The Cost of Learning Directed Cuts

Thomas Gärtner[1] and Gemma C. Garriga[2]

[1] Fraunhofer IAIS, Schloß Birlinghoven, 53754 Sankt Augustin, Germany
[2] HIIT Basic Research Unit, Helsinki University of Technology, Finland
thomas.gaertner@iais.fraunhofer.de, gemma.garriga@hut.fi

Abstract. In this paper we investigate the problem of classifying vertices of a directed graph according to an unknown directed cut. We first consider the usual setting in which the directed cut is fixed. However, even in this setting learning is not possible without in the worst case needing the labels for the whole vertex set. By considering the size of the minimum path cover as a fixed parameter, we derive positive learnability results with tight performance guarantees for active, online, as well as PAC learning. The advantage of this parameter over possible alternatives is that it allows for an a priori estimation of the total cost of labelling all vertices. The main result of this paper is the analysis of learning directed cuts that depend on a hidden and changing context.

1 Introduction

Classifying vertices in directed graphs is an important machine learning setting with many applications. In this paper we consider learning problems on directed graphs with three characteristic properties: (*i*) The target concept defines a directed cut; (*ii*) the total cost of finding the cut has to be bounded before any labels are observed to asses whether it will exceed some given budget; and (*iii*) the target concept may change due to a hidden context.

For one example consider the problem of finding the source of contamination in waste water systems where the pipes are the edges of the digraph and the direction is given by the direction of the water flow. Often uncontaminated waste water can be used to fertilise fields and it is important to find the cause of contamination as quickly as possible. As each test costs time and money, we aim at a strategy that needs the least number of tests. The pipes connecting uncontaminated water with contaminated water form a directed cut.

For another example consider classifying intermediate products in some process, e.g., for manufacturing cars, as faulty or correct. The process can be represented by a directed graph and the concept defines a directed cut as typically faults that appear in an intermediate product will also be present in later stages of the product. The directed cut we are interested in consists of all edges connecting correct to faulty intermediate products. Furthermore, the concept may depend on a hidden variable as some pre-assembled parts may vary and the fault may occur only for some charges and not for others. In order to be able to trade off between the cost of having a faulty product and the costs for finding the cause of the fault, tight performance guarantees are needed.

J.N. Kok et al. (Eds.): ECML 2007, LNAI 4701, pp. 152–163, 2007.

Performance guarantees proposed in machine learning literature can be distinguished into concept-dependent and concept-independent. *Concept-dependent guarantees* state that the performance of the learning algorithm depends on the *unknown* target concept's complexity. *Concept-independent guarantees* state that the performance of the learning algorithm depends on the instance space's complexity, or in a transductive setting on the given training and test sets. For real-world applications, one often faces the question whether the costs of labelling a whole dataset exceeds a given budget or not. In this case, concept-independent, transductive, bounds are to be preferred over concept-dependent guarantees.

Our first result is that for learning directed cuts we can achieve tight, concept-independent guarantees. Based on a fixed size of the minimum path cover, we establish logarithmic performance guarantees for online learning, active learning, and PAC learning. We furthermore show which algorithms and results carry over to learning intersections of monotone with anti-monotone concepts.

We then turn to the more complex setting of learning with a hidden context, i.e., the unknown concept depends on an unobservable variable that can change any time. In order to enable query learning algorithms to identify one of the true concepts, the usual query model is not sufficient. We hence propose a novel type of queries and give learning algorithms able to cope with concept drift due to hidden changes in the context. In particular, it is necessary (and sufficient) that the learning algorithm can query three different vertices at a time where it is ensured that the answers will be provided on the basis of the same concept. Worst case guarantees in this setting are related to adverserial learning.

The main purpose of this paper is to summarise our findings on the application of the size of the minimum pathcover as a concept-independent, transductive learning parameter for learning directed cuts with and without changing context. Due to space limitations we only sketch proofs.

2 Preliminaries

2.1 Directed Graphs

For any $k \in \mathbb{N}$ we denote $\{1, \ldots, k\}$ by $[\![k]\!]$ and the Boolean values '$true$','$false$' by $\top, \bot$, respectively with $\Omega = \{\top, \bot\}$.

A *directed graph* (*digraph*) is a pair (V, E) where V is the set of *vertices* and $E \subseteq V^2$ is the set of *edges*. For a digraph G we will sometimes denote its vertices by $V(G)$ and its edges by $E(G)$. The *induced subgraph* of $G = (V, E)$ by a subset of vertices $U \subseteq V$ is the digraph $G[U] = (U, E[U])$ where $E[U] = E \cap U^2$. A *subgraph* of G is any digraph $G' = (V', E')$ with $V' \subseteq V(G)$ and $E' \subseteq E[V']$.

For a digraph (V, E) and two sets $U, U' \subseteq V$ we define $E(U, U') = \{(u, u') \in E \mid u \in U \setminus U' \wedge u' \in U' \setminus U\}$. The *children* of U in a digraph (V, E) are then expressed as $\delta^+(U) = \{v \in V \mid (u, v) \in E(U, V \setminus U)\}$ and its *parents* as $\delta^-(U) = \{v \in V \mid (v, u) \in E(V \setminus U, U)$. The *contraction* of a set of vertices $U \subseteq V$ on a digraph $G = (V, E)$ is the digraph $(\{u\} \cup V \setminus U, E[V \setminus U] \cup (\{u\} \times E(U, V \setminus U)) \cup (E(V \setminus U, U) \times \{u\}))$, i.e., a new digraph which has U replaced by a single vertex $u \notin V$.

A *walk* in a directed graph G is a sequence $p_1 \dots p_i \dots p_n$ of vertices of G such that $\forall i \in [\![n-1]\!] : (p_i, p_{i+1}) \in E$ and $|p| = n$ is the *length* of the walk p. We denote the (possibly infinite) set of walks in G by $\mathcal{P}(G)$ and define the binary relations $\leq_G, \geq_G$ such that $u \geq_G v \Leftrightarrow v \leq_G u \Leftrightarrow \exists p \in \mathcal{P}(G) : p_1 = u \wedge p_{|p|} = v$. Whenever G is clear from the context it will be omitted. For any directed graph, these relations form a preorder on the vertex set. For any preorder $\leq$ and vertex set $U \subseteq V(G)$ we will use $\min U = \{u \in U \mid \nexists v \in U \setminus \{u\} : v \leq u\}$ and similarly $\max U = \{u \in U \mid \nexists v \in U \setminus \{u\} : u \leq v\}$. A digraph G is *strongly connected* iff $\forall u, v \in V(G) : u \geq v \wedge v \geq u$. A *strongly connected component* of a graph is a maximal strongly connected induced subgraph. A directed graph G is *acyclic* or a *DAG* iff it does not have a strongly connected component. On a directed acyclic graph the relations $\leq_G, \geq_G$ form a partial order on the vertex set. A walk on a directed acyclic graph is called a *path* or a *chain*. A set of vertices $U \subseteq V(G)$ is an *antichain* if $\forall u, v \in U : u \geq v \vee v \geq u \Rightarrow u = v$.

2.2 Learning Directed Cuts

Classification on graphs involves identifying a subset of the vertices, say $C \subseteq V$. We investigate identifying directed cuts. Formally, a set C defines a *directed cut* $E(C, V \setminus C)$ in a digraph if and only if $E(V \setminus C, C) = \emptyset$ (see, e.g., [1]). Due to the bijection between sets C defining directed cuts and the set of *cut edges* $E(C, V \setminus C)$ we will us the term *directed cut* to refer to C as well as to $E(C, V \setminus C)$. In the partially contaminated water system example, the uncontaminated water defines a directed cut corresponding to the set of pipes connecting contaminated to uncontaminated water.

In terms of the partial order $\geq$ induced by the digraph (V, E), a directed cut can then be seen as a set of vertices $U \subset V$ such that $\nexists u \in U, v \in V \setminus U : v \geq u$. Furthermore, identifying a directed cut can be seen as learning a labelling function $y : V \to \Omega$ where $\forall (u, v) \in E : y(v) \Rightarrow y(u)$. We call such a labelling a *monotone concept on a digraph* or *consistent with the directed graph* in the sense that the preorder of the vertices is respected by the labelling. A monotone concept y corresponds to the directed cut $C_y = \{v \in V \mid y(v)\}$ which we will sometimes call the *extension* of y and $E \cap (C_y \times V \setminus C_y)$ will be called the *concept cut*. The elements of the concept cut will be referred to as *cut edges*.

Without loss of generality it is sufficient to consider learning on DAGs. Any result for DAGs (defining a partial order) directly carries over to general digraphs (defining a preorder). This holds as directed cuts can never "cut" a strongly connected component. Learning on a general digraph is hence equivalent to learning on a digraph that has all strongly connected components contracted. Also, without loss of generality we can always add a 'super-source' and a 'super-sink' such that the new graph has a unique minimum and maximum.

2.3 The Cost of Learning

We consider the following question: Given an oracle that can provide us with hints about the fixed but unknown concept to be learned and some cost for each

hint, what is the worst case total cost needed to induce a satisfactory hypothesis? This question is important, for instance, when only a given budget is available for the task of finding a satisfactory hypothesis. Different concrete learning settings considered in literature are distinguished by the type of the hints given by the oracle, the cost function, and the definition of satisfactory hypotheses.

For *active learning* the oracle supplies the learning algorithm with the true label of any instance selected by the learning algorithm. Each hint, i.e., answer to a query, has a fixed cost and the hypothesis is satisfactory iff it is equal to the concept. Calls to the labelling oracle are called *membership queries*. For *online learning* the oracle selects an instance, makes the learning algorithm guess its label and then supplies the learning algorithm with the true label. Each wrong guess has a fixed cost and the hypothesis is satisfactory iff it is equal to the concept. The maximum total cost is the *worst case missclassification bound*. For *PAC learning* the oracle supplies the learning algorithm with a labelled instance drawn according to a fixed but unknown distribution. Each example has a fixed cost and a hypothesis is satisfactory iff it can be guaranteed that its error for examples drawn from the same distribution is smaller than some ϵ with probability larger than $1 - \delta$ for some δ. The total cost is the *sample complexity*.

As we are considering a transductive setting, i.e., the whole vertex set (without any labels) is input to the learning algorithm, we are interested in achieving costs logarithmic in the size of the graph. Hence, we consider a concept not learnable on a digraph in one of the above settings if we can not achieve polylogarithmic cost bounds. It can be seen that the general problem of identifying a directed cut is not learnable in any of the above settings. Consider the DAG $(\llbracket n+2 \rrbracket, (\{n+1\} \times \llbracket n \rrbracket) \cup (\llbracket n \rrbracket \times \{n+2\}))$. Given $y(n+1) \wedge \neg y(n+2)$, and any information about the vertices $\llbracket n \rrbracket \setminus \{u\}$ for some $u \in \llbracket n \rrbracket$, we can not infer the label of u. This implies that—in the worst case—it is impossible to identify the correct directed cut without costs in the order of the size of the graph.

2.4 Fixed Parameter Learnability

As directed cuts are in general not learnable, it is important to identify tractable subclasses. Performance parameters proposed in machine learning literature can be distinguished into concept-dependent and concept-independent parameters. Concept-dependent parameters define subclasses of the concept class. Concept-independent parameters define subclasses of the instance space. An example of the first case is the VC dimension of k-term monotone DNF formulae; an example of the latter case is the VC dimension of monotone Boolean formulae over n variables. Concept-independent parameters may be further distinguished into transductive ones and inductive ones.

The related setting of identifying a cut in an undirected graph (see e.g. [2] for a derivation of the VC dimension of small cuts) has been found to be fixed parameter learnable where the parameter is the size of the concept cut, i.e., the number of edges between differently labelled vertices. Although we can show that directed cuts are also fixed parameter learnable if the size of the directed concept cut is considered as the parameter, the dependency of the total cost on

Table 1. Total cost of learning monotone concepts as well as intersections of monotone and anti-monotone concepts in a DAG (V, E) for various cost models and fixed size of the minimum path cover Q^*. A '$=$' indicates an exact result for all directed graphs, '$\leq$' ('$\geq$') indicate tight upper (lower) bounds.

	Monotone Concepts	Intersections
Active query bound	$\leq \lvert Q^*\rvert \log \lvert V\rvert$	$= \lvert V\rvert$
Online mistake bound	$\leq \lvert Q^*\rvert \log \lvert V\rvert$	$\leq \lvert Q^*\rvert + 2\lvert Q^*\rvert \log \lvert V\rvert$
VC dimension	$= \lvert Q^*\rvert$	$\leq 2\lvert Q^*\rvert; \geq \lvert Q^*\rvert$
PAC sample complexity	$\leq \left\lceil \frac{\lvert Q^*\rvert}{\epsilon} \ln \frac{\lvert Q^*\rvert}{\delta} \right\rceil$	$\leq \left\lceil \frac{2\lvert Q^*\rvert}{\epsilon} \ln \frac{2\lvert Q^*\rvert}{\delta} \right\rceil$

a parameter of the concept is often not desireable. In particular in cases where we want or need to estimate the total cost of learning the cut a priori, concept-dependent parameters such as the size of the concept cut are not sufficient.

In this paper we concentrate on learning monotone concepts on DAGs for which the size of the minimum path cover of the vertices is bounded. This parameter is concept-independent and transductive. Hence, opposed to the size of the concept cut, it can be used to a priori estimate the total cost of classification.

3 Learning with Fixed Minimum Path Cover Size

Table 1 summarises the results that can be obtained when the size of the minimum path cover Q^* of the DAG is fixed. In all but one of the considered learning settings, monotone concepts (directed cuts) as well as the intersection of monotone and anti-monotone concepts are fixed-parameter learnable, i.e., the total cost of learning is $O(\lvert Q^*\rvert \log \lvert V\rvert)$. The remainder of this section first introduces path covers and then describes the performance guarantees in more details.

3.1 Path Covers of DAGs

A *path cover* of $U \subseteq V$ in a DAG $G = (V, E)$ is a set $Q \subseteq \mathcal{P}(G)$ of paths such that $U = \bigcup_{p \in Q} V(p)$. The algorithms we present later and/or their analysis rely on a minimum path cover of $V(G)$. We refer to an arbitrary minimum path cover as Q^* and assume wlog that Q^* contains only maximal paths and hence that each path starts at the source (labelled $\top$) and ends at the sink (labelled $\bot$).

The size of the minimum path cover is upper bounded by the maximum dicut which is hard to find. In contrast to the concept cut it is not lower bounded by the minimum dicut and can be computed in polynomial time (see, e.g., [3] for algorithms) from the graph without knowing the concept. It can indeed be much smaller than the concept cut.In fact, while the concept cut size is lower bounded by the minimum dicut, the minimum path cover can even be smaller than the

minimum dicut. Last but not least, the minimum path cover has the intuitive property that learning gets the easier the more edges we observe. Note that this is not the case when using the size of the concept cut.

3.2 Learning Monotone Concepts

To see that the results of Table 1 are tight when learning directed cuts (i.e. monotone concepts), consider the digraph $(\llbracket n{+}2\rrbracket, (\{n{+}1\}\times\llbracket n\rrbracket)\cup(\llbracket n\rrbracket\times\{n{+}2\}))$. Given $y(n+1)\wedge\neg y(n+2)$, and any information about the labels of the vertices $\llbracket n\rrbracket\setminus\{u\}$ for some $u\in\llbracket n\rrbracket$, we can not infer the label of u.

The algorithms for active and online learning perform binary search on each path in the path cover independently. For active learning this means always querying the vertex furthest from the nearest vertex with known or inferred label in the path, i.e., the vertex half way between the smallest known positive and the largest known negative of the path. With each query we can then learn the label of at least half of the remaining vertices. For online learning binary search means always predicting according to the closest vertex in the path. With each mistake we can deduce the label of at least half of the remaining vertices.

That the VC dimension is smaller than the size of the minimum path cover follows by induction over the path cover size and two observations: Firstly, on each path, monotone concepts can only shatter one vertex. Secondly, introducing more edges can not allow us to shatter more vertices. That the VC dimension is equal to the size of the minimum path cover follows then by Dilworth's theorem [4]. This theorem relates the size of the minimum path cover to the size of the largest antichain. Observing that each antichain can be shattered by monotone concepts gives the desired result. As efficiently finding a consistent hypothesis is trivial, directed cuts can be efficiently PAC learned. A direct proof of the sample complexity is very similar to the proof for axis-aligned rectangles and will be included in an extended version of the paper.

Notice that learning directed cuts generalises learning of monotone Boolean formulae by considering each Boolean instance as a vertex of a directed hypercube, i.e., the digraph $(2^{\llbracket n\rrbracket}, \{(U, U\cup\{v\}) \mid U\subseteq\llbracket n\rrbracket\wedge v\notin U\})$. An antichain of size $\binom{n}{\lceil n/2\rceil}$ in this graph is found as $\{U\subseteq V \mid |U| = \lceil n/2\rceil\}$ and that this is indeed the largest anti-chain follows from Sperner's theorem [5]. This VC dimension is exactly the one obtained (without path covers) in [6]. However, for data that is only a subset of the hypercube, our transductive guarantee can be (much) better. For instance, it can be seen that the size of the minimum path cover of a set of Boolean instances does not change when redundant attributes are introduced. Most other results on learning of monotone Boolean formulae consider particular sets of functions with restricted complexity, e.g. the size of the smallest decision tree representing the concept [7]. As complexity is a concept dependent parameter, we can not use it to estimate classification costs a priori. Our results are complementary to these bounds, as depending on the concept class and the training and test instances, one or the other bound is tighter.

3.3 Learning Intersections of (Anti-) Monotone Concepts

Given some functions $y_i : V \to \Omega$ monotone on a digraph (V, E) and constants $b_i \in \Omega$ we call the concept $y_\cap(v) = \bigwedge_i b_i \operatorname{xor} y_i(v)$ an *intersection of monotone and anti-monotone concepts* on G. The total cost of learning such concepts is summarised in the second column of Table 1.

To see that active learning is not possible in this setting, consider any ordering $i_1, \dots i_{|V|}$ in which an algorithms queries the vertices of any digraph. As the intersection concept class contains the empty set as well as any singleton vertex, given $y(i_1), \dots y(i_{|V|-1}) = \bot$ we can not conclude the label of $i_{|V|}$.

For online learning, on the other hand, a small modification of the above described algorithm for learning directed cuts suffices to obtain a mistake bound logarithmic in the number of vertices: As long as we have not observed any label on a path, we predict $\bot$. As soon as we make the first mistake on a path, we split the problem in two subproblems, each equivalent to learning a directed cut.

The proof of the VC dimension being upper bounded by twice the size of the minimum path cover follows along the lines of the proof of the monotone case. However, we now only have a bound, not an exact result. To see that there are graphs for which the VC dimension of intersections of monotone and anti-monotone concepts is smaller than twice the size of the minimum path cover, consider the graph $([4], \{(1,2),(2,4),(1,3),(3,4)\})$. This graph has an antichain of size two and hence no smaller path cover. However, we can not shatter all four vertices as we can never achieve that $y(1) = y(4) = \top = \neg y(2) = \neg y(3)$. The precise VC dimension depends in this case on the maximum $\mathcal{U} \subseteq [V]^2$ with $\forall U, U' \in \mathcal{U}, u \in U, u' \in U' : u \leq u' \vee u' \leq u \Rightarrow U = U'$.

4 Learning Despite Changing Concepts

In many real world learning problems the target concept may change depending on a hidden context such as unobservable variations in the environment. For instance, consider again the directed graph representing the assembly of a product from subparts. Learning a directed cut corresponds in this example to finding those stages of the process that introduced faults. Indeed, whether a fault is introduced at some stage or not, may depend on an unknown property, like variations in basic parts. Hence, the fault may occur only sometimes, depending on this hidden context. In order to be able to trade off between the cost of having a faulty product and the costs needed to find the cause of the fault, tight performance guarantees for learning with hidden contexts are needed.

For that, we consider a variant of active learning where the target concept $y_b : V \to \Omega$ also depends on a *hidden context* b that may change at any time. We can view this as having different monotone concepts C_b for each value of b and the oracle chooses a b for each query. For simplicity we restrict ourselves to Boolean contexts $b \in \Omega$. The aim of the learning algorithm is to discover either of the true concepts, $C_\top$ or $C_\bot$, despite the changing concepts.

To obtain worst case bounds for this setting, we consider the oracle as an adversarial 'evil' oracle that chooses the hidden context to maximise the cost of

learning either of the directed cuts. Although the context does not need to be (and often will not be) a function of the query, to prove negative results it will be useful to sometimes view it in this way. Key roles in the analysis of the learning algorithms will be played by the edges connecting vertices with different labels in one of the concepts, the *cut edges*, as well as by the elements of the symmetric difference of the two true concepts, the *distinction points* $C_\top \triangle C_\bot$ (the elements in either of the concepts but not in both).

Similar to the settings without a hidden context, a setting will be called learnable if there is a learning algorithm that can be guaranteed to identify one of the true concepts while asking only a number of queries (poly-) logarithmic in the number of vertices. However, we will now also encounter settings in which we can not find a true concept even by querying all vertices.

In the remainder we will show that traditional membership queries are not sufficient to learn directed cuts when a hidden context is present. One way to enable learning is to allow the algorithm to ask for the label of multiple vertices at the same time, which the oracle is then forced to answer according to the same concept. We will call these "m-ary" queries, where $m \in \mathbb{N}$ is the number of vertices that can be queried at once and will be answered from the same concept, i.e., during one m-ary query the context does not to change. We use the notation $\langle\cdot\rangle_m : V^m \to \Omega^m$ for m-ary queries. We will show that 2-ary queries are only slightly more powerful than 1-ary queries and that in general directed cuts with a hidden context are not learnable with 2-ary queries. However, we will also describe an algorithm that is able to correctly identify one of the true concepts with logarithmically (in the size of the graph) many 3-ary queries.

4.1 1-Ary Membership Queries

With traditional membership queries, directed cuts with hidden context are not learnable even if the size of the minimum path cover is fixed. We distinguish two cases depending on the size of the symmetric difference $C_\bot \triangle C_\top$ of the concepts.

Consider first the case $|C_\bot \triangle C_\top| \geq 2$. Here, we can find an adversarial context such that any learning algorithm can discover neither of the two true concepts. Let $v \in \max C_\bot \triangle C_\top$ where the maximum is taken with respect to the partial order induced by the DAG and assume without loss of generality that $v \in C_\top$. Then, any vertex before v in the partial order (an ancestor) belongs to both $C_\top$ and $C_\bot$. Hence, the set $S = C_\bot \cup \{v\}$ is monotone. The oracle can pretend to answer from S by choosing $C_\bot$ for all queries u with $u \neq v$ and choosing $C_\top$ only if $u = v$. Observing these answers from the oracle, the learning algorithm can not guess either of the true concepts.

In fact, even in the extreme case with the symmetric difference $|C_\bot \triangle C_\top| \leq 1$, directed cuts are not learnable: The answers of the oracle can not be distinguished from the above case without knowing $|C_\bot \triangle C_\top| \leq 1$.

For an illustrative example consider the path $([\![4]\!], \{(1,2),(2,3),(3,4)\})$ with the concepts $C_\bot = [\![1]\!]$ and $C_\top = [\![3]\!]$. If the query $u \in [\![2]\!]$, the oracle answers according to $C_\top$, otherwise according to $C_\bot$. Then, no matter the queries, the learning algorithm will only observe the 'fake' concept $[\![2]\!]$. Now, consider the

concepts $C_\perp = [\![1]\!]$ and $C_\top = [\![2]\!]$. Even if the oracle answers consistently according to $C_\top$, the learning algorithm has no means of distinguishing this case from the previous one. Even more surprisingly, also in the seemingly trivial case $C_\top = C_\perp = [\![2]\!]$, we have no means to distinguish the answers of the oracle from the answers in the previous case.

4.2 2-Ary Membership Queries

We consider now 2-ary membership queries, i.e., the learning algorithm can ask for the label of two vertices simultaneously and the answers will be given from the same concept. After each 2-ary query, the context may change. We denote 2-ary queries of vertices v and v' by $\langle v, v' \rangle_2$.

As one would expect, 2-ary queries are more powerful than traditional (1-ary) queries. For an example where this holds, consider a single path $([\![n]\!], \{(i, i+1) \mid i \in [\![n-1]\!]\})$ and coinciding concepts $C_\top = C_\perp = [\![c]\!]$ for some $1 < c < n$. In this case we can find out whether $[\![i]\!]$ is a true concept or not by simply querying the vertices $\langle i, i+1 \rangle_2$ in one 2-ary query. If the answer is $\langle i, i+1 \rangle_2 = (\top, \perp)$ we have found a true concept ($i = c$) otherwise not. As we assumed $C_\top = C_\perp$ we can find this cut edge with logarithmically many queries.

However, already the simple setting of a single path with symmetric difference $|C_\top \triangle C_\perp| \geq 1$ is not learnable with 2-ary queries. In this case, no matter which queries the learning algorithm asks, the oracle can always answer such that it only reveals a distinction point (an element of the symmetric difference) and 'hides' the concept cut. In fact, a naive learning algorithm that queries all pairs of vertices connected by an edge suffices to find at least one distinction point. If the oracle would try not to reveal a distinction point, it would need to answer one 2-ary query of adjacent vertices by $\top, \perp$ and it would hence reveal us one true concept, which it will—of course—avoid.

If we find a distinction point we can query the pair of vertices adjacent to the distinction point, say u, v. Through the answer we can either identify another distinction point or we can be sure that one of the two concepts has a cut bettween u and v. We can repeat this procedure, now querying the vertices adjacent to the distinction area, until we find no further distinction point. Then, we can be sure that the distinction area is adjacent to a cut edge, however, we cannot be sure which side of the distinction area is adjacent to the cut.

Consider the illustrative example $([\![3]\!], \{(1,2),(2,3)\})$. We will show that even when asking the full set of possible query combinations, the corresponding answers can be such that deducing one of the true concepts is not possible. Suppose now, all possible 2-ary queries are answered as follows: $\langle 1,2 \rangle_2 = (\top, \top)$; $\langle 2,3 \rangle_2 = (\perp, \perp)$; $\langle 1,3 \rangle_2 = (\top, \perp)$. The learning algorithm knows from these answers that: First, vertex 2 belongs to the symmetric difference of the two true concepts, i.e. $2 \in C_\top \triangle C_\perp$; second, there is one of the true concepts that contains vertices $1, 2$; and third, there is one of the true concepts that does not contain vertices $2, 3$. Yet this information does not allow the learning algorithm to certainly identify a true concept. It might be that: (*i*) $C_\top = [\![1]\!], C_\perp = [\![2]\!]$; (*ii*) $C_\top = [\![1]\!], C_\perp = [\![3]\!]$; and (*iii*) $C_\top = \emptyset, C_\perp = [\![2]\!]$. For each possibility, there is a context such that the

oracle will produce the above mentioned set of answers. Hence, the learning algorithm has no means to distinguish these possibilities. Notice that in this case knowing $|C_\top \triangle C_\bot| = 1$ would help us identify both true concepts while even knowing $|C_\top \triangle C_\bot| = 2$ would not be sufficient to surely identify one concept.

4.3 3-Ary Membership Queries

In this section, we consider 3-ary membership queries. We give an algorithm that is guaranteed to find one of the true concepts for any DAG and any pair of concepts with at most $|Q^*|^2 + 2|Q^*| \log |V|$ many 3-ary membership queries. The learning algorithm can be summarised in three steps:

1. Find an edge that corresponds to a cut edge in one of the true concepts on each path in the path cover or find a distinction point ($\leq |Q^*| \log |V|$).
2. Check that the cut edges (from step 1) on each path belong to the same true concept or find a distinction point ($\leq |Q^*|^2$).
3. Given a distinction point, find any of the two concepts ($\leq 2|Q^*| \log |V|$).

Note that the main difference between 3-ary queries and 2-ary ones is that as soon as we have a distinction point, we can use it to perform a coordinated binary search in both true concepts at the same time.

Step one (*finding a cut edge or a distinction point*) could in principle proceed as described above for 2-ary membership queries, however, with 3-ary queries we can find the cut or a distinction point on each path p in $\log |V|$ many queries: We repeatedly perform binary search by querying the smallest known positive on p, the largest known negative on p as well as the vertex half way between them. As long as we do not get a distinction point on a path, we assume all answers of the oracle correspond to the same concept. Notice that even if the oracle changed the concept but the answers stay consistent with those received so far, the strategy of the learning algorithm needs no change. Unless we get a distinction point, we can reduce the number of vertices on this path for which we do not know any label by half. Hence, after $\log |V|$ many queries we will either get a distinction point or a cut edge. If we do not have a distinction point yet, we proceed with the same procedure on the next path.

Step two (*finding a distinction point*) needs to check wether the individual cut edges that we found in step one for each path, correspond all to the same concept. If this is the case then we are done, otherwise we find a distinction point. This can be achieved with $|Q^*|^2$ many queries as follows: For a pair of paths $p, q \in Q^*$ denote the corresponding cut edges found in step one by (u_p, v_p) and (u_q, v_q), respectively. The learning algorithm can find out if there is one true concept which both edges correspond to, by asking only two 3-ary queries $\langle u_p, v_p, u_q \rangle_3$ and $\langle u_p, v_p, v_q \rangle_3$. If the oracle answers consistently with its previous answers then such a true concept exists. If the oracle deviates from its previous answers, we have found a distinction point. We proceed with the next pair of paths until we get a distinction point or have checked that there is a concept in which all edges found in step one are correct cut edges.

Step three (*finding a concept given a distinction point*) is the simplest of the three steps and will be performed after in step one or two the learning algorithm found a distinction point. It proceeds simply by performing a binary search on each path in the path cover for both concepts at the same time. The 3-ary query will always include the distinction point as one of the queries, and the two remaining ones will be used for the binary search. After at most $2|Q^*| \log |V|$ many queries (in fact we may subtract the number of queries made in step one) we can be sure to have found one of the two concepts.

5 Related Work

Recently, the use of cuts in learning on graphs is becoming more and more popular. Usually motivated by the 'cluster assumption' [8] one tries to find a small cut separating positive labeled vertices from negative labeled ones. for instance, work in [9] extends an earlier approach based on finding the minimum cuts in a graph by adding randomness to the graph structure. In [2] it is shown that the VC dimension of small cuts is bounded by the size of the cut. The disadvantage of these approaches is that the performance depends on a property of the concept and not on an efficiently computable property of the graph like the minimum path cover. Furthermore, in the kind of applications we mentioned in the introduction, it is questionable whether the concept cut is indeed small.

It is also possible to derive concept-dependent results for learning directed cuts. We show this here for active learning: As long as there is a walk from source to sink, choose the shortest such walk and perform binary search on this path. As soon as a concept cut edge is found, remove it and iterate. At least every $\log |V|$ iterations a concept cut edge is removed and the algorithm terminates with the corret cut after as many iterations as there are edges in the cut.

Learning directed cuts is also closely related to learning monotone Boolean formulae as these can be modelled by directed cuts on the directed hypercube. Most performance guarantees for learning monotone Boolean formulae are concept-dependent. We showed that our results imply concept-independent and transductive performance guarantees for learning monotone Boolean formulae.

Learning with concept drift due to hidden context has for instance been investigated in [10,11]. The setting is different from ours in that once a concept is learned, it can become invalid only after a certain interval of time. To the best of our knowledge, learning monotone concepts with hidden context that can change at arbitrary points in time has not been investigated.

6 Conclusions

The question whether a learning task is feasible with a given budget is an important problem in real world machine learning applications. Recent learning theoretical work has concentrated on performance guarantees that depend on the complexity of the target function or at least on strong assumptions about the target function. In this paper we proposed performance guarantees that only

make natural assumptions about the target concept and that otherwise depend just on properties of the unlabelled training and test data. This is in contrast to related work on learning small cuts in undirected graphs where usually the size of the concept cut is taken as a (concept-dependent) learning parameter.

Concepts on digraphs, which are a natural model, e.g., for technical processes, are typically monotonic. In this paper we proposed the size of the minimum path cover as a performance parameter for learning directed cuts (i.e., monotone concepts on digraphs). On the one hand, this parameter can efficiently be computed from unlabelled training and test data only; and is, on the other hand, sufficiently powerful to make directed cuts fixed parameter learnable in the active, online, as well as PAC learning frameworks.

In many real world learning problems, an additional challenge is often that the concept that a learning algorithm tries to identify changes depending on some hidden context. We hence extended the usual query learning model to include a hidden context that can change at any time and explored learnability with a more powerful query model. In particular, we show that to enable learnability despite a changing concept, it is necessary (and sufficient) that the learning algorithm can query three different vertices at a time where it is ensured that the answers will be provided on the basis of the same concept. While, in this paper, we concentrated on learning on directed graphs, we believe that this setting can also have significant impact in other domains where some relevant variables are unobservable. We will study such domains as part of our future work.

References

1. Korte, B., Vygen, J.: Combinatorial Optimization: Theory and Algorithms. Springer, Heidelberg (2002)
2. Kleinberg, J.: Detecting a network failure. In: FOCS, IEEE, Los Alamitos (2000)
3. Ntafos, S., Hakimi, S.: On path cover problems in digraphs and applications to program testing. IEEE Transactions on Software Engineering (1979)
4. Dilworth, R.: A decomposition theorem for partially ordered sets. Annals of Mathematics (1948)
5. Sperner, E.: Ein satz über untermengen einer endlichen menge. Mathematische Zeitschrift (1928)
6. Procaccia, A.D., Rosenschein, J.S.: Exact vc-dimension of monotone formulas. Neural Information Processing — Letters and Reviews (2006)
7. O'Donnell, R., Servedio, R.A.: Learning monotone decision trees in polynomial time. In: IEEE Conference on Computational Complexity, IEEE, Los Alamitos (2006)
8. Chapelle, O., Zien, A.: Semi-supervised classification by low density separation. In: AISTATS (2005)
9. Blum, A., Lafferty, J., Rwebangira, M.R., Reddy, R.: Semi-supervised learning using randomized mincuts. In: ICML'04, vol. 13 (2004)
10. Harries, M.B., Sammut, C., Horn, K.: Extracting hidden context. Machine Learning 32(2), 101–126 (1998)
11. Widmer, G., Kubat, M.: Effective learning in dynamic environments by explicit context tracking. In: ECML '93, pp. 227–243 (1993)