Towards mining and learning with networked examples

Yuyi Wang and Jan Ramon
Department of Computer Science K.U.Leuven
Celestijnenlaan 200 A, 3001 Heverlee, Belgium
firstname.lastname@cs.kuleuven.be

ABSTRACT

An important challenge related to the prediction of relationships between groups of vertices or the properties of such relationships (e.g., link prediction) is that links are not independent (they share vertices) and hence the common assumption that examples are drawn identically and independently does not hold. A related problem occurs in frequent pattern mining, where it is non-trivial to define an appealing frequency measure for measuring the support of a pattern in a network. In this paper, we discuss a line of research aiming at solving these two problems in an elegant way by defining a measure which both describes the generalization power of a sample and is an anti-monotonic, normalized graph support measure. We review earlier work, discuss recent results, and suggest directions for future work.

Categories and Subject Descriptors

I.2 [**ARTIFICIAL INTELLIGENCE**]: Learning; G.2.2 [**DISCRETE MATHEMATICS**]: Graph Theory—*Hypergraphs, Network problems*

General Terms

Theory

1. INTRODUCTION

Recently there is an increasing amount of networked data, i.e., data represented with networks of connected entities. One of the challenges of machine learning in the context of such networked data is that the common assumption that examples are distributed identically and independently does not hold. In fact, in many applications making such assumption does not give good results. A better understanding of the statistical aspects of learning in a networked setting would be beneficial to guide further research in that setting.

In our research, we consider a weaker form of independence assumption, and derive for it a learning bound and a pattern frequency measure. This paper explains the earlier work, our ongoing research and a roadmap for further work. The contributions of this paper are threefold. First, we introduce a concept of networked examples, i.e., examples for a machine learning task sharing parts of their features. Second, under a weaker independence assumption, we show a learning bound which is better than in earlier work. Third, we point out a relationship between graph support measures in the context of pattern mining and measures for the statistical power of a sample.

The paper is structured as follows. First, after some preliminaries in Section 2, in Section 3 we define more precisely our setting for networked examples. Then, Section 4 reviews work on support measures and relates it to our setting. In Section 5 we will formulate the problem of learning from networked examples and state the weaker independence assumption we make. In Section 7 we discuss MBNL, a Measure for Bernstein-bound-optimal Learning from Networked examples, including the learning bound (the detailed reasoning can be found in [14]) and pattern frequency measure it allows for. In Section 8 we review related work on learning in a network. In Section 9 we conclude with a discussion and pointers for future work.

Eleventh Workshop on Mining and Learning with Graphs. Chicago, Illinois, USA

2. PRELIMINARIES

In this section, we briefly review some basic definitions on graphs. A labeled graph is a triple (V, E, λ) where V is a set of vertices, E is a set of edges and λ is a labeling function which assigns every vertex (and/or every edge) a label. We denote the set of vertices of a graph G with $V(G)$, the set of edges with $E(G)$ and the labeling function λ_G. We denote the set of all graphs with $\mathcal{G}$. A hypergraph is a pair (V, E) where V is a set of vertices and E is a set of hyperedges. Two graphs P and D are isomorphic iff there exists a bijective mapping $\pi : V(P) \mapsto V(D)$ such that 1) for all $v \in V(P)$, $\lambda_P(v) = \lambda_D(\pi(v))$, 2) any two vertices u and v of P are adjacent in P if and only if $\pi(u)$ and $\pi(v)$ are adjacent in D. The mapping π is called the isomorphism mapping. A graph P is a subgraph of a graph D iff $V(P) \subseteq V(D)$ and $E(P) \subseteq E(D)$. A graph P is subgraph isomorphic to a graph D iff P is isomorphic to a subgraph of D. An embedding of P in D is an isomorphism mapping between P and a subgraph of D. The set of all embeddings of P in D is denoted $Emb(D, P)$. We will often call P the (subgraph) pattern and D the database graph or network.

3. A SETTING FOR NETWORKED EXAMPLES

In this paper, we consider a setting where examples share part of their features. The i.i.d. assumption does not always hold in this setting. For instance, suppose that we are interested in predicting whether a given person likes a given movie. We could ask a set of persons to grade five of the movies they have seen in the past. Then, we want to predict for a new visitor (drawn from the same distribution as our training persons) whether he will like a newly introduced movie (having features drawn from the same distribution as the movies in the past). Our training examples (each containing a person ID, a movie ID and a grade) are not all independent since each person graded several movies and all movies were graded by several persons. Still, we would like to get a generalization guarantee.

More formally, given a graph D and a subgraph pattern P, we say that a P-example e is an embedding under subgraph isomorphism of P in G, i.e., $e \in Emb(D, P)$. For instance, in our movie grading problem, we could represent our data as follows: we represent every movie with a vertex labeled `movie`, every person wtih a vertex labeled `person` and every grade with a vertex labeled `grade`. We connect every grade to the person vertex who gave the grade and the movie vertex which was graded. We also connect every pair of persons who are friends. For our pattern graph P, we would have vertices $V(P) = \{1, 2, 3\}$, edges $E(P) = \{\{1, 2\}, \{2, 3\}\}$ and labeling function $\lambda_P = \{(1, \texttt{person}), (2, \texttt{grade}), (3, \texttt{movie})\}$.

$Emb(D, P)$ induces a hypergraph on the vertices $V(G)$ of G. In particular, we denote with $Ex(D, P)$ the hypergraph for which $V(Ex(D, P)) = V(D)$ and $E(Ex(D, P)) = \{e \in V(D)^{V(P)} \mid \exists \pi \in Emb(D, P), \forall v \in V(P) : e_v = \pi(v)\}$, where hyperedges $e \in E(Ex(D, P))$ are indexed with vertices of P, e.g., in our movie grading database if $e \in E(Ex(D, P))$ then e_1 refers to a person vertex in D. For ease of notation and without loss of generality we assume that the pattern vertices are the integers from 1 to $|V(P)|$, as is the case in the above example. A feature map on a graph D is a mapping ϕ assigning to each vertex and edge of D a feature. Features may be single values or may be structured (e.g. consist of vectors). While labels should be preserved by subgraph isomorphism, features should not. We also use the notation ϕ as a function on examples: for any example e, we call $\phi(e) = [\phi(e^{(1)}), \phi(e^{(2)}), \ldots, \phi(e^{(k)})]$ the feature vector of e. For instance, in our movie rating example, ϕ could assign to movies $m \in V^{(1)}$ pairs $(genre, length)$, to persons $p \in V^{(2)}$ a triple $(gender, age, nationality)$ and to a rating $r \in V^{(3)}$ a pair $(watching\_time, movie\_version)$. ϕ would therefore assign to an example a triple with in total 8 values.

4. COUNTING FREQUENCY

In our movie grading database, we considered already triples of movies, grades and persons. However, there may be many other patterns of vertices satisfying specific relationships and of interest to consider. For instance, considering pairs of friends we may discover interesting correlations between their properties or the types of movies they watch.

The task of pattern mining is concerned with collecting all such patterns of interest. A classic form of pattern mining is frequent (subgraph) pattern mining. There, patterns which occur often in a database (i.e., patterns which are frequent) are considered as interesting. However, in order to perform frequent pattern mining, one needs a frequency measure (also called support measure). In particular, given a pattern $P \in \mathcal{G}$ and a network $D \in \mathcal{G}$, a *support measure* is a function $sup : \mathcal{G} \times \mathcal{G} \mapsto \mathbb{R}^+$ which measures how frequently the pattern P occurs in the network D.

4.1 Desirable properties

For simple settings such as itemset mining, a simple support measure is to count the number of transactions matched by the pattern. However, in the context of subgraph patterns in a single network, the issue is less straightforward as several articles have demonstrated [11, 1, 5, 2].

We say that a support measure is *anti-monotonic* if $Sup(D, P) \leq Sup(D, p)$ whenever p is a subgraph of P, i.e., $p \preceq P$. The anti-monotonicity of the support measure (or more generally interestingness measure) plays a very important role in the design of a subgraph pattern miner as it makes it possible to prune the search space [9]. For instance, just using the number of occurrences of a subgraph pattern as its support does not give an anti-monotonic support measure because the number of embeddings of a pattern in a fixed network may be exponential in the pattern size. In [1], the authors proposed an anti-monotonic support measure, which we call *min-image* : $minImage(D, P) = \min_{v \in V(P)} |\{\phi(v) \mid \phi \in Emb(D, P)\}|$. However, the *min-image* measure may overestimate the statistical evidence of a pattern [15]. We will discuss such statistical concerns in Section 7.

Calders et al. [2] pointed out that being anti-monotonicity is not sufficient to be a good support measure. For example, a support measure that just returns a constant is anti-monotonic, but not informative. [2] proposed using a situation in which occurrences of a subgraph pattern occur independently (i.e., they do not overlap according to some notion of overlap) as a reference. In particular, the notion of a normalized graph support measure was defined: a support measure is *normalized* if every subgraph pattern which only has non-overlapping occurrences in a network has a support in this network that equals the number of occurrences.

4.2 Overlap-graph based support measures

An important class of anti-monotonic normalized support measures relies on the notion of overlap. There are different types of overlap, e.g. two embeddings of a pattern have vertex-overlap if they share a vertex. There is a relationship between overlapping embeddings of a pattern and dependent observations. For instance, if we have a movie graded by two different people, the movie-related features of the corresponding examples would be equal (depend on each other). At the same time, the two embeddings would overlap (more specifically, would vertex-overlap [12]).

In order to study overlap, [12] introduced the notion of *overlap graph*. In an overlap graph G_D^P, the vertices represent embeddings of a given subgraph pattern P, and two vertices are adjacent if and only if the corresponding embeddings overlap in the network D (according to some notion of overlap, such as sharing a vertex or an edge). An overlap graph therefore indicates how often a subgraph pattern occurs in the network, and how independent these occurrences are. An overlap-graph based support measure (OGSM) takes an overlap graph of a subgraph pattern in a network as its input, and outputs the support of that subgraph pattern in that network. Vanetik et al. [11] proposed the *MIS* measure, i.e., the size of the maximum independent set of the overlap graph, $MIS(D, P) = \max_{I \subseteq V(G_D^P)} \{|I| \mid$

$\forall e \in E(G_D^P) : |e \cap I| \leq 1\}$. This is intuitively appealing since it measures how often we observe a subgraph pattern occurring independently. However, computing (or even approximating) the *MIS* of an overlap graph is untractable [4]. Calders et al. [2] showed that any normalized anti-monotonic support measure should be larger than the *MIS* measure and smaller than the MCP measure (the minimum clique partition, another NP-hard to approximate number). Moreover, they proposed the Lovász theta function (see, e.g., [8]), $\vartheta(D,P) = \vartheta(G_D^P)$, which is computable in time polynomial in the size of the overlap graph using semidefinite programming (SDP). Unfortunately, even the best known polynomial time algorithms [6] don't scale sufficiently well to compute $\vartheta(D,P)$ for large D.

4.3 An overlap-hypergraph based measure

[15] proposed a new anti-monotonic, normalized support measure `s` that is based on bounding the value of all occurrences of a subgraph pattern that share a particular part of the network. The `s` value can be computed efficiently as it is the solution to a linear program (LP). In particular, given a pattern P and a network D, we weight every embedding in $Emb(D,P)$ a nonnegative value. The `s` value takes the maximization of the sum of the weights under the constraints that the sums of the weights of the embeddings which share a certain vertex in D should be smaller than or equal to 1. The measure `s` is not a traditional OGSM, because its output does not merely depend on the overlap graph, but on a more fine-grained *overlap hypergraph* H_D^P not only representing that two embeddings overlap but also representing the element (e.g. the shared vertex) in which they overlap. One can think of the H_D^P considered in [15] as a dual of the graph $Ex(D,P)$ considered here: in $Ex(D,P)$, vertices are the pieces of information (referring to objects in the world and their features) and hyperedges group these objects into examples, while in H_D^P vertices are the examples (embeddings of P) and hyperedges group examples sharing a common piece of information (e.g. vertex of D).

5. LEARNING ASSUMPTIONS

We now return to the case we have one pattern P fixed and want to learn about a target value of the P-examples in D. For instance, in our movie-grading example we may want to predict whether a new person will like a new movie. Even though the results below apply more generally, in this paper we assume that $Ex(D,P)$ is a k-partite graph, i.e., there is a partition $\{V^{(i)}\}_{i=1}^{|V(P)|}$ of $V(D)$ such that each hyperedge $e \in E(Ex(D,P))$ intersects every set of the partition in exactly one vertex. This is the case in our movie grading example: the sets of movies, grades and persons are disjoint.

For convenience, we will use the following shorthand notations. We use V and E to refer to $V(Ex(D,P))$ and $E(Ex(D,P)) = \{e_l\}_{l=1}^m$ respectively. The number of partitions of V (and the arity of the hyperedges in E is denoted with k. The number of examples (hyperedges in E) is denoted by m, and the cardinality of the partition $V^{(i)}$ is denoted by n_i, i.e., $|V^{(i)}| = n_i$. The j-th vertex of the partition $V^{(i)}$ is denoted by $v^{(i,j)}$ where $1 \leq j \leq n_i$. We denote the i-th component of an edge e as $e^{(i)}$, which is a vertex in $V^{(i)}$. Hence, two edges e_a and e_b *overlap* if and only if there exists $1 \leq i \leq k$ such that $e_a^{(i)} = e_b^{(i)}$. We also define $x^{(j,l)} = \phi(v^{(j,l)})$ and $x_l = \phi(e_l)$. With every example e_i there is also associated a target value y_i. We denote the labeled example with $\mathbf{z}_i = (\mathbf{x}_i, y_i)$ We call the space of possible values of $\mathbf{x}_l$ the input space and the space of all possible target values $\mathcal{Y}$ the output space. We make the following assumptions:

- The feature of every vertex in the partitions $V^{(i)}$ is drawn identically and independently from a fixed but unknown distribution ρ_i. We define $\rho_e(e) = \prod_{i=1}^k \rho_i(\phi(e^{(i)}))$.
- Especially, these features are independent from the edges in which they participate, i.e., $\rho_j(\phi(v^{(j,l)})) = \rho_j(\phi(x^{(j,l)}|E))$.
- Moreover, all hyperedges $e_l \in E$ (examples) get a target value y_l drawn identically and independently from $\rho_{y|\mathbf{x}}$. Even if the hyperedges share vertices, still their target value is sampled i.i.d. from some fixed but unknown distribution $\rho_{y|\mathbf{x}}$ based on their (possibly identical) feature vector.

So one can choose freely which vertices participate in which hyperedges, as long as this selection process is completely independent from the drawing of features for the vertices and the drawing of target values. Our analysis of the sample error $\mathcal{E}$ holds no matter what the distributions ρ_i and $\rho_{y|\mathbf{x}}$ are, as long as the above assumptions hold. In our movie rating example, it may or may not be realistic that these assumptions hold. In particular, if ratings are obtained from visitors of a cinema, then probably some visitors will already have a preference and will not choose movies randomly. On the other hand, if ratings are obtained during an experiment or movie contest where a number of participants or jury members are asked to watch a specific list of movies, one could randomize the movies to increase fairness, and in this way our assumptions would be satisfied. There are several other situations where our assumptions could be realistic. For instance, one could consider medical randomized trials where patients get a random (possibly placebo) treatment.

6. LEARNING STRATEGIES

In this section we discuss two plausible but suboptimal strategies for learning under the assumptions formulated above.

The EQW method In [7], the author shows an inequality which can be used to bound the error on averaging a function over networked samples. Let us now consider a learning strategy we call EQW (EQual Weight) and which learns from a set of networked examples in the same way as if they were i.i.d. (i.e., without weighting them as a function of the network structure). We can use the results in [7] to bound the sample error of EQW, of which we give one example below. In this paper, we consider the Empirical Risk Minimization principle which aims to find a minimizer of the empirical risk in a proper hypothesis space $\mathcal{H}$ to approximate the target function.

THEOREM 1. *Let $\mathcal{H}$ be a compact and convex set of functions from $\mathcal{X}$ to $\mathcal{Y}$. Assume $\mathcal{H}$ is M-bounded, i.e.,* $\sup_{f \in H} |f(x) - y| \leq M$ *holds almost everywhere on $\mathcal{Z}$ according to the probability distribution ρ on $\mathcal{Z}$. Let $h \in \mathcal{H}$ be the function minimizing the square loss over some training set (satisfying the assumptions stated earlier) containing m examples.*

Then for all $\epsilon > 0$,

$$\Pr\Big(\mathcal{E}(h_{eqw}) \geq \epsilon\Big) \leq \mathcal{N}\Big(\mathcal{H}, \frac{\epsilon}{12M}\Big) \exp\Big(-\frac{3m\epsilon}{1400\chi^*(G_D^P)M^4}\Big).$$

where the covering number $\mathcal{N}(\mathcal{H}, \tau)$ is the number of balls of radius τ needed to cover $\mathcal{H}$. $\chi^*(G_D^P)$ is the fractional chromatic number of G_D^P.
The result above shows a larger sample may result in a poorer bound since $\chi^*(G)$ can also become larger.

The IND method A straightforward idea to learn from a networked sample is to find a subset of training examples which corresponds to non-overlapping hyperedges. Due to our assumptions, such set will be an i.i.d. sample. We can then perform algorithms on this subset for learning. We call this method the IND method. To bound the sample error of this method, we can directly use the existing result. Under the same conditions as in Theorem 1, we get

$$\Pr\big(\mathcal{E}(h_{ind}) \geq \epsilon\big) \leq \mathcal{N}\Big(\mathcal{H}, \frac{\epsilon}{12M}\Big) \exp\Big(-\frac{n_{ind}\epsilon}{300M^4}\Big)$$

where n_{ind} is the size of the subset of independent examples.

For any overlap graph G_D^P, it holds that (see, e.g., [3]), $\frac{|V(G_D^P)|}{\chi^*(G_D^P)} \leq \alpha(G_D^P)$ where α is the independence number and an tight upper bound of n_{ind}. Therefore, the IND method will give us a better bound than the EQW method if we could find such a large subset of independent examples, but is computationally intractable because finding a maximum size set of independent examples is NP-hard.

7. BERNSTEIN-BOUND-OPTIMAL LEARNING FROM NETWORKED EXAMPLES

We propose a computationally efficient method based on the support measure discussed in 4.3. It allows for a better bound on the sample error than the IND method. Let w^* be an optimal assignment maximizing the objective function of the linear program used to compute $\mathsf{s}(H)$. One can show that the value $\mathsf{s}(H)$ is always greater than $\alpha(G_D^P)$ The following is our main result.

THEOREM 2. *Assume the same conditions of Theorem 1. Let h_{mbnl} minimize the weighted empirical risk $\frac{1}{\mathsf{s}}\sum_{i=1}^{m} w_i^*(h_{mbnl}(x_i) - y_i)^2$. Then,*

$$\Pr\big(\mathcal{E}(h_{mbnl}) \geq \epsilon\big) \leq \mathcal{N}\Big(\mathcal{H}, \frac{\epsilon}{12M}\Big) \exp\Big(-\frac{\mathsf{s}\epsilon}{300M^4}\Big).$$

8. RELATED WORK

[10] uses the concentration bound from [7] and studies a problem similar to ours and analyses the EQW method described above.

In [13], the authors consider a similar setting of networked examples. They use a dual representation where vertices correspond to examples and edges to dependencies, i.e., the overlap graph. While we assume a worst case over all possible dependencies, and allow to model explicitely causes of dependencies (represented with vertices which can be incident with more than two edges), this work assumes a bounded covariance between pairs of examples connected with an edge (excluding possible higher-order interactions). While we use our model to show learning guarantees, [13] shows corrections for the bias (induced by the dependencies between examples) on statistical hypothesis tests. It seems plausible that both models can be applied for both the learning guarantee and statistical testing tasks.

In the technical report [14] we discuss some more related work which is less related to the graph mining / statistical relational learning setting.

9. DISCUSSION AND FUTURE WORK

We can conclude that for overlapping (non-i.i.d.) examples it is possible to model and quantify the dependencies, and use this to bound the expected prediction error of a learner. We showed preliminary steps in that direction, using a weighting method and simple empirical risk minimization, and a lot of open questions remain. The ability of our idea to model dependencies is rather general, e.g. 'overlap' is a fairly general concept which can be defined according to the application domain and the needs of the analysis. For instance, maybe one could believe that friends influence each other heavily and therefore two triples (movie, grade, person) overlap both if they share the same person or if the two persons are friends. We also anticipate that it is possible to use a similar strategy to prove learning guarantees in related networked settings. Even though we believe that the measure discussed in Section 7 is optimal for Bernstein-inequality based proofs, we would like to investigate whether we can find measures and weightings optimal in a more general way. In this paper, we also pointed to a relationship with pattern mining, and we believe that exploring further this connection could be interesting. Finally, a long-term goal is to further weaken our independence assumptions.

Acknowledgments This work was supported by ERC Starting Grant 240186 "MiGraNT: Mining Graphs and Networks: a Theory-based approach"

10. REFERENCES

[1] B. Bringmann and S. Nijssen. What is frequent in a single graph? In *PAKDD'08*, pages 858–863, 2008.
[2] T. Calders, J. Ramon, and D. Van Dyck. All normalized anti-monotonic overlap graph measures are bounded. *Data Mining & Knowledge Discovery*, 23(3):503–548, 2011.
[3] R. Diestel. *Graph theory*. Springer-Verlag, 2010.
[4] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-Complete. In *FOCS'91*, pages 2–12, 1991.
[5] M. Fiedler and C. Borgelt. Support Computation for Mining Frequent Subgraphs in a Single Graph. In *Proceedings of MLG'07*, 2007.
[6] G. Iyengar, D. J. Phillips, and C. Stein. Approximating semidefinite packing programs. *SIAM Journal on Optimization*, 21(1):231–268, 2011.
[7] S. Janson. Large deviations for sums of partly dependent random variables. *Random Structures & Algorithms*, 24.3:234–248, 2004.
[8] D. E. Knuth. The sandwich theorem. *The Electronic Journal of Combinatorics*, 1:1–48, 1994.
[9] M. Kuramochi and G. Karypis. Finding frequent subgraph patterns in a large sparse graph. *Data Mining & Knowledge Discovery*, 11(3):243–271, 2005.
[10] N. Usunier, M.-r. Amini, and P. Gallinari. Generalization error bounds for classifiers trained with

interdependent data. In *Advances in Neural Information Processing Systems 18 (NIPS 2005)*, pages 1369–1376. MIT Press, 2006.

[11] N. Vanetik, E. Gudes, and S. E. Shimony. Computing frequent graph subgraph patterns from semistructured data. In *Proc. of the IEEE International Conference on Data Mining (ICDM'02)*, pages 458–465, 2002.

[12] N. Vanetik, S. E. Shimony, and E. Gudes. Support measures for graph data. *Data Mining and Knowledge Discovery*, 13(2):243–260, 2006.

[13] T. Wang, J. Neville, B. Gallagher, and T. Eliassi-Rad. Correcting bias in statistical tests for network classifier evaluation. In *Proc. of ECML/PKDD'11*, 2011.

[14] Y. Wang and J. Ramon. Learning from networked examples in a k-partite graph. arXiv:1306.0393.

[15] Y. Wang, J. Ramon, and T. Fannes. An efficiently computable subgraph pattern support measure: counting independent observations. *Data Mining and Knowledge Discovery*, pages 1–34, 2013.