Learning Distributions from Random Walks [*]

Funda Ergün **S Ravi Kumar** **Ronitt Rubinfeld**

Department of Computer Science
Cornell University
Ithaca, NY 14853.

Abstract

We introduce a new model of distributions generated by random walks on graphs. This model suggests a variety of learning problems, using the definitions and models of distribution learning defined in [6]. Our framework is general enough to model previously studied distribution learning problems, as well as to suggest new applications. We describe special cases of the general problem, and investigate their relative difficulty. We present algorithms to solve the learning problem under various conditions.

1 INTRODUCTION

In this paper, we introduce a new model of distributions generated by random walks on graphs. This model suggests a variety of learning problems, using the definitions and models of distribution learning defined by Kearns *et. al.* [6]. Our framework is general enough to model various noise processes, the Hamming ball distribution learning problem studied by [6], and the evolutionary tree model studied by Farach and Kannan [2]. Other possible applications to problems in context-sensitive spelling correction and unsupervised learning are suggested.

In the most general description of our framework, the distributions are generated by the following process, which takes a specific graph, whose edges are labeled with transition probabilities, as input: First, a starting node is chosen according to a distribution from an allowable subset of nodes of the graph, called *centers*. Second, a random walk starting at the chosen center is taken according to transition probabilities on the edges of the graph. Finally, the name of the last node of the random walk is output.

The parameters defining the distribution are the set of centers that the walks start from, the length of the walk, the distribution according to which a center is chosen, the structure of the graph, and the transition probabilities. On this model it is natural to define a variety of learning problems depending on which parameters are known and which are unknown to the learner.

In the model of distribution-learning defined in [6], the learner has access to a set of samples generated according to the specific distribution that it is trying to learn. The random bits of the process that generates the distribution are totally hidden from the learner, but the learner has access to certain information about the nature of the process generating the distribution. One possible learning problem in this context involves efficiently constructing a mechanism to generate a distribution that is close to the target distribution, i.e., *learning a generator*. Another learning problem involves efficiently constructing a mechanism which given a string, evaluates the probability that it is output by the distribution to within a good approximation, i.e., *learning an evaluator*. Our methods allow one to learn both an evaluator and generator as defined in [6].

We describe in precise terms several interesting special cases of this general problem. The first is the problem of learning the distribution when only the centers are unknown to the learner, called the *hidden centers* problem. Another special case is the problem of learning the distribution when the transition probabilities are unknown, called the *hidden transitions* problem. We show that the hidden transitions problem is at least as hard as the hidden centers problem.

Next, we focus on finding learning algorithms for the hidden centers problem on large classes of graphs. We begin with an algorithm that nontrivially approximates distributions from bounded degree graphs with certain properties, that was inspired by an algorithm from [6]. Our running time is polynomial in $k \cdot d^l$ where k is the number of centers, d is the degree bound of the graph and l is the length of the walks, but depends only logarithmically on the number of nodes in the graph. This is useful for applications that have very large numbers of nodes but low degree. We then investigate a second

[*]This work is partially supported by an NSF Career Award CCR-9624552 and the Alfred P. Sloan Research Award. The first and second authors are also supported by NSF grant DMI-91157199.

line of algorithms which uses linear algebra techniques to approximate the target distribution to within any desired accuracy in the L_1 norm. We give an algorithm that works for graphs that have well-conditioned transition probability matrices. The algorithm runs in time polynomial in the number of nodes, but outputs a hypothesis whose size depends only on the number of centers. This leads to space efficient hypotheses that are attractive when the graph is large but the number of centers is small. We also give a more efficient algorithm for graphs whose transition matrices and submatrices of transition matrices are well-conditioned.

1.1 APPLICATIONS

We describe how previously suggested distribution learning problems can be viewed within our framework. We also suggest other settings in which our framework may be of interest.

Hamming Ball Distributions. In [6], the problem of learning Hamming ball distributions is studied. This model is natural for concepts in which there are "canonical" examples of the concept, and in which the probability of an output decreases as the number of attributes in common with the canonical examples decreases. More precisely, there is a set K of centers, each of which is an n-bit binary vector. The output distribution is defined as follows: Choose a center uniformly from K, flip each of its bits independently with probability p, and then output the resulting vector.

The Hamming ball distributions are equivalent to distributions generated by random walks on the following graph: Let G be the complete graph on 2^n nodes, each node representing a different n-bit binary vector. The probability of going from vector u to v depends on the number of bits flipped to turn u into v — if the Hamming distance between u and v is h, then the edge (u, v) is assigned the probability $p^h(1-p)^{n-h}$. The learning problem of [6] is exactly the distribution learning problem from random walks of length 1 on G, where the transition probabilities are known but the centers are unknown.

We mention the *hypercube distribution*, that is related, but not equivalent, to the Hamming ball distribution: Let G be the complete graph on 2^n nodes, each node representing a different n-bit binary vector. The probability of going from vector u to vector v is $\frac{1}{n+1}$ if the Hamming distance between u and v is ≤ 1, and 0 otherwise. The distribution is generated by choosing a random center and performing a random walk of length pn.

Noise Processes. In general, our framework can also be viewed as follows: the centers are "prototypes" and the random walk on the graph is a "noise" process that perturbs the prototypes. In this framework, a walk along an edge corresponds to a "change" of an attribute of the prototype. The Hamming ball and the hypercube distributions are natural graphs on which to study such noise processes.

Cavender-Farris Distributions. The second model that fits into our framework is the model of Cavender-Farris Trees used to study evolution [2]. In this model there is a rooted (directed) tree with probabilities associated with each edge. The nodes have labels from $\{0,1\}$. The label of the root is determined according to some probability p_r. The nodes are labeled as follows. For any edge (u, v), let P_{uv} denote the probability on an edge (u, v). Then, the label on node v is determined by flipping the label of node u with probability P_{uv}. The distribution output is the vector containing the labels of each of the leaf nodes in some predefined order. The learner, however, is not given the probabilities used to generate the labels. Given a set of such vectors, it is shown how to reconstruct the probabilities on the tree as well as the distribution on the initial bit at the root, thus giving a learning algorithm for the distribution.

This model is a special case of our model where the probabilities on the edges of the graph are not known to the learner. We show this by a reduction from the problem of learning the tree T in the model of [2] to a learning problem in our model. Given T, let G be a leveled graph whose nodes have labels that are of the form (i, x), where i denotes the level containing the node and x is an n-bit binary vector. At the first level are the two nodes $(0,(0,\ldots,0))$ and $(0,(1,\ldots,1))$. Construct level $i+1$ from level i as follows. The nodes in level $i+1$ are $(i+1, x)$ for all possible n-bit binary vectors x (even though some of them will be unreachable from level i at the end of the procedure and can be deleted). Given two nodes $u = (i, x)$ and $v = (i+1, x')$, use the following procedure to determine whether to insert an edge from u to v. Call as *the state of level* i in T the string that would be output at the leaves if no bit flipping were to take place in any level below i, given the current node labels at state i. Add an edge (u, v) to G if the structure of T allows level $i+1$ to have state x' given that level i has state x. For instance, if T is a binary tree of height 2, (0000) at level 0 (corresponding to a 0 at the root) is not consistent with (0111) at level 1, since the only valid states for level 1 would be (1111), (1100), (0011), and (0000). Once G has as many levels as T, the construction is complete. The two top-level nodes are the centers, and the hypothesis class is restricted by the dependencies among probabilities resulting from the fact that there is a many-to-many dependence (which can be expressed explicitly) between the set of probabilities on T and those on G. The answer to the learning problem on G from this hypothesis class then can be used to determine the probabilities on T.

Context-Sensitive Spelling Correction. Another setting in which that our model may be useful is one similar to that studied by Golding and Roth [3] for use in context-sensitive spelling correction. They study the problem of finding and correcting spelling mistakes which are not found by dictionary based approaches, because the misspelled word is also an English word. An example is "I had fruit pie for desert tonight," where the spell checker should know to change "desert" to "dessert". Words that need to be disambiguated are modeled by *confusion sets*. In our example, the confusion set is $\{desert, dessert\}$. One method that is suggested associates every member of a confusion set with a characteristic vector containing the words and grammatical structures that are likely to occur in the vicinity of the given word. The "correct" version of each word in the text is determined using a Bayesian approach. Given the characteristic vectors of each member of the confusion set and the corresponding information about the current word, the member of the confusion set that was most likely to have generated the current word is chosen. Since even words within confusion sets have multiple usages, it may be useful to represent the word as a *collection* of characteristic vectors. For each member of a confusion set, one can think of the true characteristic vector as a set of centers which generates a distribution on the observed vectors via a random walk on a graph in which nodes correspond to vectors (such as the hypercube). Given the observed vectors of a single member of the confusion set, the learner learns the distribution on the vectors. Then, the spell checker may use the distribution information in order to choose the most likely member of the confusion set analogously to [3].

Classification and Unsupervised Learning. Finally, consider a classification system in which a small number of classes recursively branch off into smaller classes. For instance, the class of credit card holders have yuppies and women as subclasses. These subclasses have further subclasses of their own, such as patrons of a specific restaurant. It is possible for a subclass to be part of two larger classes. A natural way to model such a system is via a directed acyclic graph. Consider the distributions generated from random walks starting at one of a small number of centers. Learning algorithms for such distributions might be used to suggest interesting subclasses of people to target for marketing promotions. One such classification system is readily available as part of the Penn Tree Bank.

2 PRELIMINARIES

Given any distribution D on $X = \{0,1\}^n$, let $D(x)$ denote the probability that D outputs x. We use $x \in D$ to denote that x is chosen according to probability distribution D. We use $[n]$ to denote $\{1, 2, \ldots, n\}$.

Given two distributions D_1, D_2 on $\{0,1\}^n$, some measures that capture the distance between them are,

(a) the L_1-norm defined by

$$L_1(D_1, D_2) = \sum_{x \in X} |D_1(x) - D_2(x)|,$$

(b) the Kullback-Leibler (KL) divergence defined by

$$\mathrm{KL}(D_1 || D_2) = \sum_{x \in X} D_1(x) \log \frac{D_1(x)}{D_2(x)}.$$

Note that the first is a true metric while the second is not, since it does not satisfy symmetry and triangle inequality. KL-divergence, however, captures the distance in an information-theoretic sense [1].

In order to formally define what it means to learn a distribution, we need the following definitions of a generator and an evaluator for a distribution from [6].

Definition 1 *Let $\mathcal{D}_n$ be a class of distributions over $\{0,1\}^n$. We say that $\mathcal{D}_n$ has* polynomial-size generators *if there are polynomials $p(\cdot)$ and $r(\cdot)$ such that for any $n \geq 1$, and for any distribution $D \in \mathcal{D}_n$, there is a circuit F_D, of size at most $p(n)$ and with $r(n)$ input bits and n output bits, whose induced distribution on $\{0,1\}$ is exactly D when the distribution of the $r(n)$ bits is uniform. Thus, if $\overline{r} \in \{0,1\}^{r(n)}$ is a randomly chosen vector, then the random variable $F_D(\overline{r})$ is distributed according to D.*

Definition 2 *Let $\mathcal{D}_n$ be a class of distributions over $\{0,1\}^n$. We say that $\mathcal{D}_n$ has* polynomial-size evaluators *if there is a polynomial $p(\cdot)$ and such that for any $n \geq 1$, and for any distribution $D \in \mathcal{D}_n$, there is a circuit E_D, of size at most $p(n)$ and with n input bits, that, on input $\overline{y} \in \{0,1\}^n$ outputs the binary representation of the probability assigned to $\overline{y}$ by D. Thus, if $\overline{y} \in \{0,1\}^n$, then $E_D(\overline{y})$ is the weight of $\overline{y}$ under D. We call E_D an evaluator for D.*

One can treat an evaluator E as a distribution where the probabilities are defined according to the output of E.

We define what it means to learn a distribution in time $t(\epsilon, \delta, n)$. Our definition is based on the ones in [6]:

Definition 3 *Let $\mathcal{D}_n$ be a class of distributions. We say that $\mathcal{D}_n$ is* ϵ-learnable *in time $t(\epsilon, \delta, n)$ under distance measure d with a generator (resp. evaluator) if there is an algorithm that, when given inputs $0 < \delta \leq 1$, $\epsilon > 0$ and access to an oracle for any unknown target distribution $D \in \mathcal{D}_n$, runs in time $t(\epsilon, \delta, n)$ and outputs a generator F (resp. an evaluator E) that with probability at least $1 - \delta$, satisfies $d(D, F)$ (resp. $d(D, E)$) $\leq \epsilon$.*

Analogous definitions of learning a class of distributions by a hypothesis class can be made.

3 THE RANDOM WALK MODEL

Distributions Generated by Random Walks. Our distributions come from a special class $\mathcal{D}_n$ generated by random walks on a graph in the following manner.

Let $G = (V, E)$ be a directed graph where the nodes have labels from $\{0,1\}^n$ for some $n > 0$. We use N to denote $|V|$, $N \leq 2^n$. P is an assignment of transition probabilities to the edges of G such that for all vertices $v \in V$, the probabilities on the transitions out of v sum to 1. P can be represented by a transition matrix in the following manner: Each node is associated with both a row and column in P, and the entry P_{ij} is equal to the probability of a transition to node i given that one is at node j. The columns of P (but not necessarily the rows) sum up to 1 (P is *stochastic*). Since G can be reconstructed by including edges for each transition that has nonzero probability in P, we only need to specify P in order to specify the distribution. However, it is often convenient to refer to the edges and nodes of the graph. Let K be a distribution on V. K will typically have nonzero support only on a small subset of V, referred to as the *centers.* L is a distribution on positive integers.

Definition 4 $\mathrm{rw}(P, L, K)$ *is a distribution where an output is generated by the following procedure: Pick an integer l according to L. Pick a node v according to K. Take a random walk starting from v and picking edges according to the probabilities dictated by P. After the l^{th} step output the current node.*

The Learning Model. In this model the learner has access to a set X of samples independently generated from the target distribution. Let $\mathcal{P}_N$ be a class of graphs on N nodes and their transition probabilities. Let $\mathcal{K}_N$ be a class of distributions on N nodes. Let $\mathcal{L}$ be a class of distributions on positive integers. The concept class $\mathcal{C}_N \subseteq \mathcal{P}_N \times \mathcal{L}_N \times \mathcal{K}_N$ is a set of distributions where each member of $\mathcal{C}_N$ is of the form $\mathrm{rw}(P, L, K)$, for $P \in \mathcal{P}_N$, $L \in \mathcal{L}_N$ and $K \in \mathcal{K}_N$. Depending on the nature of the learning problem, this class is restricted in terms of the class $\mathcal{C}_N$. For instance, if the graph and the transition probabilities (P^*) are assumed to be known by the learner, then the hypothesis class of the learner is restricted to $\{P^*\} \times \mathcal{L}_N \times \mathcal{K}_N$. Since some combinations of restrictions are more natural than others, we give names to them. When all parameters are known except for K, we call the learning problem the *hidden centers problem.* We investigate a special case of the hidden centers problem, when $\mathcal{K}_N$ contains only $\mathcal{K}_N^k$, the distributions that have weight on at most k nodes. If P is hidden but all else is known we call it the *hidden transitions problem.* We explore relationships between variants of the problem in Section 4.

4 RELATIONSHIPS BETWEEN VARIANTS OF THE LEARNING PROBLEM

Depending on the amount and nature of the information that is available to the user, the learning problem and its difficulty varies. In this section we show some relationships between different variants of the problem.

4.1 HIDDEN CENTERS VERSUS HIDDEN TRANSITIONS

It can be shown that learning the distribution where the probabilities on the edges are not known (hidden transitions) is at least as difficult as when the distributions on the centers is not known (hidden centers).

Given a distribution P, let $\mathcal{P}^+$ be the class of transition probabilities that can be achieved by adding one more node to the graph with indegree 0. More formally, $\mathcal{P}^+ = \{P^+ \mid P^+_{u,v} = P_{u,v}\ \forall u, v \in [N], \sum_{v\in[N]} P^+_{v,N+1} = 1,\ P^+_{N+1,v} = 0,\ \forall v \in [N]\}$. Let K^+ be distribution on $[N+1]$ which outputs $N+1$ with probability 1. Given distribution L, define L^+ as follows: $L^+(u+1) = L(u)\ \forall u \in \mathcal{Z}$. Given a class of distributions $\mathcal{L}$, define $\mathcal{L}^+$ to be $\mathcal{L}^+ = \{L^+ | L \in \mathcal{L}\}$.

Theorem 5 *If there is an ϵ-learning algorithm with a generator (resp. evaluator) for the hidden transitions concept class $\mathcal{P}^+ \times \mathcal{L}^+ \times \{K^+\}$, then there is an ϵ-learning algorithm with a generator (resp. evaluator) for the hidden centers concept class $\{P\} \times \mathcal{L} \times \mathcal{K}$.*

Proof. We give a learning algorithm for $\{P\} \times \mathcal{L} \times \mathcal{K}$. Suppose the target distribution is $D = \mathrm{rw}(P, L, K)$. Consider a new distribution $D' = \mathrm{rw}(P', L', K')$ as follows. Let $K' = K^+$. Let the graph underlying P' be the same as that underlying P, with the addition of an extra node $N+1$. Node $N+1$ has no incoming edges, and has an outgoing edge to every center in the support of K, with transition probability $K(v)$. Note that D' outputs the same distribution as D and D' is in $\mathcal{P}^+ \times \mathcal{L}^+ \times \{K^+\}$. Thus an algorithm for learning $\mathcal{P}^+ \times \mathcal{L}^+ \times \{K^+\}$ by a generator (resp. evaluator) gives an algorithm for learning $\{P\} \times \mathcal{L} \times \mathcal{K}$. □

4.2 FIXED VERSUS BOUNDED LENGTH WALKS

Given an algorithm to learn in polynomial time distributions generated by fixed length walks on a graph, it is often possible to learn certain types of distributions generated by bounded-length walks. For the hidden centers problem, this is achieved through taking the graph for the bounded-length walk distribution, adding self-loops to every node v such that the transition probabilities on the self-loops determine the distribution on the bounded-length walks, and scaling the transition probabilities of the other edges directed out of v so that they sum to 1. The learner then learns the distribution on the new graph for fixed-length walks.

For example, on level graphs, if one can learn distributions generated by fixed length walks one can learn distributions generated by walks with the distributions on lengths that correspond to sums of independently chosen random $(0, 1)$ variables. More precisely, given any $p_1, \ldots, p_l$ such that $0 \leq p_i \leq 1$, define random variable $L = \sum_{i=1}^{l} L_i$, where L_i is a random $(0, 1)$ variable that is equal to 1 with probability p_i. On the level graph, nodes at level i are given self-loops with transition probability p_i, and the transition probabilities of the other edges directed out of i are scaled so that the probabilities of all edges directed out of i sum to 1. Then, distributions generated by random walks of length l on the new graph are equivalent to distributions generated by random walks of length distributed as L on the old graph.

For graphs that are not leveled, adding a self-loop of constant probability p at each node generates a distribution corresponding to walk lengths distributed according to $L(i) = (1-p)^i p^{l-i}$, which is the normal distribution.

5 HIDDEN CENTERS VIA SET COVER

In this section we consider the concept class $\mathcal{C}_N^{\beta,l,k,d}$ of distributions of the type $\mathrm{rw}(P, L, K)$, where the outdegree of any node in the graph is at most d, and at any one node, all of the nonzero outgoing transition probabilities are equal. L is a distribution that always picks a fixed integer l. K is the uniform distribution on an unknown subset of k centers from P. Let $0 \leq \beta \leq 1$ be such that for any walk of length l starting from any node in the graph and progressing according to P, the probability of being at node v after l steps is at most β^l for any node v.

For graphs with good expansion, β is known to be small by the following fact. Let λ_2 denote the second eigenvalue of a graph.

Fact 6 *Let $P^t(u, v)$ denote the probability that a random walk on an expander starting from node u is at node v after t steps and let $\Pi(u)$ denote the stationary probability of u. Then, $P^t(u, v) = \Pi(v) + O(\lambda_2^t \sqrt{\Pi(v)/\Pi(u)})$.* [1]

Since λ_2 is bounded by a constant less than one for expanders, by this fact, $\beta^l = \max_{v,u} P^l(u, v)$ is bounded away from one.

In [6], the learning problem is solved by finding a good set of centers and defining K' as the uniform distribution over this set. They show that the distribution generated by picking the starting points according to K' is close to the distribution generated by picking them according to K. The algorithm presented here is inspired by the one in [6].

Algorithm. Let X be the set of samples and $m = |X|$. To produce an approximation to $D_T = \mathrm{rw}(P, L, K)$, we first form a list of nodes that are likely candidates for centers as follows. We take all the nodes that are exactly l steps away from every $x \in X$ and include them in our list of candidate centers. For every candidate node y in this list, we keep an associated set of sample points that one can reach in l steps from y. The next step is to cover the set X of sample points using these smaller subsets and to return the candidate centers associated with the subsets used in the set-cover as our set of centers. We perform the set-cover using a greedy approximation algorithm as in [6]. This algorithm returns at most $k \lg m$ subsets (and corresponding centers). Then our hypothesis distribution is $D_H = \mathrm{rw}(P, L, K')$, where K' is the uniform distribution on the centers. For technical reasons, we mix D_H with the uniform distribution and return $D_{H'}(x) = (1-q)D_H(x) + q/N$. We will determine q later.

Let

$$\rho = \lg\left(\frac{N^2 d^l k \lg m}{N^2 - N + d^l k \lg m}\right) - \frac{l}{(\beta d)^l k} \lg \frac{1}{\beta}$$

[1]When the indegrees and outdegrees equal d, Π is the uniform distribution.

Theorem 7 *The above algorithm is a $(\rho + \epsilon)$-learning algorithm for the class $\mathcal{C}_N^{\beta,l,k,d}$ under the* KL*-divergence that runs in time polynomial in $(\frac{k^2 \lg^6 N}{\epsilon^4} + \frac{\lg^2 n}{\epsilon^2} \lg \frac{2}{\delta}) d^l$*

The proof of Theorem 7 follows directly from the following theorem and its proof.

Theorem 8 $\mathrm{KL}(D_T \| D_{H'}) \leq \rho + \epsilon$.

Proof. Let R denote the set of nodes x such that $D_T(x) > 0$. By definition,

$$\begin{aligned} \mathrm{KL}(D_T \| D_{H'}) &= \mathop{\mathrm{E}}_{x \in D_T}\left[\lg \frac{D_T(x)}{D_{H'}(x)}\right] \\ &= \sum_{x \in R} D_T(x) \lg \frac{1}{D_{H'}(x)} - \sum_{x \in R} D_T(x) \lg \frac{1}{D_T(x)}. \end{aligned}$$

We first lower bound $\sum_{x \in R} D_T(x) \lg(1/D_T(x))$. We know that for all $x \in R$, $1/(d^l k) \leq D_T(x) \leq \beta^l$. Then,

$$\begin{aligned} \sum_{x \in R} D_T(x) \lg \frac{1}{D_T(x)} &\geq \sum_{x \in R} \frac{1}{d^l k} \lg \frac{1}{\beta^l} \\ &\geq \frac{l}{(\beta d)^l} k \lg \frac{1}{\beta}. \end{aligned}$$

Next, we upper bound

$$\sum_{x \in R} D_T(x) \lg \frac{1}{D_{H'}(x)} = \mathop{\mathrm{E}}_{x \in D_T}\left[\lg \frac{1}{D_{H'}(x)}\right].$$

We do not have immediate access to the value of this summation, but we estimate it using an upper bound on $\mathrm{E}_{x \in X}[\lg(1/D_{H'}(x))]$. Let k be the number of centers of D_T. Then greedy set cover will generate a solution with at most $k \lg m$ centers. Due to the fact that no node has outdegree $> d$, we know that for all $x \in X$, $D_H(x) \geq 1/(d^l k \lg m)$, and $D_{H'}(x) \geq \frac{1-q}{k(\lg m) \cdot d^l} + \frac{q}{N}$. Setting $q = 1/N$,

$$\mathop{\mathrm{E}}_{x \in X}\left[\lg\left(\frac{1}{D_{H'}(x)}\right)\right] \leq \lg\left(\frac{N^2 d^l k \lg m}{N^2 - N + d^l k \lg m}\right).$$

Using this bound and the uniform convergence result of Haussler [5], one can conclude that for large enough X (whose size we will determine later) $\mathrm{E}_{x \in D_T}[\lg D_{H'}(x)]$ is within ϵ of $\mathrm{E}_{x \in X}[\lg D_{H'}(x)]$ with probability at least $(1-\delta)$. Combining with the previous bound, we have $\mathrm{KL}(D_T \| D_{H'}) \leq \rho + \epsilon$.

We now compute the minimum sample set size required to guarantee uniform convergence.

To use the result in [5] one needs to bound $D_{H'}(x)$ both from above and below. Note that $1/N^2 \leq D_{H'}(x) \leq 1$, and thus $2 \lg N \geq \lg(1/D_{H'}(x)) \geq 0$. Using the uniform convergence result of [5], a sample size of

$$m \geq O\left(\frac{k^2 \lg^6 N}{\epsilon^4} + \frac{\lg^2 n}{\epsilon^2} \lg \frac{2}{\delta}\right)$$

suffices. □

We now present a bound on the distance between D_H and D_T.

Corollary 9 $L_1(D_T, D_H) \le 2/N + (2\lg 2)\sqrt{\rho + \epsilon}$.

To show this, we give bounds on $L_1(D_H, D_{H'})$ and $L_1(D_T, D_{H'})$.

Lemma 10 $L_1(D_H, D_{H'}) \le 2/N$.

Proof. $L_1(D_H, D_{H'}) = \sum_x |D_H(x) - D_{H'}(x)| = \sum_x |q(D_H(x) - 1/N)| \le q\sum_x (D_H(x) + 1/N) = 2q$. By our choice of $q = 1/N$, the lemma follows. □

Lemma 11 $L_1(D_T, D_{H'}) \le (2\lg 2)\sqrt{\rho + \epsilon}$.

Proof. It can be shown [1] that for any two distributions D, D', $2\ln 2\sqrt{\mathrm{KL}(D||D')} \le L_1(D, D')$. We plug the bound obtained in Theorem 8 on $\mathrm{KL}(D_T||D_{H'})$ into the equation to get the bound. □

Proof (of Corollary 9)**.** From the triangle inequality of the L_1-metric and the results of Lemmas 10 and 11, the proof follows. □

6 HIDDEN CENTERS VIA LINEAR ALGEBRA

In this section we look at learning concepts from concept class $\mathcal{C}_N^\alpha$, which is of the form $(\{P\}, \mathcal{K}_N^{k,\alpha}, \mathcal{L}_N)$. $\mathcal{K}_N^{k,\alpha}$ is the subset of $\mathcal{K}_N^k$ (defined in Section 3) of distributions on at most k centers, where each center u is output with probability strictly greater than α. $\mathcal{L}_N$ contains only the distribution that outputs l with probability 1. We assume that P is nonsingular and that the bound α is known to the learner.

The set $\mathcal{H}$ of hypothesis distributions is $\mathcal{C}_N^\alpha$. Since the hypothesis class only contains hypotheses where the number of centers is at most k, learning and storing such a distribution in terms of its centers lead to significant compression over storing the probability of outputting each node.

6.1 NONSINGULAR P

Let $\overline{b}$ be a vector representing the target distribution. Our sample set X comes from the target distribution $D_T = \mathrm{rw}(P, L, K)$.

Algorithm. Calculate $\hat{b}$, the observed distribution, where entry i takes the value $1/|X|$ times the number of occurences of i in X. Construct and solve the equation $P^l\hat{x} = \hat{b}$ for $\hat{x}$. Label as centers those nodes whose entries in $\hat{x}$ are greater than ϵ.

Let $\overline{x}$ denote the vector representation of the actual distribution K on the centers. Since $\hat{b}$ is only an approximation to the vector $\overline{b}$, we consider the stability of the system in order to show that $\hat{x}$ is a good approximation to $\overline{x}$. We use theorems from [4] to show the following error bound.

Theorem 12 $||\hat{x} - \overline{x}||/||\overline{x}|| \le \kappa(P) \cdot ||\hat{b} - \overline{b}||/||\overline{b}||$.

Here, $\kappa(A) = ||A^l|| \cdot ||(A^l)^{-1}||$ is called the *condition number* of A. $||A||$ denotes any norm of matrix A. In general we will be interested in the 1-norm, which is defined by

$$||A||_1 = \max_{1\le j\le n} \sum_{i=1}^{m} a_{ij}$$

for an $m \times n$ matrix A.

With this bound, we can state the following theorem about learning via equation solving.

Theorem 13 *The class* $\mathcal{C}_N^{2\epsilon}$ *is* ϵ*-learnable under the* L_1*-norm in time* $\mathrm{poly}(1/\epsilon, 1/\delta, N)$.

Proof. We use the bound on the sensitivity of such a system and look at sufficient samples so that the error in each row of $\hat{b}$ is less than $\epsilon/(\kappa(P^l)N)$. Thus, $||\hat{x} - \overline{x}||$ is bounded by $\epsilon/\kappa(P^l)$. Since $||\overline{x}|| = ||\overline{b}|| = 1$, the total error $||\hat{x} - \overline{x}||$ is upper bounded by ϵ by Theorem 13. In the worst case any one element of $\hat{x}$ can have error ϵ. Thus any center node appears with probability strictly greater than ϵ and any noncenter node appears with probability at most ϵ. Therefore it is not possible to have a node i that has zero probability and a node j that has nonzero probability in $\overline{x}$ such that $\hat{x}_i \ge \hat{x}_j$. Picking the k elements with the highest values returns exactly the k nonzero elements in D_T.

To analyze the run-time, we need to determine how many samples are needed to ensure with high probability that each reachable node will have error at most $\epsilon/(\kappa(P^l)N)$. Using Hoeffding bounds for each entry separately, we conclude that $\mathrm{poly}(1/\epsilon, 1/\delta, N)$ samples are sufficient. □

6.2 VERY LARGE GRAPHS

For very large graphs, we give an algorithm whose runtime and sample complexity is polynomial in the number of samples that can be generated and d^l, and only depends logarithmically on the number of nodes in the graph. Given a graph with probabilities represented by matrix P, and a set X of samples, let $T(P, X, l)$ be the following transformation on P. Remove from P the rows and columns for all nodes except (a) those nodes from which a walk can reach a member of X within l steps, (b) those nodes that are reachable from the nodes mentioned in (a) within l steps. As a result of this "pruning" of the graph, there will be nodes in the new graph whose outgoing probabilities do not add up to 1. For any such node i, add a new node v_i to the graph and add an edge (i, v_i) whose probability makes the outgoing probability of i equal to 1. Let v_i have a self-loop with probability 1.

In this section we consider the graphs with probability matrix P that satisfies the following properties: For any set of k centers in the graph, with high probability over the set X of samples generated, $T(P, X, l)$ is a nonsingular matrix.

A class of graphs that satisfies this property is the graphs that are acyclic with the exception that each node has a self-loop. Such a graph is upper triangular, and this property is preserved by the transformation T.

We now define the notion of the condition number of the result of such a transformation.

Definition 14 *The k-*center condition number *of a matrix P representing a graph and its transition probabilities, denoted $\kappa_{\max}(P,l,k)$, is the maximum over all possible center sets of size k and all possible polynomial size sample sets X for each of these center sets (where the walk is of length l), of the quantity $\kappa(T(P,X,l))$.*

With this information, we have a bound on the condition number of the pruned matrix for any sample set arising from center sets of size k. Therefore, our problem reduces to the case of the previous section, only with a matrix of size $O(kd^{2l})$, where the condition number is at most $\kappa_{\max}(P,l,k)$.

Acknowledgements. We thank Dana Ron and Mike Kearns for illuminating discussions regarding uniform convergence.

References

[1] T. Cover and J. Thomas. *Elements of Information Theory.* John Wiley, 1991.

[2] M. Farach and S. Kannan. Efficient algorithms for inverting evolution. In *Proceedings of the 28th ACM Symposium on Theory of Computing,* pages 230–236, 1996.

[3] A. Golding and D. Roth. Applying winnow to context-sensitive spelling correction. In *Machine Learning: Proceedings of the 13th International Conference,* pages 182–190, 1996.

[4] G. H. Golub and C. F. Van Loan *Matrix Computations.* The Johns Hopkins University Press, 1991.

[5] D. Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. In *Information and Computation,* 100:78–150, 1992.

[6] M. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R. Schapire, and L. Sellie. On the learnability of discrete distributions. In *Proceedings of the 26th ACM Symposium on Theory of Computing,* pages 273–282, 1994.