Matrices, Compression, Learning Curves: formulation, and the GROUPNTEACH algorithms

Bryan Hooi[1], Hyun Ah Song[1], Evangelos Papalexakis[1], Rakesh Agrawal[2], and Christos Faloutsos[1]

[1] Carnegie Mellon University, Pittsburgh, PA 15213, USA,
bhooi@andrew.cmu.edu, {hyunahs,epapalex,christos}@cs.cmu.edu
[2] Data Insights Laboratories,
ragrawal@acm.org

Abstract. Suppose you are a teacher, and have to convey a set of object-property pairs ('lions eat meat'). A good teacher will convey a lot of information, with little effort on the student side. What is the best and most intuitive way to convey this information to the student, without the student being overwhelmed? A related, harder problem is: how can we assign a numerical score to each lesson plan (i.e., way of conveying information)? Here, we give a **formal definition** of this problem of forming learning units and we provide a **metric** for comparing different approaches based on information theory. We also design an **algorithm**, GROUPNTEACH, for this problem. Our proposed GROUPNTEACH is *scalable* (near-linear in the dataset size); it is *effective*, achieving excellent results on real data, both with respect to our proposed metric, but also with respect to encoding length; and it is *intuitive*, conforming to well-known educational principles. Experiments on real and synthetic datasets demonstrate the effectiveness of GROUPNTEACH.

1 Introduction

If you were given Figure 1 (c) and (d) to memorize, which would you find easier to memorize? If you were a zoology teacher, how would you come up with a lesson plan to teach the facts in Table 1, containing animals and their properties?

In our formulation, our facts consist of simple *(object, property)* pairs (Table 1a). Informally, our problem can be stated as follows:

Informal Problem 1 (Transmission / Teaching Rate Problem) *Given a large, sparse binary matrix whose rows represent objects, columns represent properties, in which ones represent facts, how do we measure how good a particular encoding of the matrix is for student learning, and how do we optimize this metric?*

Table 1 illustrates our intuition behind the solution: most people would agree that randomly stating facts ('salmons have fins') would be painful for the student. A good teacher would group animals and properties, as in Table 1b, and use analogies and comparison, such as 'tigers are like lions, but have stripes.'

Our contributions are as follows:

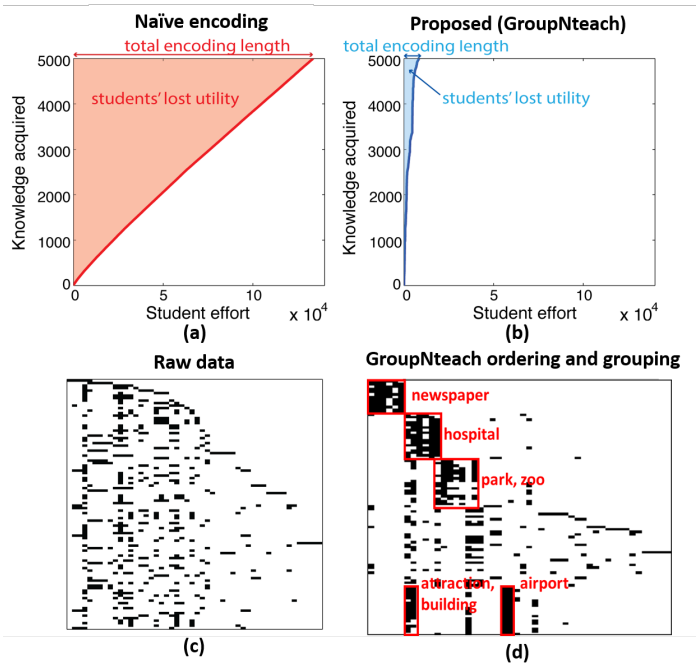


Fig. 1: **GROUPNTEACH agrees with intuition**. GROUPNTEACH (b) encodes facts with much lower total encoding length and students' lost utility than the (a) naive encoding, which encodes the nonzero entries of the matrix one by one. As a side effect, given randomly ordered data (c), GROUPNTEACH finds a (d) reordering and groupings of the facts along with labels which are intuitive .

- **Problem Formulation**: we formulate the *Transmission Rate Problem* formally in the context of matrix compression, and define how in the context of an educational setting, our goals differ from those of the standard matrix compression problem.
- **Optimization Goal**: we formulate a new metric which prioritizes consistent learning, rather than purely maximum compression, which maximizes student utility. This enables us to design *parameter-free* algorithm that picks optimal parameters for the defined criterion.
- **Algorithm**: we propose GROUPNTEACH, an algorithm for encoding and ordering a set of facts for student learning. GROUPNTEACH has the following properties:
  1. Scalable: it scales near-linearly in the data size, allowing it to scale to datasets with millions of facts.
  2. Effective: it encodes real datasets more efficiently than standard approaches for encoding sparse data, under both compression length and our metric.
  3. Intuitive: it follows educational principles such as grouping related concepts.

Reproducibility: All datasets and code we use are publicly available `http://www.cs.cmu.edu/~hyunahs/tol`.

A snapshot of our results is shown in Figure 1. The proposed GROUPNTEACH outperforms the baseline (Dot-by-Dot) in terms of the student's lost utility, and total encoding length. Also, GROUPNTEACH automatically reorders and groups facts of the matrix as a by-product. A brief flowchart of GROUPNTEACH is shown in Figure 2, which will be discussed in more detail in later sections.

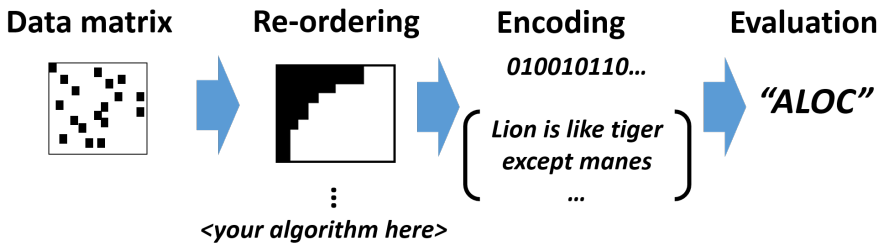


Fig. 2: **Flowchart of GROUPNTEACH.** GROUPNTEACH reorders the input data, encodes the information, and finally evaluates each plugin in GROUPNTEACH and returns the winning result. GROUPNTEACH tries several plugins to reorder the data, but any alternate reordering method can be plugged in 'your algorithm here.'

	ears	fins	stripes	lungs	gills	carnivore	...
salmon		•			•		
tiger	•		•	•		•	
jaguar	•			•		•	
tuna		•			•		
lion	•			•		•	
⋮							

(a) Raw data matrix.

	ears	lungs	carnivore	stripes	fins	gills	...
lion	•	•	•				
tiger	•	•	•	•			
jaguar	•	•	•				
tuna					•	•	
salmon					•	•	
⋮							

(b) Re-ordered data matrix.

Table 1: **Good grouping leads to good teaching:** matrix consisting of facts, which are (animal, property) pairs. Note that version (b) is much easier to describe and remember.

2 Background and Related Work

Support from Learning Theory. Improving student learning has been a great interest in various domains including psychology, education [10, 7], as well as data mining [11, 14]. In our study, we find that algorithms that do well under our metrics indeed agree

with educational instructional principles [10], which are explained in Table 2. In the last column, we show the keyword of GROUPNTEACH (Table 4) that reflects the principles.

	details	example	GROUPNTEACH
P1 **(Linking)**	link multiple objects together	'Tigers, lions, and jaguars all have teeth'	'ranges'
P2 **(Pre-conceptions)**	refer and relate to prior knowledge	'Tigers have teeth' 'So do lions'	'like'
P3 **(Comparison)**	compare and contrast multiple instances	'Tigers are like lions except that they have stripes'	'except'

Table 2: Instructional principles found in [10], examples of how GROUPNTEACH conforms to these principles, and the corresponding language in GROUPNTEACH.

Matrix Compression. There are several methods for efficiently compressing a matrix [16]. Our goal is different, particularly due to the educational setting: (a) we want metrics that prioritize consistent learning rather than pure encoding length; (b) we want to encode new information based on a learner's existing knowledge.

Bipartite Clustering and Binary Matrix Reordering. Various bipartite clustering and matrix reordering algorithms [5, 17] can be plugged in to our reordering stage as shown in Figure 2. In Table 3, we compare GROUPNTEACH to related algorithms in terms of the functions they perform such as ordering, compression, and grouping, as well as their relation to educational and human computer interaction (HCI) principles.

Minimum Spanning Trees. In one of our proposed plugins, GROUPNTEACH-Tree, we adopt EUCLIDEAN-MST [13].

3 Proposed Metric

How can we assign a numerical score for comparing different ways of teaching a collection of facts to students?

Definition 1 *Performance-for-Price curve* $p(n)$ Given an encoding algorithm, define $p(n)$ as the number of nonzero entries of the matrix that are decodable based on the first n bits output by the encoding algorithm (e.g. see Figure 1; top).

Definition 2 *Area Left of Curve* (*ALOC*) The *ALOC* metric is the area left of the curve $p(n)$. Lower *ALOC* is better.

Utility interpretation. Assuming the students gain utility at each time step according to how much they know, the total utility gained by students is the area under the curve $p(n)$. Then *ALOC* corresponds to the utility lost by a student over time compared to having known all the information in advance.

ALOC uses the number of bits transmitted as the units along the x-axis when plotting $p(n)$ since this represents the amount of attention that students need to understand the lesson content. Hence transmitting the message efficiently lowers the amount of effort students need to understand the material.

	Functions	Matrix Factorization	Matrix Compression[4, 2]	FCA[6]	GROUPNTEACH
	Ordering				✓
	Compression	✓	✓		✓
	Grouping	✓	✓	✓	✓
Teachability	P1: *11th* Linking	✓	✓	✓	✓
	P2: *13th* Pre-conceptions				✓
	P3: *17th* Comparison				✓
HCI	"Shneiderman mantra"		✓	✓	✓

Table 3: **GROUPNTEACH qualitatively outperforms competitors.** Block-based methods in general link multiple objects (**P1**), but only GROUPNTEACH teaches new concepts based on students' existing concepts (**P2**) and communicates based on similarity and difference. (**P3**) The Shneiderman mantra refers to communicating a high-level summary first, followed by finer details.

4 Proposed Method: GROUPNTEACH

In this section, we describe GROUPNTEACH; it finds efficient and interpretable encodings for a collection of facts, and as a by-product, it sequences the facts so as to find groupings of related facts, and a good teaching order for the groups.

All-engulfing approach. Since real-world datasets differ greatly in their underlying patterns, we expect different algorithms to perform well on different datasets. Hence, we propose four plugins, each designed to perform well on a particular type of dataset. Our *all-engulfing* approach tries each plugin to encode a dataset, and chooses the encoding with lowest *ALOC*. The complete algorithm is given in Algorithm 1.

Algorithm 1: GROUPNTEACH : encoding method that also returns grouped and ordered data

Data: Data matrix M (a binary matrix of objects by properties)
Result: Encoding for M and re-ordering and groupings of rows and columns of M
Plugins = {*Block, Tree, Chain, Fishbone*} *(any other heuristics can be added)*;
$i^* = \arg\min_i ALOC(\text{Plugins}[i](M))$ *(find best performing component)*;
Output binary representation of i^* *(index of the winning method)*;
Output *encoding* of M using method Plugins[i^*];
Reorder rows and columns of M using the orderings induced by GROUPNTEACH-Chain (see Section 6);
Group rows and columns according to Algorithm 5;

Table 4 summarizes the encoding structure and keywords each plugin uses to encode information.

Method	Encoding structure	Keywords	Encoding
Block	*(r-id, c-id)* + *(row-length)* + *(column-length)* ***except*** + *(r-id, c-id)* + ***except*** + *(r-id, c-id)* + ... + ***end-statement***	**except** **end-statement**	1 0
Tree / Chain	*(r-id)* + *(comparison r-id)* + **except** + *(c-id)* + **except** + *(c-id)* + ... + **end-statement**		
Fishbone	*(length of block)* + **except** + *(c-id)* + **except** + *(c-id)* + ... + **end-statement**		

Table 4: Encoding structure used for each method. *r-id* and *c-id* refer to *row index* and *column index*, respectively. The **except** keyword communicates exceptions, e.g. for the Block plugin, exceptions are the zeroes within the current block. For Tree and Chain, exceptions are differences between the current and compared rows. **end-statement** terminates the list of exceptions.

5 GROUPNTEACH-plugins

In this section, we give detailed explanations on each of the plugins.

5.1 GROUPNTEACH-Block

GROUPNTEACH-Block, explained in Algorithm 2, is designed to encode block-structured data highly efficiently. It does this by clustering the rows and columns to produce dense blocks, then re-orders the block regions, and encodes the block information (starting point, row and column length of the block), along with the missing elements in the block, and the additional elements outside the defined blocks.

5.2 GROUPNTEACH-Tree

GROUPNTEACH-Tree encodes a row by describing its differences from a similar row. For example, to describe *tigers*, we say that *tigers* are like *lions* except that they have stripes. Since real-world datasets typically have many similar items, most rows end up having very short encodings.

The first row encoded is the row with the most ones, encoded directly as a binary string. All subsequent rows are encoded using statements like 'row i is like row j except in positions $k, l, \ldots$'. This means row i can be obtained by starting with row j and flipping the bits in positions $k, l, \ldots$. To construct this encoding, GROUPNTEACH-Tree first constructs a distance function $d(i, j)$, equal to the number of differences between row i and row j. Then finding an encoding is equivalent to constructing a spanning tree:

Algorithm 2: GROUPNTEACH-BLOCK.

Data: Data matrix M
Result: Encoding of M
Step 1: Partition rows and columns into groups;
  Input: M, k;
  Cluster rows and columns of matrix M into k groups using NMF;
  Output: row and column indices for each cluster regions;
Step 2: Get the best permuted M_{pm^*};
  Input: cluster information;
  Compute density of each block;
  Output: M_{pm^*}, The best permuted matrix with highest density in the diagonal blocks;
Step 3: Encode M_{pm^*};
  Using ENCODE-BLOCK(M_{pm^*}), encode the top-left corner of the block, row and column lengths of the block.;
  Output: Encoding of M;

for example, if we encoded row i based on similarity to row j, then (i, j) is an edge of weight $d(i, j)$ in the corresponding tree. It is a tree because each row has exactly one ancestor (the row used to encode it), except the root. Then, we can minimize the number of differences we need to encode by minimizing the weight of the spanning tree. We could do this by constructing a distance matrix $d(i, j)$ between the rows, then finding the MST using e.g. Kruskal's algorithm. Since Kruskal's algorithm would require quadratic time, however, we instead use the Euclidean MST algorithm which takes $O(n \log n)$ time, as given in Algorithm 3.

Algorithm 3: GROUPNTEACH-TREE: fast approximate minimum spanning tree-based encoding method

Data: Data matrix M
Result: Row-wise encoding for M
Let $(M_i)_{i=1}^n$ be the rows of M;
Generate random vectors $f_1, \ldots, f_p \in \mathbb{R}^n$;
for *i=1,...,m* **do**
  $x_i = (M_i \cdot f_1, \ldots, M_i \cdot f_p)$ (*Construct feature vectors*)
$T =$ EUCLIDEAN-MST$(x_1, \ldots, x_m)$;
Choose row index r with largest row sum;
Output M_r;
Let O be a BFS traversal of T with root r;
for *each edge (i, j) in O* **do**
  Let $D(i, j)$ be the set of column indices at which M_i differs from M_j;
  Output $\langle i, j, D(i, j)\rangle$ (*output that row j is like row i except in columns $D(i, j)$*)

5.3 GROUPNTEACH-Chain

GROUPNTEACH-Chain is similar to GROUPNTEACH-Tree: we also encode each row based on a similar row. However, unlike the tree pattern of GROUPNTEACH-Tree, here each row is encoded based on comparison to the last encoded row. For example, we may encode *lions* based on *tigers*, then *jaguars* based on *lions*, and so on, forming a chain. This allows the encoding of *lions* to not encode its parent *tigers*, since its parent *tigers* can be deduced from being the last animal encoded. This encodes each row more cheaply and is more efficient for sparse data.

To find a good ordering of the rows, we first use the same random projections method as GROUPNTEACH-Tree to obtain feature representations $x_1, \ldots, x_m$ of the rows. We then use the Euclidean distance between x_i and x_j as a proxy for the number of differences between rows i and j. Starting from the row with the most ones, we repeatedly find the next row to encode as follows: randomly sample a fixed k of the remaining unused rows; choose the closest of these k rows as the next row to encode; then continue this until all rows are encoded.

5.4 GROUPNTEACH-Fishbone

GROUPNTEACH-Fishbone aims to efficiently encode data with uneven number of ones in each row, such as power-law degree distributions which are common in online communities [2]. GROUPNTEACH-Fishbone rearranges as many ones as possible to the top-left of the matrix, then takes advantage of the density of that region to encode the data efficiently. To do this, GROUPNTEACH-Fishbone first reorders the rows and columns in descending order of their row or column sum. Then, it encodes the top row by encoding a number k followed by a list of exceptions $p, q, r, \ldots$. This indicates that except at positions $p, q, r, \ldots$, the row contains k ones, then $n - k$ zeroes. k is chosen by trying all possible k and using the shortest encoding. For efficiency, we terminate the search for k early if the current encoding is some fixed constant C bits worse than the best found encoding. Having encoded the top row, we then encode the first column of the remaining matrix in the same way, and so on, as shown in Algorithm 4.

Algorithm 4: GROUPNTEACH-FISHBONE.

Data: Data matrix M
Requires: ENCODE-ROW, a function that encodes a vector of length n by comparing it to k ones followed by $n - k$ zeroes, and listing all exceptions to this pattern;
Result: Encoding of M
Reorder rows and columns of M in descending order of row and column sums;
while M *is non-empty* **do**
  Let $(M_i)_{i=1}^n$ be the rows of M;
  $k^* = \arg\min_k \text{LENGTH}(\text{ENCODE-ROW}(M_1, k)))$;
  Output ENCODE-ROW(M_1, k^*);
  Remove M_1 from M and transpose M;

5.5 Extensibility

GROUPNTEACH can be broken down into two parts: *reorganization (reordering of rows and columns)*, and *encoding of the reorganized matrix*. GROUPNTEACH can be easily extended by plugging in any matrix reorganization method, such as Cross Association [4] or METIS [8].

6 GROUPNTEACH-post processing: ordering, grouping and curriculum development

As a by-product, GROUPNTEACH produces an intuitive ordering and grouping of the objects in the dataset, as was shown in Figure 1.

The process of grouping is described in Algorithm 5.

Algorithm 5: GROUPINGCODE: groupings of the related facts on the reordered matrix

Data: Reordered data matrix $\tilde{M}$ recovered as output of GROUPNTEACH, threshold C
Result: A set of groups $\mathcal{G}$ for the reordered data matrix $\tilde{M}$
Group data in $\tilde{M}$ into rectangles by combining nearby entries if they form rectangular blocks;
while *not converged* **do**
  for $i = 1, \ldots, |\mathcal{G}|$ **do**
    for $j = 1, \ldots, |\mathcal{G}|$ **do**
      Let R be the smallest bounding box covering R_i and R_j;
      if $\frac{\textit{number of 1s in} R}{\textit{area of} R} \geq C$ **then**
        remove R_i and R_j from $\mathcal{G}$, and add R to $\mathcal{G}$;
  Output $\mathcal{G}$;

7 Experiments

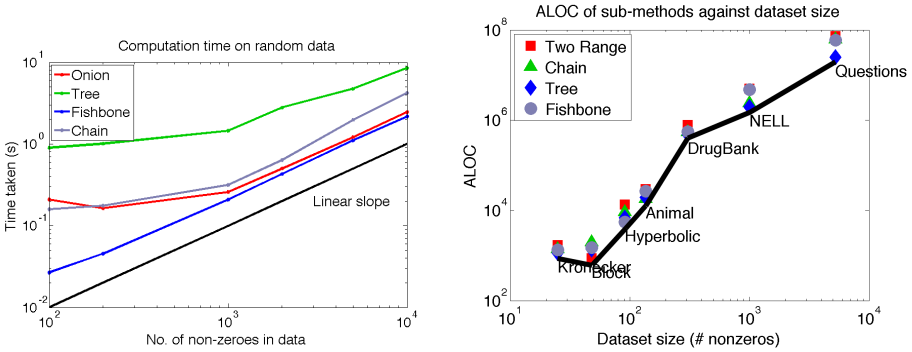
In this section we demonstrate the efficiency and effectiveness of GROUPNTEACH using real and synthetic datasets. We implemented GROUPNTEACH in MATLAB; all experiments were carried out on a 2.4 GHz Intel Core i5 Macbook Pro, 16 GB RAM, running OS X 10.9.5. Our code and all our datasets are publicly available at `http://www.cs.cmu.edu/~hyunahs/tol`. We used 100 features ($p = 100$) for GROUPNTEACH-Chain and GROUPNTEACH-Tree, and threshold $C = 50$ for GROUPNTEACH-Fishbone. The real datasets used are shown in Table 5. The synthetic datasets used are: 1.KRONECKER: a 256×256 Kronecker graph [12], 2.BLOCKS: two 50×70 blocks of ones in a matrix, 3.HYPERBOLIC: a 20×20 matrix containing 3 overlapping communities of sizes 20, 8 and 4, each resembling a scale-free network.

	size	number of nonzeros	content
ANIMAL [3]	34 by 13	136	animal-property
NELL [1]	212 985 by 217	1.1 million	object-category
DRUG-BANK [9]	1581 by 16 883	109 339	drug-property
QUESTIONS [15]	60 by 218	5252	question-answer

Table 5: Real datasets used.

We conducted several experiments to answer the following questions: **Q1. Scalability, Q2. Effectiveness, Q3. Discoveries.**

Q1. Scalability: Figure 3 (a) shows the linear or near-linear performance of our algorithms. The algorithms are run on random matrices of varying number of rows, fixed to 1000 columns and an average of 10 ones per row.



(a) (b)

Fig. 3: (a) **GROUPNTEACH scales linearly.** GROUPNTEACH scales linearly with the input size (The line $y = cx$ for $c = \frac{1}{10000}$ is added for comparison). (b) **GROUPNTEACH needs all plugins**: different plugins win on different datasets (lower is better). The black lower bound is the final result by GROUPNTEACH.

Q2. Effectiveness: We demonstrate that the multiple plugins of GROUPNTEACH allow it to do well on diverse types of data. Figure 3 (b) shows that the various plugins of GROUPNTEACH do well on different types on data: GROUPNTEACH-Block for block-wise, GROUPNTEACH-Fishbone HYPERBOLIC, GROUPNTEACH-Chain and GROUPNTEACH-Tree datasets with similar rows or columns (which is the case for the real datasets). No one method dominates the others.

Q3. Discoveries: As a by-product, GROUPNTEACH automatically reorders and groups the data. We analyze this property using the DRUG-BANK dataset; GROUPNTEACH finds a teaching order with several desirable characteristics, as shown in Figure 4.

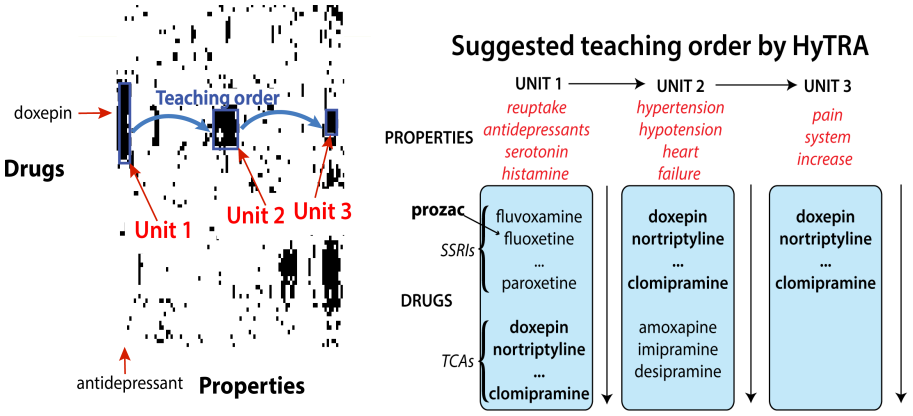


Fig. 4: **GROUPNTEACH leads to curriculum discovery. Left:** re-ordered and grouped DRUG-BANK drug-property data. GROUPNTEACH constructs a teaching order. **Right:** curriculum constructed by GROUPNTEACH. The units ('blobs') obtained are intuitive: e.g. drugs in the TCA family are known to have several groups of effects, which our algorithm groups as follows. Unit 1: their antidepressant properties; Unit 2: their cardiac side-effects; and, some of them, Unit 3: treating chronic pain.

8 Conclusion

In this paper, we considered the problem of teaching a collection of facts while minimizing student effort. Our contributions are as follows:

- **Problem Formulation**: we define the problem of transmitting a matrix of objects and properties adhering to principles from the theory of (human) learning.
- **Optimization Goal**: We define an appropriate optimization goal; minimizing *ALOC* (maximizing student utility).
- **Algorithm**: We propose GROUPNTEACH, an *all-engulfing* method that encodes the data while reordering and grouping the data. We evaluate GROUPNTEACH on synthetic and real datasets, showing that it encodes data more efficiently than a naive encoding approach, measured using both *ALOC* and total encoding length.
- **Ordering of Groups**: When applying GROUPNTEACH on real datasets, we find that the orderings and groupings it produces are meaningful.

Acknowledgments. This material is based upon work supported by the National Science Foundation under Grant No. IIS-1247489

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053.

This work is also partially supported by an IBM Faculty Award and a Google Focused Research Award. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the

views of the National Science Foundation, or other funding parties. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

1. Read the web. `http://rtw.ml.cmu.edu/rtw/`.
2. M. Araujo, S. Günnemann, G. Mateos, and C. Faloutsos. Beyond blocks: Hyperbolic community detection. In *ECML-PKDD*, pages 50–65, 2014.
3. R. Bro, E. E. Papalexakis, E. Acar, and N. D. Sidiropoulos. Coclustering - a useful tool for chemometrics. *Journal of Chemometrics*, 26(6):256–263, 2012.
4. D. Chakrabarti, S. Papadimitriou, D. S. Modha, and C. Faloutsos. Fully automatic cross-associations. In *ACM KDD*, pages 79–88, 2004.
5. I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *7th ACM SIGKDD*, pages 269–274. ACM, 2001.
6. B. Ganter, G. Stumme, and R. Wille. *Formal concept analysis: foundations and applications*, volume 3626. Springer Science & Business Media, 2005.
7. F. Gobet, P. C. Lane, S. Croker, P. C. Cheng, G. Jones, I. Oliver, and J. M. Pine. Chunking mechanisms in human learning. *Trends in cognitive sciences*, 5(6):236–243, 2001.
8. G. Karypis and V. Kumar. METIS-unstructured graph partitioning and sparse matrix ordering system, version 2.0. 1995.
9. C. Knox, V. Law, T. Jewison, P. Liu, S. Ly, A. Frolkis, A. Pon, K. Banco, C. Mak, V. Neveu, et al. Drugbank 3.0: a comprehensive resource for omics research on drugs. *Nucleic Acids Research*, 39(suppl 1):D1035–D1041, 2011.
10. K. R. Koedinger, J. L. Booth, and D. Klahr. Instructional complexity and the science to constrain it. *Science*, 342(6161):935–937, 2013.
11. K. R. Koedinger, E. Brunskill, R. S. J. de Baker, E. A. McLaughlin, and J. C. Stamper. New potentials for data-driven intelligent tutoring system development and optimization. *AI Magazine*, 34(3):27–41, 2013.
12. J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani. Kronecker graphs: An approach to modeling networks. *JMLR*, 11:985–1042, 2010.
13. W. B. March, P. Ram, and A. G. Gray. Fast Euclidean minimum spanning tree: algorithm, analysis, and applications. In *ACM KDD*, pages 603–612, 2010.
14. N. Matsuda, W. W. Cohen, and K. R. Koedinger. Teaching the teacher: Tutoring SimStudent leads to more effective cognitive tutor authoring. *IJAIED*, pages 1–34, 2014.
15. B. Murphy, P. Talukdar, and T. Mitchell. Selecting corpus-semantic models for neurolinguistic decoding. In *ACL *SEM*, pages 114–123. Association for Computational Linguistics, 2012.
16. R. E. Tarjan and A. C.-C. Yao. Storing a sparse table. *CACM*, 22(11):606–611, 1979.
17. H. Zha, X. He, C. Ding, H. Simon, and M. Gu. Bipartite graph partitioning and data clustering. In *10th CIKM*, pages 25–32. ACM, 2001.