuned extra parameters on the remaining 20%. We repeated this process 5 times with a different 80-20% spit in each run. The results of the 5 runs were averaged in reporting results. Figure 3 and Table 2 summarize the results of CGL.Rank, CGL.Class, 1NN and SVM. All the methods used the English words as the representation scheme in this first set of experiments.

2. The datasets are available at `http://nyc.lti.cs.cmu.edu/teacher/dataset/`

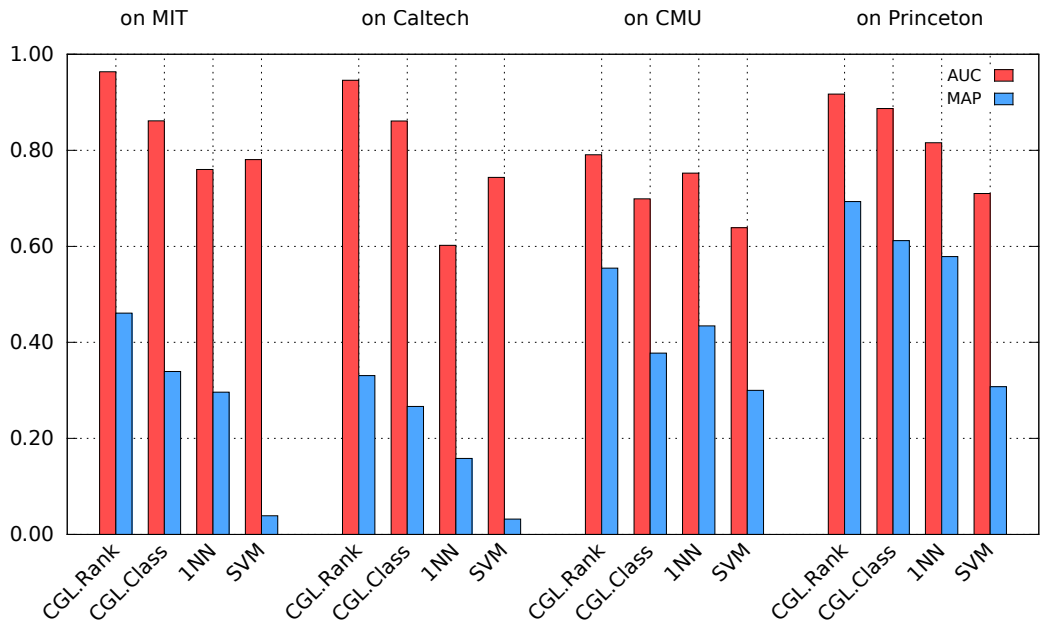


Figure 3: Different methods in within-university prerequisite prediction: All the methods used words as concepts.

| Algorithm | Data | AUC | MAP |
|---|---|---|---|
| CGL.Rank | MIT | 0.96 | 0.46 |
| CGL.Class | MIT | 0.86 | 0.34 |
| 1NN | MIT | 0.76 | 0.30 |
| SVM | MIT | 0.78 | 0.04 |
| CGL.Rank | Caltech | 0.95 | 0.33 |
| CGL.Class | Caltech | 0.86 | 0.27 |
| 1NN | Caltech | 0.60 | 0.16 |
| SVM | Caltech | 0.74 | 0.03 |
| CGL.Rank | CMU | 0.79 | 0.55 |
| CGL.Class | CMU | 0.70 | 0.38 |
| 1NN | CMU | 0.75 | 0.43 |
| SVM | CMU | 0.64 | 0.30 |
| CGL.Rank | Princeton | 0.92 | 0.69 |
| CGL.Class | Princeton | 0.89 | 0.61 |
| 1NN | Princeton | 0.82 | 0.58 |
| SVM | Princeton | 0.71 | 0.31 |

Table 2: Results of within-university prerequisite prediction using words as concepts.

Notice that the AUC scores for all methods are much higher than the MAP scores. The high AUC scores derive in large part from the fact that AUC gives an equal weight to the system-predicted true positives, regardless of their positions in system-produced ranked lists. On the other hand, MAP weighs more heavily the true positives in higher positions of ranked lists. In other words, MAP measures the performance of a system in a harder task: Not only the system needs to find the true positives (along with false positives), it also needs to rank them higher than false positives as possible in order to obtain a high MAP score. Using a more concrete example, a totally useless system which makes positive or negative predictions at random with 50% of the chances will have an AUC score of 50%. But this system will have an extremely low score in MAP because the chance for a true positive to randomly appear in the top of a ranked list will be low when true negatives dominate in the domain. Our datasets are from such a domain because each course only requires a very small number of other courses as prerequisites. Back to our original point, regardless the popularity of AUC in link detection evaluations, its limitation should be recognized: the relative performance among methods is more informative than the absolute values of AUC.

As we can see in Figure 3, the relative ordering of the methods in AUC and MAP are indeed highly correlated across all the datasets. MAP emphasizes positive instances in the top portion of each ranked list, and hence is more sensible for measuring the usefulness of the system where the user interacts with the system-recommended ranked list of prerequisites per query (a course in the test set).

Comparing the results of all the methods, we see that CGL.Rank clearly dominates the others in both AUC and MAP on most datasets. CGL.Class is the second best, outperforming 1NN and SVM. Comparing 1NN and SVM, these two methods are comparable in AUC on average; however, 1NN is better than SVM in MAP on all the four datasets.

One may wonder why SVM has the relatively poor performance in course-level prerequisite prediction and in particular under the MAP metric, given that it works well for text classification and learning to rank for retrieval in general. We argue that SVM is suffering from the major difficulty in prerequisite prediction due to the extremely sparse labeled positive training instances (prerequisite pairs). Recall that in text classification SVM optimizes one $w$ for each category; however, in prerequisite prediction SVM uses the same $w$ to generate the ranked lists for all possible queries (i.e., test-set courses). Thus the labeled training instances per query is extremely sparse on average, resulting in the poor performance we observed. In other words, SVM generalized too much from the extremely small set of labeled training instances when optimizing the global $w$. kNN (or 1NN) on the other hand, suffers less than SVM because the inference in kNN is based on the local training instances in the neighborhood of each query, instead of a global generalization for all queries.[3] Nevertheless, neither kNN nor SVM is competitive in comparison with our proposed CGL methods, which is evident in this set of evaluation results.

3. Notice that SVM, as described in (6), is essentially going to produce identical ranked list of prerequisite courses for any given course. To see this, consider any given course $i$, SVM scores the remaining courses by $score_j := (x_i - x_j)^\top w = x_i^\top w - x_j^\top w$ for different $j$'s. Since $i$ is given, the first term $x_i^\top w$ can be ignored during the ranking as it is shared across all the scores. As the result, the ranked list becomes irrelevant to $x_i$ itself. The above analysis holds for any course $i$ hence the ranked list of prerequisites will be identical for all the courses, leading to a low MAP score. In contrast, CGL scores course $j$ by $score_j = x_i^\top A x_j = (Ax_i)^\top x_j := w_i^\top x_j$ where the ranking coefficient $w_i$ is "personalized" for the $i$-th course, and hence is able to produce diverse ranked lists for different courses.

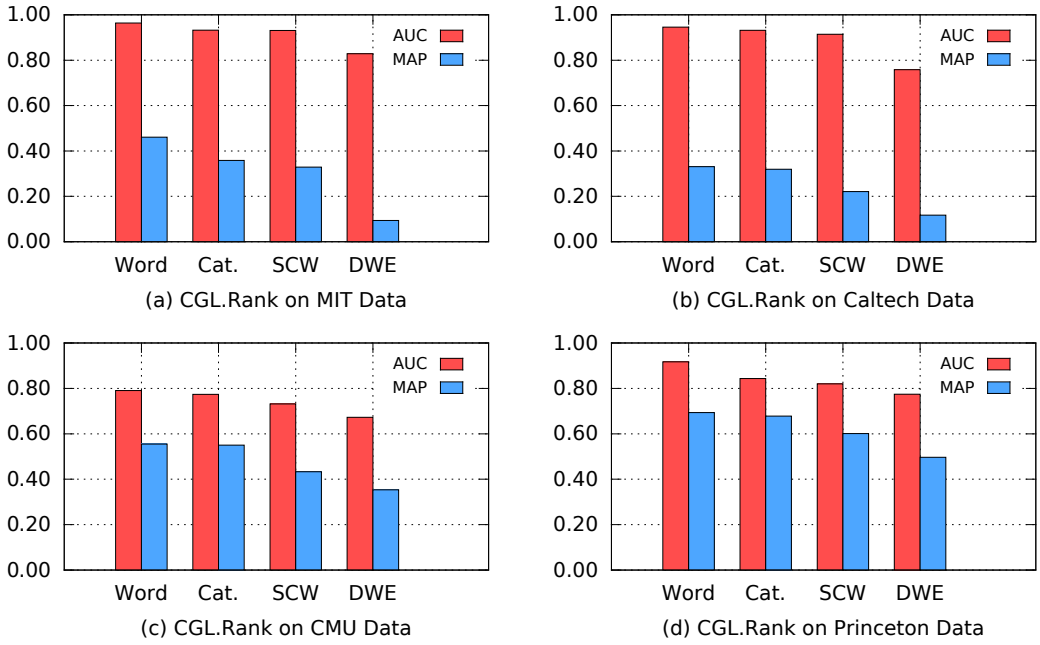


Figure 4: CGL.Rank with different representation schemes in within-university prerequisite prediction

### 6.2 Effects of Representation Schemes

Figure 4 and Table 3 summarize the results of CGL.Rank with the four representation schemes described in Section 2.1, i.e., *Word*, *Cat*, *SCW* (sparse coding of words) and *DWE* (distributed word embedding), respectively. Again the scores of AUC and MAP are not on the same scale, but the relative performance suggest that *Word* and *Cat* are competitive with each other (or *Word* is slightly better on some datasets), followed by *SCW* and then *DWE*. In the rest of the empirical results reported in this paper, we focus more on the performance of CGL.Rank with *Word* as the representation scheme because it performs better, and the space limit does not allow us to present all the results for every possible permutation of method, scheme, dataset and metric.

### 6.3 Cross-University Prerequisite Prediction

In this set of experiments, we fixed the same test sets which were used in within-university evaluations, but we alter the training sets across universities, yielding transfer learning results where the models were trained with the data from a different university than those where they were tested. By fixing the test sets in both within- and cross-university evaluations we can compare the results on a common basis. The competitive performance of *Cat* in comparison with *Word* is encouraging, given that Wikipedia categories are defined as general knowledge, and the classifiers (SVM's) we used for category assignment to courses were trained on Wikipedia articles instead of the course materials (because we do not have

| Concept | Data | AUC | MAP |
|---|---|---|---|
| Word | MIT | 0.96 | 0.46 |
| Cat. | MIT | 0.93 | 0.36 |
| SCW | MIT | 0.93 | 0.33 |
| DWE | MIT | 0.83 | 0.09 |
| Word | Caltech | 0.95 | 0.33 |
| Cat. | Caltech | 0.93 | 0.32 |
| SCW | Caltech | 0.91 | 0.22 |
| DWE | Caltech | 0.76 | 0.12 |
| Word | CMU | 0.79 | 0.55 |
| Cat. | CMU | 0.77 | 0.55 |
| SCW | CMU | 0.73 | 0.43 |
| DWE | CMU | 0.67 | 0.35 |
| Word | Princeton | 0.92 | 0.69 |
| Cat. | Princeton | 0.84 | 0.68 |
| SCW | Princeton | 0.82 | 0.60 |
| DWE | Princeton | 0.77 | 0.50 |

Table 3: CGL.Rank with four representations

human assigned Wikipedia category labels for courses). This means that the Wikipedia categories indeed have a good coverage on the concepts being taught in universities (and probably MOOC courses), and that our pooling strategy for selecting a relevant subset of Wikipedia categories is reasonably successful.

Table 4 and Figure 5 show the results of CGL.Rank (using words as concepts). Recall that the MIT and Caltech data cover the complete course catalogs, while the CMU data only cover the Computer Science and Statistics, and the Princeton data only over the Mathematics. This implies that we can only measure transfer learning on pairs where the training university contains the disciplines in the test university. By comparing the red bars (when the training university and the test university is the same) and the blue bars (when the training university is not the same as the test university), we see some performance loss in the transfer learning, which is expected. Nevertheless, we do get transfer, and this is the first report on successful transfer learning of educational knowledge, especially the prerequisite structures in disjoint graphs, across different universities through a unified concept graph. The results are therefore highly encouraging and suggest continued efforts to improve. Those results also suggest some interesting points, e.g., MIT might have a better coverage of the topics taught in Caltech, compared to the inverse. And, MIT courses seem to be closer to those in Princeton (Math) compared with those of CMU.

### 6.4 CGL vs Sparse-CGL

In this subsection we evaluate the performance of CGL (CGL.Rank) and sparse-CGL (Section 4) in course-level prerequisite prediction. The two methods were compared under *budget constraints*, by which we mean that the number of allowed links in the system-induced concept graph was controlled as the condition for comparison. In other words, we control the

| Training | Test | MAP | AUC |
|---|---|---|---|
| MIT | MIT | 0.46 | 0.96 |
| Caltech | MIT | 0.13 | 0.88 |
| Caltech | Caltech | 0.33 | 0.95 |
| MIT | Caltech | 0.25 | 0.86 |
| CMU | CMU | 0.55 | 0.79 |
| MIT | CMU | 0.34 | 0.70 |
| Caltech | CMU | 0.28 | 0.62 |
| Princeton | Princeton | 0.69 | 0.92 |
| MIT | Princeton | 0.46 | 0.72 |
| Caltech | Princeton | 0.43 | 0.58 |

Table 4: CGL.Rank in within-university and cross-university settings

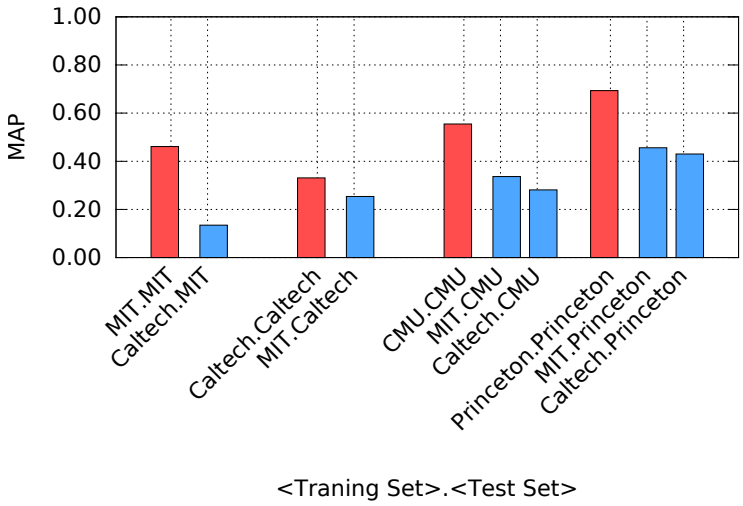


Figure 5: Results of CGL.Rank based on words in the tasks of within-university (red) and cross-university (blue) prerequisite prediction.

graph sparsity by varying the number of allowed non-zero elements in $A$ [4], and compare the performance of the two methods conditioned on each fixed degree of graph sparsity.

Figure 6 compares the performance of the two methods on the datasets of MIT, Caltech, CMU and Princeton, in the task of within-university prediction of course-level prerequisite relations. The horizontal axis of each graph specifies the number of non-zero elements allowed in matrix $A$, ranging from $2^7$ to $2^{12}$. The vertical axis in each graph is MAP on

4. The concept graph induced by CGL is typically dense, which was sparsified (given the desired sparsity) by keeping the most dominating elements of $A$ and setting the remaining elements to zero. With sparse-CGL, on the other hand, sparse graphs were directly induced by the optimization algorithm by adjusting the values of $\ell_1$-regularization strength $\lambda$.

the left and AUC on the right for each data set. The CGL curves are in blue, and the sparse-CGL curves are in red.

Clearly, sparse-CGL consistently outperformed CGL in these experiments over most budget regions on all the datasets, both in MAP and AUC. These results are highly encouraging for effective and efficient interaction or navigation by users over the system-induced concept graph, where an interpretable and relatively sparse graph is often preferable than a densely connected graph. The budget constrained graphs would also lead to scalable curriculum planning and fast computation for on-demand recommendation.

### 6.5 CGL vs Trans-CGL

In this subsection we compare the course-level prerequisite prediction performance of CGL (in particular, CGL.Rank) against its transductive extension. The two methods are tested over the aforementioned four datasets under the words representation scheme. All experiments are conducted under the same 5-fold cross-validation setting as described in 6.1. We use MAP and AUC as our evaluation metrics and summarize the results in Table 5.

From Table 5 we see that the link prediction performance of trans-CGL dominates that of CGL. This justifies our previous arguments that making a good use of massive unlabeled course-level links provided in the training set can help us get better prediction performance.

Meanwhile, we notice that the performance gain obtained by trans-CGL over MIT is not as large as that over other three institutions. Since the MIT dataset has substantially larger amount of course-level labels, we conjecture that trans-CGL, as many other transductive/semi-supervised learning approaches in general, is more advantageous when the available supervision is insufficient (compared to the amount of hidden information in the unlabeled data). To validate this thought, we repeat the experiments of the two methods over the *down-sampled* MIT dataset. That is, we only use a random subset of available course-level prerequisite links for training, and gradually vary the size of the training subset to get multiple set of results. The results are summarized in Table 6.

| | Institution | MIT | Caltech | CMU | Princeton |
|---|---|---|---|---|---|
| MAP | CGL | 0.482 | 0.477 | 0.482 | 0.436 |
| | trans-CGL | **0.485** | **0.499**$^\dagger$ | **0.539**$^\dagger$ | **0.445**$^\dagger$ |
| AUC | CGL | 0.956 | 0.929 | 0.801 | 0.634 |
| | trans-CGL | **0.957** | **0.941**$^\dagger$ | **0.818**$^\dagger$ | **0.67**$^\dagger$ |

Table 5: Comparison of the course prerequisite prediction performance between CGL and trans-CGL over MIT, Caltech, CMU and Princeton. Results of the significance tests (paired t-tests) between the best method against the other method on each dataset is denoted by a † for significance at 1% level.

### 6.6 Experiment Details

We tested the efficiency of our proposed algorithms (based on the optimization formulation after variable reduction) on a single machine with an Intel i7 8-core processor and 32GB

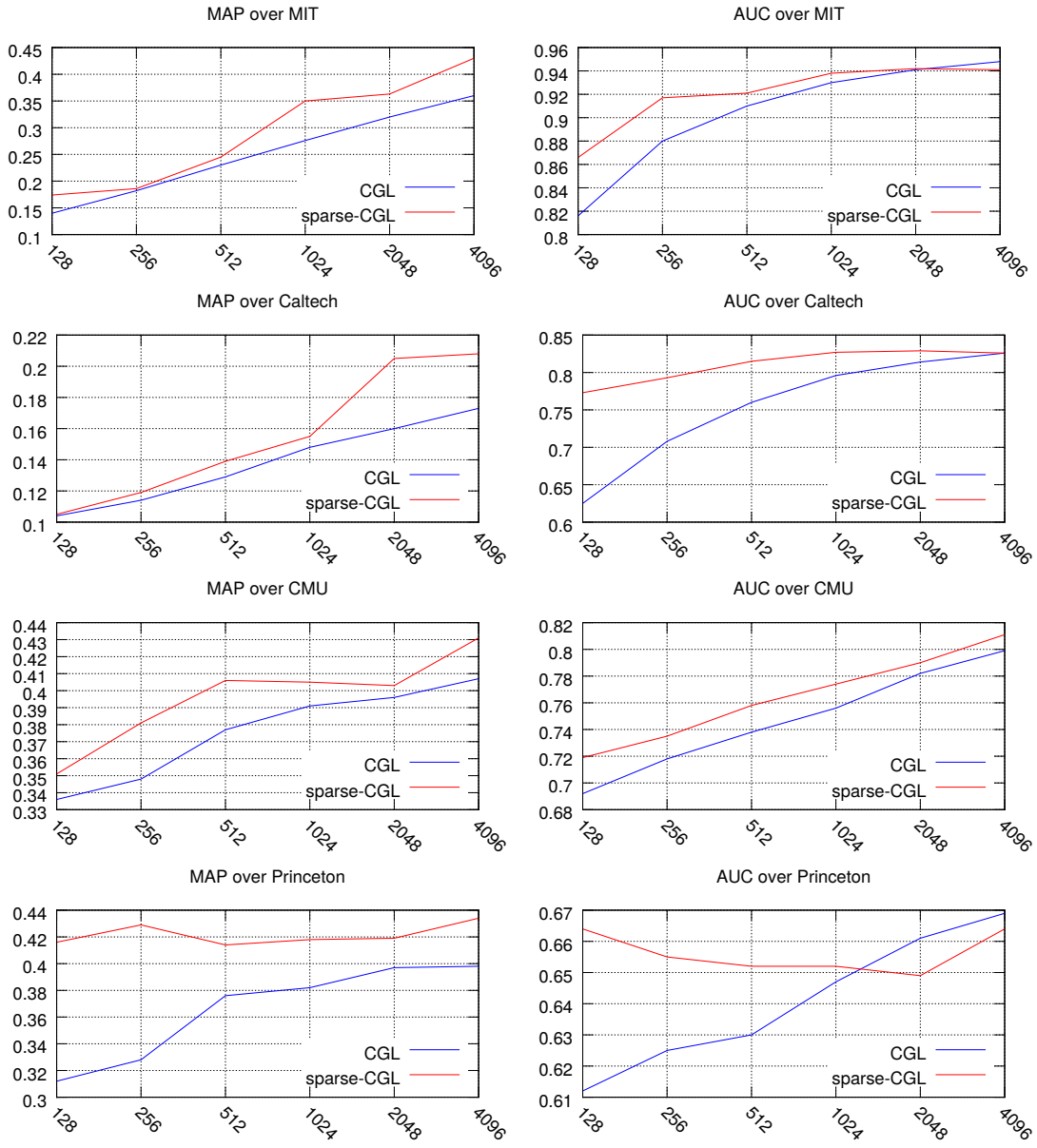


Figure 6: Comparison of CGL (blue curves) and sparse-CGL (red curves) in the prediction of course-level prerequisite relations within MIT, Caltech, CMU and Princeton, respectively. The x-axis of each graph specifies the budget constraint in log-scale, which is the number of allowed non-zero elements in matrix $A$. The y-axis measures the performance in MAP on the left, and in AUC on the right.

| | Training Data Down-sampling Rate | $\frac{1}{10}$ | $\frac{1}{9}$ | $\frac{1}{8}$ | $\frac{1}{7}$ | $\frac{1}{6}$ | $\frac{1}{5}$ | $\frac{1}{4}$ | $\frac{1}{3}$ | $\frac{1}{2}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| MAP | CGL | 0.183 | 0.149 | 0.192 | 0.231 | 0.245 | 0.264 | 0.254 | 0.258 | 0.289 |
| | trans-CGL | **0.202** | **0.153** | **0.213** | **0.249** | **0.26** | **0.285** | **0.262** | **0.274** | **0.307** |
| AUC | CGL | 0.843 | 0.832 | 0.842 | 0.872 | **0.873** | 0.876 | **0.873** | 0.897 | 0.905 |
| | trans-CGL | **0.874** | **0.838** | **0.845** | **0.875** | 0.872 | **0.892** | 0.868 | **0.91** | **0.914** |

Table 6: Comparison of the course prerequisite prediction performance between CGL and trans-CGL over the MIT dataset, as the training size varies from 10% to 50%.

RAM. On the largest MIT dataset with 981,009 training triplets and 490,505 test triplets, CGL.Rank with gradient descent took 37.3 minutes and 1490 iterations to reach the convergence rate of $10^{-3}$. To achieve the same objective value, the accelerated gradient descent took 3.08 minutes with 401MB memory at 103 iterations, and the inexact Newton method took only 43.4 seconds with 587MB memory. CGL.Class is equally efficient as CGL.Rank in terms of run time, though the latter is superior in terms of result quality. As for our 1NN baseline, it took 2.88 hours since a huge number ($2 \times 981,009 \times 490,505$) of dot-products need to be computed on the fly.

The CPU time consumed by trans-CGL.Rank is similar to that by CGL.Rank. As for sparse-CGL, it took the accelerated proximal gradient method 2.07 minutes to reach the convergence rate of $10^{-3}$ on MIT with 3.9GB peak memory consumption. The resulting concept graph has 1519 nonzero entries.

## 7. Discussion and Related Work

Whereas the task of inferring prerequisite relations among courses and concepts, and the task of inferring a concept network from a course network in order to transfer learned relations are both new, as are the extensions to the SVM algorithms presented, our work was inspired by methods in related fields, primarily:

In collaborative filtering via matrix completion the literature has focused on only one graph, such as a bipartite graph of $u$ users and $m$ preferences (e.g. for movies or products). Given some known values in the $u$-by-$m$ matrix, the task is to estimate the remaining values (e.g. which other unseen movies or products would each user like) (Su et al., 2009). This is done via methods such as affine rank minimization (Fazel, 2002) that reduce to convex optimization (Boyd & Vandenberghe, 2004).

Another line of related research is transfer learning (Do & Ng, 2005; Yang, Hanneke, & Carbonell, 2013; Zhang, Ghahramani, & Yang, 2008). We seek to transfer prerequisite relations between pairs of courses within universities to other pairs also within universities, and to pairs that span universities. This is inspired by but different from the transfer learning literature. Transfer learning traditionally seeks to transfer informative features, priors, latent structures and more recently regularization penalties (Kshirsagar, Carbonell, & Klein-Seetharaman, 1990). Instead, we transfer shared concepts in the mappings between course-space and concept-space to induce prerequisite relations.

Although our evaluations primarily focus on detecting prerequisite relations among courses, such a task is only one direct application of the automatically induced univer-

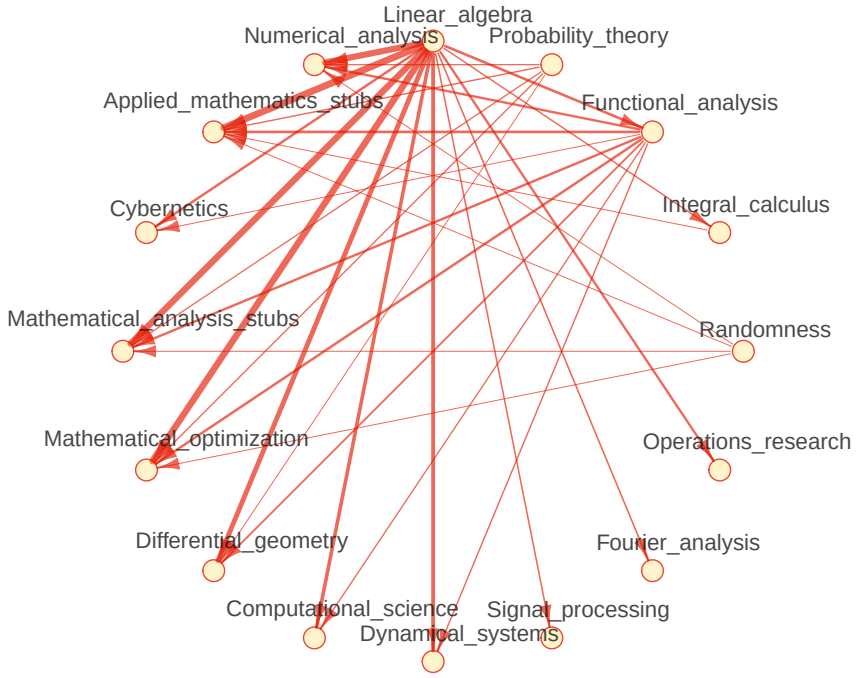


Figure 7: A visualization of concept graph produced by CGL.Rank based on 10,051 academic Wikipedia categories, 2322 courses and 1173 prerequisite relations in MIT OpenCourseWare. Each node denotes a concept (i.e. a Wikipedia category), and the strength of each link encodes the prerequisite strength between a pair of concepts. Concepts of small degrees and links with weak strength are removed via thresholding for visualization purposes.

sal concept graph. Other important applications include automated or semi-automated curriculum planning for personal education goals based on different backgrounds of students, and modularization in course syllabus design by instructors. Both tasks require the interaction between humans (students or teachers) and the system-induced concept graph as well as the system-recommended options and optimal sequences of ordered courses. Figure 7 visualizes a sub-graph in the system-induced concept space based on the course materials (including partially observed prerequisite structure) from `MIT OpenCourseWare`: the nodes are Wikipedia categories (concepts), and the links are system-predicted partial orders for instructors to follow in teaching those concepts, or for students to follow in learning those concepts. As one can see, *Linear Algebra*, *Probability Theory* and *Functional Analysis* are the "hubs" (with a high degree of out-links) in the graph, indicating that those concepts are more fundamental for one to acquire before pursuing more advanced topics such as *Differential Geometry*, *Cybernetics* and *Fourier Analysis*.

Let us further illustrate the potential use of CGL.Rank and its further development for the problem of inferring prerequisite relationship when observed prerequisites among courses are (almost) not available, e.g., in the context of MOOC. Recall that MOOC courses are

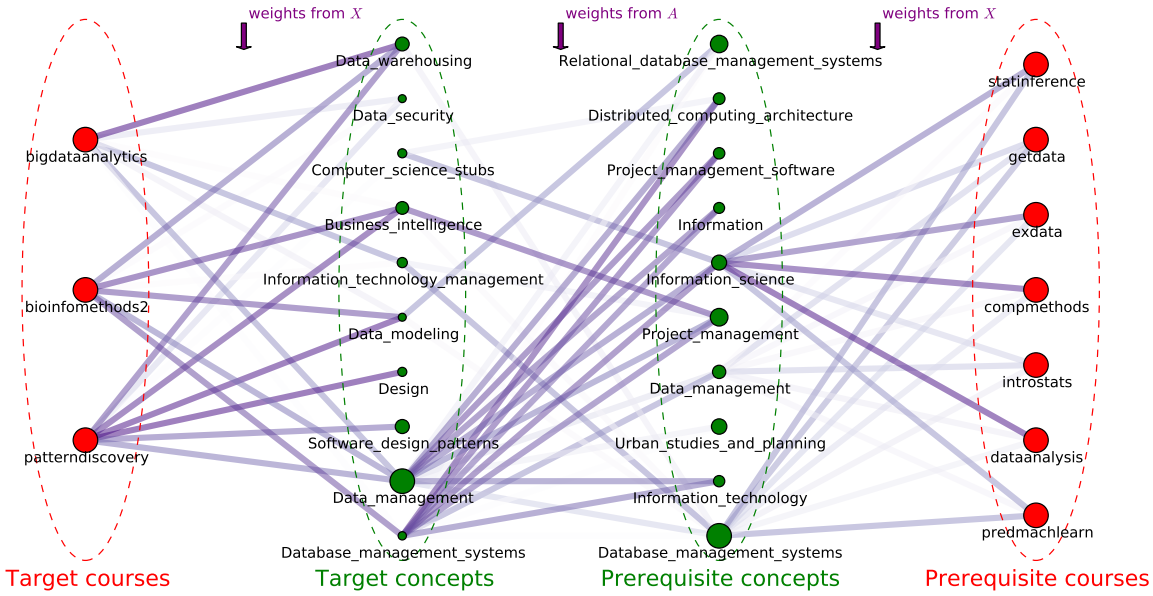


Figure 8: An example of prerequisite course recommendation on `Coursera` using the concept graph learned from the `MIT OpenCourseWare` dataset. We map the courses (red, left) that the student wants to learn to the concept space of Wikipedia categories, find the prerequisite concepts and then map back to `Coursera` courses (red, right). The sizes of the concept nodes in the middle (green) are proportional to aggregated weights of the corresponding links, and the strengths of course-concept mapping and concept-concept prerequisite relations are shown by the color intensity of edges (purple).

offered by different institutions across the world, where cross-provider prerequisite links among courses are not explicitly available. For instance, among the 900+ courses that `Coursera` currently offers, about a half of them do not mention anything about the required background; as for the remaining half, the mentioned background requirements are often vague, e.g., "Undergraduate-level networking know-how is recommended" for the course of *Cloud Networking*. Such overly generic and vague notions are not sufficiently informative for students to understand the true prerequisite relationship among courses, and also not very useful for intelligent systems to learn the prediction of latent prerequisites automatically. [5]

The example in Figure 8 illustrates the key idea of applying the CGL.Rank algorithm to address the problems above. The first column on the left consists of three `Coursera` courses that a student wants to take, i.e., "Big Data Analytics for Healthcare" (*bigdata-analytics*), "Bioinformatic Methods II"(*bioinfomethods2*) and "Pattern Discovery in Data Mining"(*patterndiscovery*), respectively. The second column of nodes are the top-10 universal concepts (Wikipedia categories) assigned by our classifiers (Section 2.1 *Cat*) to these courses, and the color intensity of the edges between the 1st and the 2nd columns reflects

5. `Coursera` does offer so-called *specializations*, where each specialization consists of a sequence of courses on the same topic. It is unclear whether those specializations may serve as prerequisite relations.

the confidence scores (matrix $X$) of category assignments. The size of each concept node is proportional to the aggregated confidence scores of the corresponding edges, indicating the relative importance of that concept. The third column of nodes are the top-10 prerequisite concepts (also from Wikipedia categories) of the concepts shown in the 2nd column, and the color intensity of the edges between the 2st and 3nd columns reflects the automatically induced strengths of concept-level links (Matrix $A$) by our CGL.Rank algorithm, which used the `MIT OpenCourseWare` data as the training set. The nodes in the 4th column are the `Coursera` courses that best cover the prerequisite concepts in the 3rd column; the edges from the 3rd column are similar to that between the 1st and 2nd columns. Together, the chained network allows us to make an inference about the connections between the target courses (the left-most column) and the prerequisite courses (the right-most column).

As the reader can see in Figure 8, many of the courses and prerequisites are highly relevant and focus on the primary dimension of data analytics, but still constitute an incomplete set as the second dimension of biotechnology is not represented. We are in the process of refining our methods before thorough testing can be performed—we offer this example as work-in-progress indicating current challenges and future directions.

## 8. Concluding Remarks

We conducted a new investigation on automatically inferring directed graphs at both the course level and the concept level, to enable prerequisite prediction for within-university and cross-university settings.

We proposed three approaches: a classification approach (CGL.Class), a learning to rank approach (CGL.Rank), and a nearest-neighbor search approach (kNN). Both CGL.Class and CGL.Rank (deploying adapted versions of SVM algorithms) explicitly model concept-level dependencies through a directed graph, and support an interlingua-style transfer learning across universities, while kNN makes simpler prediction without learning concept dependencies. To tackle the extremely high-dimensional optimization in our problems (e.g., $2 \times 10^8$ links in the concept graph for the MIT courses), our novel reformulation of CGL.Rank enables the deployment of fast numerical solutions. On our newly collected datasets from MIT, Caltech, CMU and Princeton, CGL.Rank proved best under MAP and ROC/AUC, and computationally much more efficient than kNN.

We extended the aforementioned CGL algorithm to further produce a sparse concept graph based on $\ell_1$-regularization (sparse-CGL), and to leverage the information in massive unlabeled course pairs based on graph-regularization (trans-CGL). We developed scalable optimization strategies in support of these new formulations, and conducted experiments which empirically showed that sparse-CGL was able to give a more interpretable concept graph, and that trans-CGL significantly and consistently improved the performance of the ordinary CGL in course prerequisite prediction.

We also tested four representation schemes for course content: using the original words, using Wikipedia categories as concepts, using a distributed word representation, and using sparse word encoding. The first two: original words and Wikipedia-derived concepts proved best. Our results in both the within- and cross-university settings are highly encouraging.

We envision that the cross-university transfer learning of our approaches is particularly important for MOOCs where courses come from different providers and across institutions, where there are seldom any explicit prerequisite links. A rich suite of future work includes:

- Testing on cross-institution or cross-course-provider prerequisite links. We have tested cross-university transfer learning, but the inferred links are within each target university, rather than cross-institutional links. A natural extension of the current work is to predict cross-institutional prerequisites. For evaluation we will need labeled ground truth of cross-institutional prerequisites.
- Cross-language transfer. Using the Wikipedia categories and entries in different languages, it would be an interesting challenge to infer prerequisite relations for courses in different languages by mapping to the Wikipedia category/concept interlingua.
- Extensions of the inference from single source to multiple sources, from single media (text) to multiple media (including videos), and from single granularity level (courses) to multiple levels (including lectures).
- Deploying the induced concept graph for personalized curriculum planning for students (as in Section 7) and for syllabus design and course modularization by teachers.

## Acknowledgments

We thank the reviewers for their helpful comments. This work is supported in part by the National Science Foundation under grants IIS-1216282, IIS-1350364 and IIS-1546329.

## References

MIT OpenCourseWare. http://ocw.mit.edu/index.htm. Accessed: 2016-03-31.

Airola, A., Pahikkala, T., & Salakoski, T. (2011). Training linear ranking SVMs in linearithmic time using red–black trees. *Pattern Recognition Letters*, *32*(9), 1328–1336.

Al-Rfou, R., Perozzi, B., & Skiena, S. (2013). Polyglot: Distributed word representations for multilingual nlp. *CoNLL-2013*, 183.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Candès, E. J., & Recht, B. (2009). Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, *9*(6), 717–772.

Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2008). Coordinate descent method for large-scale l2-loss linear support vector machines. *The Journal of Machine Learning Research*, *9*, 1369–1398.

Chen, Y., Perozzi, B., Al-Rfou, R., & Skiena, S. (2013). The expressive power of word embeddings. *ICML 2013 Workshop on Deep Learning for Audio, Speech, and Language Processing*.

Chung, F. R. K. (1997). *Spectral graph theory*, Vol. 92. American Mathematical Soc.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural Language Processing (almost) from Scratch. *The Journal of Machine Learning Research*, *12*, 2493–2537.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2001). *Introduction to algorithms*. MIT Press and McGraw-Hill.

Davie, A. M., & Stothers, A. J. (2013). Improved bound for complexity of matrix multiplication. *Proceedings of the Royal Society of Edinburgh: Section A Mathematics*, *143*(02), 351–369.

Dembo, R. S., Eisenstat, S. C., & Steihaug, T. (1982). Inexact Newton methods. *SIAM Journal on Numerical analysis*, *19*(2), 400–408.

Do, C., & Ng, A. Y. (2005). Transfer learning for text classification. In *Proceedings of NIPS-05*.

Fazel, M. (2002). *Matrix rank minimization with applications*. Ph.D. thesis, Stanford University.

Gopal, S., & Yang, Y. (2013). Recursive regularization for large-scale classification with hierarchical and graphical dependencies. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 257–265. ACM.

Hoyer, P. O. (2004). Non-negative matrix factorization with sparseness constraints. *The Journal of Machine Learning Research*, *5*, 1457–1469.

Joachims, T., Li, H., Liu, T.-Y., & Zhai, C. (2007). Learning to rank for information retrieval (LR4IR 2007). In *SIGIR Forum*, Vol. 41, pp. 58–62.

Johnson, C. R. (1990). Matrix completion problems: a survey. In *Proceedings of Symposia in Applied Mathematics*, Vol. 40, pp. 171–198.

Kim, H., & Park, H. (2008). Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. *SIAM Journal on Matrix Analysis and Applications*, *30*(2), 713–730.

Kshirsagar, M., Carbonell, J., & Klein-Seetharaman, J. (1990). Transfer learning based methods towards the discovery of host-pathogen protein-protein interactions. In *Proc of ISMB*, Vol. 40, pp. 171–198.

Kunegis, J., & Lommatzsch, A. (2009). Learning spectral graph transformations for link prediction. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 561–568. ACM.

Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1188–1196.

Lee, C.-P., & Lin, C.-J. (2014). Large-scale linear rankSVM. *Neural computation*, *26*(4), 781–817.

Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, *401*(6755), 788–791.

Liben-Nowell, D., & Kleinberg, J. (2007). The link-prediction problem for social networks. *Journal of the American society for information science and technology*, *58*(7), 1019–1031.

Lichtenwalter, R. N., Lussier, J. T., & Chawla, N. V. (2010). New perspectives and methods in link prediction. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 243–252. ACM.

Nesterov, Y. (1983). A method of solving a convex programming problem with convergence rate o (1/k2). In *Soviet Mathematics Doklady*, Vol. 27, pp. 372–376.

Nesterov, Y. (1988). On an approach to the construction of optimal methods of minimization of smooth convex functions. *Ekonomika i Mateaticheskie Metody*, *24*, 509–517.

Rendle, S., Freudenthaler, C., Gantner, Z., & Schmidt-Thieme, L. (2009). BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pp. 452–461. AUAI Press.

Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in artificial intelligence*, *2009*, 4.

Tseng, P., & Yun, S. (2009). A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, *117*(1-2), 387–423.

Van Loan, C. F. (2000). The ubiquitous Kronecker product. *Journal of computational and applied mathematics*, *123*(1), 85–100.

Williams, C., & Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In *Proceedings of the 14th Annual Conference on Neural Information Processing Systems*, No. EPFL-CONF-161322, pp. 682–688.

Yang, L., Hanneke, S., & Carbonell, J. (2013). A theory of transfer learning with applications to active learning. *Machine learning*, *90*(2), 161–189.

Yang, Y., Liu, H., Carbonell, J., & Ma, W. (2015). Concept graph learning from educational data. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pp. 159–168. ACM.

Zhang, J., Ghahramani, Z., & Yang, Y. (2008). Flexible latent variable models for multi-task learning. *Machine Learning*, *73*(3), 221–242.

Zhang, T., & Ando, R. (2006). Analysis of spectral kernel design based semi-supervised learning. *Advances in neural information processing systems*, *18*, 1601.

Zhu, X. (2005). Semi-supervised learning literature survey. Tech. rep. TR 1530, University of Wisconsin - Madison.

Zhu, X., Ghahramani, Z., & Lafferty, J. (2003). Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of ICML-03*, Vol. 3, pp. 912–919.