Electricity Based External Similarity of Categorical Attributes

Christopher R. Palmer[1] and Christos Faloutsos[2]

[1] Vivisimo, Inc.
2435 Beechwood Blvd, Pittsburgh, PA
palmer@vivisimo.com
[2] Computer Science Department, Carnegie Mellon University
5000 Forbes Ave, Pittsburgh, PA
christos@cs.cmu.edu

Abstract. Data mining tools use similarity or distance computations as a fundamental and critical data property. Categorical attributes abound in databases. For example, the *Car Make*, *Gender*, *Occupation*, etc. fields in a car insurance database contain a great deal of useful information that is encoded as categorical values. Sadly, categorical data is not easily amenable to similarity computations. Typically, a domain expert could manually specify some or all of the similarity relationships. This is error-prone and not feasible for attributes that take on many values, nor is it useful for cross-attribute similarities, such as between *Gender* and *Occupation*. *External similarity* functions define a similarity between, say, *Car Makes* by looking at how they co-occur with the other categorical attributes. In this paper we exploit a rich duality between random walks on graphs and electrical circuits to develop an external similarity function called REP. The only previously proposed external similarity function is ad-hoc while REP is theoretically grounded. To illustrate the usefulness of REP, we conduct two experiments. First, we cluster categorical attribute values to show the relationships inferred by REP. Second, we use REP effectively as a nearest neighbour classifier.

1 Introduction

Similarity between objects is a fundamental property required for data mining applications. For example, clustering is a basic data mining tool used to understand data. Standard clustering algorithms use the similarity between objects to group them into clusters. Common visualization tools map a similarity matrix into 2 or 3 dimensional points, allowing the data set to be displayed on a terminal. Similarity is necessary or useful for many other applications.

There is a great deal of categorical data stored in databases but it is difficult to define similarity between categorical values. For example, an car insurance database may contain a field with different automobile manufacturers such as *Toyota*, *Hyundai* and *Porsche*. One solution to the lack of similarity in this field is to manually specify the 3x3 similarity table for these automobiles. However, such an approach is error-prone and becomes nearly impossible when we have a more realistic number of car manufacturers. Little else could be done if the car information appears in isolation. However, there is other information in the database such as *Age*, *Gender*, *Occupation*, and *Number of Accidents*. The goal of this research is effectively use these additional attributes to infer a superior similarity function over the automobile manufacturer categorical values. Such a similarity function is known as an *External Similarity* function [3].

Our proposed similarity function is based on random walks in graphs (equivalently, currents in circuits). We treat categorical data in this way because it naturally offers a way to define "recursive" similarities. To motivate the need for this recursiveness, consider the toy shopping basket in Figures 1 and 2. In these tables, the first customer bought *Bud*, *Diapers*, and *Chips* while the second bought *Coors*, *Diapers*, and *Cheesies*, and so on. The problem is to measure the similarity between the three beverages (*Bud*, *Coors*, and *Milk*) using the other items in the shopping baskets (*Diapers*, *Chips*, *Cheesies*, and *Cookies*).

bud	coors	milk	diapers	chips	cheesies	cookies
1			1	1		
	1		1		1	
		1	1			1

Fig. 1. First example shopping basket data

bud	coors	milk	diapers	chips	cheesies	cookies
1			1	1		
	1		1		1	
		1	1			1
					1	1

Fig. 2. Second example shopping basket data with 3rd customer

In the first table, *Bud*, *Coors*, and *Milk* should all be equally similar. Each is purchased with *Diapers* and that is the only information given. The table in Figure 2 is identical except for a fourth customer who purchased *Chips* and *Cheesies*. This purchase provides a link between the *Chips* purchased with the *Bud* and the *Cheesies* that were purchased with the *Coors* and we would like the similarity function to take into account these secondary (recursive) similarities.

To compute the similarity between *Bud* and *Coors*, we could run the following random process. Initially begin with *Bud*. Randomly pick an item that co-occurs with *Bud* in the shopping basket table. Repeat with the newly selected item. Then *Bud* and *Coors* are similar if the expected number of steps between *Bud* and *Coors* is small. In section 4 we will explore several options based on random walks of this type to help motivate the random walk based similarity function defined in section 5.

The main contribution of this paper is the development of REP, a similarity/distance function with the following properties:

1. *REP* is theoretically grounded.
2. *REP* allows comparisons between different values of the same attribute and allows comparisons between values from different attributes. For example, we can compare car types in an insurance data base but we can equally well compare a car types and an occupation.
3. *REP* may be computed efficiently using a relaxation algorithm and scales very well with data base size.
4. *REP* shows improved performance on real tasks using real data.

The remainder of this paper is organized as follows. In the following section, we describe some related work. Section 3, provides background material relating random walks to electrical circuits and explains the construction of a graph (electrical circuit), given a table of categorical data. Section 4 presents three potential, but flawed similarity/distance functions based on random walks (electrical current). Section 5 presents *REP* which corrects these flaws. Section 6 demonstrates experimentally the validity of our approach and examines the scalability of *REP*.

2 Related Work

The specific problem of clustering categorical data has been studied extensively recently [5–7] but this differs from our current study because it is based on clustering tuples that contain categorical values rather than in developing a more generally useful similarity function.

Jeh and Widow proposed a distance function for graphs using random walks [8]. To measure the distance between nodes u and v, place a random web surfer at each of these vertices. In lock-step, the surfers randomly walk the graph and we measure the expected distance until they meet at a common node. Their method has some surprising properties. All nodes in a cycle are infinitely far apart (because random walkers walking in lock step will never meet). Such a property may be unfortunate. Moreover, it is expensive to compute. Given a graph G with n nodes, the computation is based on G^2, a graph with n^2 nodes. This makes it impractical for our work.

Das and Manila proposed the only prior work on external similarity [3]. Given a probe set P and attributes A and B, define the distance between A and B as:

$$D_{fr,P}(A,B) = \sum_{D \in P} |fr(A,D) - fr(B,D)|$$

where $fr(x,y)$ is the fraction of rows containing both x and y in the data base. Two comments are in order at this point. First, without loss of generality, we can remove the probe set. The probe set can be viewed as a projection of the attributes and in this work we use all attributes (other than A and B) as the probe set. Second, this function

does not define distances recursively. For the toy example in the introduction *Milk*, *Bud*, and coors are all equally far apart.

Finally, a Klein and Randic proposed one of the alternatives that we consider in section 4 for the problem of defining the similarity between molecules [9]. They proposed a *resistance distance* which is simply the reciprocal of the *current similarity*. When we evaluate this current similarity, we will see that it is not appropriate for our requirements.

3 Background and Definitions

After providing the required notion and definitions, we present a pair of theorems which are part of the strong relationship between electrical circuits and random walks on graph. The use of electrical circuits helps provide intuition into some of the proposed similarity functions and leads to an efficient implementation of REP. To actually address the issue of categorical attributes, at the end of this section we present two constructions that convert a table of categorical data into a graph.

3.1 Definitions

Categorical A table of categorical values has n rows and m columns. The total number of (column) distinct attributes values is M. In Figure 2, there are $n = 3$ rows, $m = 7$ columns and $M = 14$ distinct attributes (each column takes on a value of 0 or 1 but we count all 14 possibilities, not just the 2 distinct attributes!).

Electricity In a circuit the *current* (I), *voltage* (V) and *resistance* (R) are related by the equation $I = V/R$. Conductance, C is the reciprocal of resistance ($C = 1/R$). Voltage is measured as a decrease in electric potential between pairs of points. We say *voltage at* x whenever the second point is obviously inferred (such as the ground). At any point in an electrical circuit the total current entering this point is the same as the total current leaving this point (the *Kirchoff's law of conservation of current*).

Walks - Let G be an edge weighted undirected graph with vertices V, edges E and edge weights $w : V \times V \to \mathcal{R}$. A (u, v) walk is a sequence of vertices beginning with u and ending with v,

$$(u = x_0 \to x1 \to \cdots \to x_n = v)$$

such that (x_{i-1}, x_i) is an edge in E. We write $u \to^* v$ to refer to a (u, v) walk. We also define $u \to^+ v$ to be a non-trivial (u, v) walk. That is, one in which $u \neq v$ and thus must include at least one step ($n > 0$).

Constrained walks $u \to^* v/S$, where S is a set of vertices from V are (u, v) walks in which none of the intermediate nodes belong to S. That is, a walk $u \to^* v/\{u, v\}$ is a walk which starts from u and then never returns to u before it reaches v (for the first time). Notionally, let $u \to^* v/x_1, x_2, \cdots, x_k$ be equivalent to $u \to^* v/\{x_1, x_2, \cdots, x_k\}$.

Random walks on G are defined by the edge weightings. Let C_u be the total of all edge weights incident with u, then the probability of moving from u to v is $w(u, v)/C_u$. That is, the probability of following each edge leaving u is simply proportional to its edge weight. Define $P(u \to^* v/S)$ as the probability that a random walk beginning from u satisfies the condition $u \to^* v/S$.

Commute distance between nodes u and v in G is the expected length of a tour that begins at u and passes through v prior to returning to u.

3.2 Electricity vs. Walks

Electricity and random walks on graphs are intimately related. Doyle and Snell provide an excellent introduction to this synergy in [4]. We present the basic information required for this work. The circuit corresponding to a graph (and visa-versa) is defined by replacing each edge (with weight $w(u, v)$) with a resister with resistance $R(u, v) = 1/w(x, y)$ (or $C(u, v) = w(x, v)$). Given the pairing of a graph and a circuit we present two theorems relating electrical properties and random walks.

Theorem 1. *Let Z be a circuit with a battery attached to u (+1 volt) and v (ground). Here Z is assumed to be a single connected circuit. Let C be the total of the conductance of all resisters in Z. Let $I(u,v)$ be the current flowing from u to v. Then, $2C/I(u,v)$ is the commute distance between u and v.*

Proof. The unweighted version of this proof is in [2], theorem 2.1. The weighted version follows naturally from that proof.

□

Theorem 2. *Let C be a circuit with a battery attached to u (+1 volt) and grounded to each element in S. Assume that $u \notin S$. Let G be the graph corresponding to C. Let E be the voltage drop at x. Then $E = P(x \rightarrow^* u/S)$ (in G).*

Proof. (sketch) - Let $E(x)$ be the voltage drop at x and then, for an electrical circuit, the law of conservation of current requires that at any node, x (other than u where $E(u) = 1$ and $s \in S$ where $E(s) = 0$):

$$\begin{aligned} \textstyle\sum_{y \sim x} \frac{E(y) - E(x)}{R(x,y)} &= 0 \\ \Rightarrow \textstyle\sum_{y \sim x} \frac{E(y)}{R(x,y)} &= E(x) \cdot 1/R(x) \\ \Rightarrow \textstyle\sum_{y \sim x} \frac{E(y) \cdot R(x)}{R(x,y)} &= E(x) \end{aligned}$$

For the random walks in G, we can write the function $f(x) = P(x \rightarrow^* u/S)$ as

$$\begin{cases} P(x \rightarrow^* u/S) = 1 & \text{if } x = u \\ P(x \rightarrow^* u/S) = 0 & \text{if } x \in S \\ P(x \rightarrow^* u/S) = \sum_{y \sim x} P(x \rightarrow y) \cdot P(y \rightarrow^* u/S) & \text{if } x \neq u, x \notin S \end{cases}$$

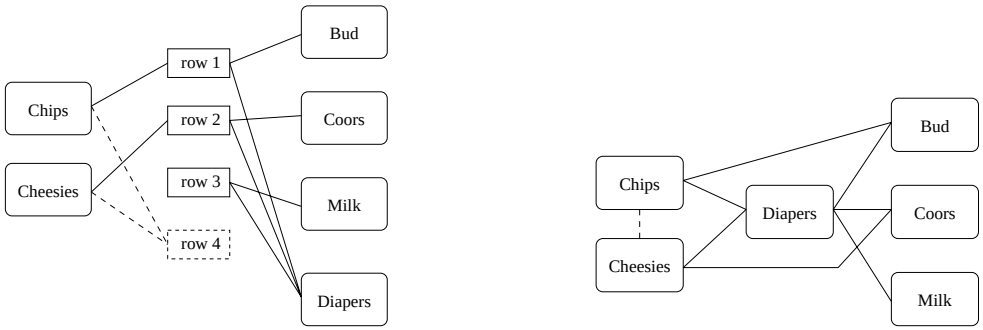
Both $f(x)$ and $E(x)$ are harmonic functions. They share boundary values and the uniqueness theorem of harmonic functions states $f(x) = E(x)$ which completes the proof.

□

Escape probability is similar to $P(x \rightarrow^* u/S)$, but is defined as the probability that a walk started from x will reach S before it returns to x.

3.3 Treating Categorical Data as a Graph

Our goal is to define a similarity function for categorical data that recursively uses co-occurrence information from the other attributes. To do so, we propose two methods for converting a table of categorical values into an edge weighted graph. The first method preserves the tuples while the second method produces a more compact graph.



(a) Tuple Attribute Graph (b) Attribute-Attribute graph

Fig. 3. Two graph representations for the table 2

Tuple-Attribute Graph Given the table of categorical data in figure 1 (where non-existent values are treated as NULLs), we will explain how to generate the *tuple-attribute* graph shown in Figure 3(a). Create a node, r_i, for each of the n rows. Create a node a_j for each of the M distinct attribute values. Place an edge (with weight 1) between r_i and a_j if and only if the attribute value j appears in row i. This construction results in a bipartite graph.

Attribute-Attribute Graph Given the table of categorical data in figure 1, we will explain how to generate the *attribute-attribute* graph shown in Figure 3(b). This time, create a node a_j for each of the M distinct attribute values. The edge set is implied by the weight function

$$w(a_i, a_j) = fr(a_i, a_j) = \text{ number of tuples containing both } a_i \text{ and } a_j$$

There is obviously a strong relationship between these two graphs. In this paper we concentrate on the smaller attribute-attribute graph. The use of the *Tuple-Attribute graph* is an area that we are exploring.

4 Electric Similarity Functions

Given the attribute-attribute graph corresponding to a table, there are several intuitive similarity functions to consider. We see that they have flaws and we use these flaws to motivate our proposed similarity measure, REP.

4.1 Flawed similarity/distance functions

Electrical Current Similarity Define $I(x, y)$ to be the current flowing between x and y in the electrical circuit where we have attached a 1 volt battery across x and y. This is a natural function with two excellent properties best expressed as random walks:

1. x and y are more similar if they are connected by shorter walks.
2. x and y are more similar if they are connected by more walks.

Unfortunately, current also has serious scale issues. Take for example the simple table in Figure 4a). Here x and y occur with a every time they appear. Also, p and q appear with b every time they appear. We expect the similarity between x and y to be the same as the similarity between p and q because their appearances of equivalent, except for scale. Unfortunately, $I(x, y) = 2 \cdot I(p, q)$ ($I(x, y) = 0.75$ for the attribute-attribute graph and $I(x, y) = 0.5$ for the tuple-attribute graph). This is a serious practical issue, causing pairs of very frequent attributes to be very similar.

X or Y	A or B
x	a
x	a
y	a
y	a
p	b
q	b

a) Electrical current similarity

X or Y	Z
x	z1
x	z2
x	z3
x	z0
y	z0
y	z4
y	z5
y	z6

b) Escape probability

Fig. 4. Examples that show poor behaviour for the proposed similarity/distance functions

Commute Distance To correct the scale problem, we considered the commute distance (expected length of a commute starting from x, reaching y and then returning to x). By theorem 1, the commute distance is simply $2C/I(x, y)$ which normalizes the distances and corrects the first example (C is the total conductance of all resisters in the circuit). Commute distance has a subtle flaw. For data that is relatively uniform (the out degree distribution of the graph does not follow a skewed distribution), the commute distance may indeed appear useful. However, for realistic data where the degree distribution follows a Zipf or power-law relationship, the commute distance degenerates. High degree nodes will have a much higher stationary probability (probability that a random walk will be at the high degree node at any given time) and consequently all the distances are skewed toward the largest nodes. This was discovered when we began the experiments discussed in section 6. When clustering attribute values, the highest degree node was invariably the focal point of the clustering and distances were not particularly useful.

Escape Probability An attempt to correct the problem with the commute distance used the *escape probability*. The escape probability is the probability of a non-trivial walk starting at x will return to x before reaching y. This has a natural definition in terms of circuits. Place a +1 volt battery at x and grounded at y, and then measure the effective conductance, C, between x and y ($C = I(x, y)$ since we have a 1 volt drop) and let $C(x)$ be the conductance of x. Then,

$$Pesc(x, y) = I(x, y)/C(x) = I(x, y) \cdot R(x)$$

(for a proof see [4], page 42). In the case of our attribute graphs, $C(x)$ is the number of rows containing x. It is this normalization based on the number of rows which made the escape probability a candidate similarity metric. To make it symmetric, we simply define

$$S_{esc}(x, y) = \frac{Pesc(x, y) + Pesc(y, x)}{2}$$

S_{esc} is also flawed because it does not account for the length of a (x, y) walk, only its existence. For example, see Figure 4b). Here there is a direct relationship between x and y and only an indirect relationship (through x and y!) between $z1$ and $z6$. However, the escape probability similarity assigns

$$S_{esc}(x, y) = .125 \text{ and } S_{esc}(z1, z6) = .25$$

which is a very poor similarity function indeed!

4.2 Proposal: Refined Escape Probability (REP)

We now combine the positive points of each method discussed above to define our proposed similarity function, the *Refined Escape Probability* (REP) similarity. Commute distance was appealing because it accounts for the length of (x, y) walks. Escape probability normalized according to frequency of an attribute but became walk-length agnostic. To correct this problem with the escape probability, we will use the concept of a *sink* node, s, that we attach to all nodes in the network. We assign resistances such that the probability of stepping from any node to the sink is $sink\_p$. We then can measure the probability of (x, y) walks that do not pass through either x, y or s. That is, given G attach the sink node as described and then define

$$S_{REP}(x, y) = R(y) \cdot P(x \rightarrow^* y/x, y, \text{ or } s)$$

In the next section we convert this to two electrical circuit computations and show that it is symmetric (which explains the use of $R(y)$ as a normalization factor).

5 *REP* Algorithm

Given the high level description of REP, we now turn out attention to making the procedure concrete and considering the efficiency of implementation. Recall that the similarity between x and y is the probability of a walk from x to y that does not pass through x, y or some new "sink" node. The sink node provides a bias toward short walks and we apply a normalization factor to make this symmetric. In this section, we will complete the algorithm by:

1. Explaining the addition of the *sink* nodes to the graph.
2. Showing that S_{REP} is symmetric.
3. Converting the probability definition of S_{REP} into a definition based on electrical circuits.
4. Using a relaxation algorithm based on Kirchhoff's laws to solve these circuits.

5.1 Adding *sink* Nodes to a Graph

From a table of categorical attributes (or some other source), we have a graph G_0. The problem is to construct a graph, G, from G_0 by adding a new sink node, s, and adding an edge from every node, x, of G_0 to s such that the probability of a random step from x to s is some constant, $sink\_p$. To do this, we note that $P(x \to s) = w(x,s)/C_G(x)$ where $C_G(x)$ is the total conductance of x in the graph G and $w(x,s)$ is the weight of the edge. Now, since $C_G(x) = C_{G_0}(x) + w(x,s)$ we just solve the equation

$$\begin{aligned} \frac{w(x,s)}{C_{G_0}(x)+w(x,s)} &= p \\ C_{G_0}(x) + w(x,s) &= w(x,s)/p \\ w(x,s) \cdot (1/p - 1) &= C_{G_0}(x) \\ w(x,s) &= C_{G_0}(x)\frac{1}{1/p-1} \end{aligned}$$

And thus it is easy to add the required edges to the graph.

5.2 S_{REP} is Symmetric

Theorem 3. *Let* $S_{REP}(x,y) = R_y \cdot P(x \to^* y/x,y,s)$ *then* $S_{REP}(x,y) = S_{REP}(y,x)$.

Proof. Let $W = (x = u_0 \to u_1 \to \cdots \to u_k = y)$ be any (x,y) walk where $u_i \notin \{x,y,s\}$ for $1 \le i < k$. From W we can also define the walk $\bar{W} = (y = u_k \to u_{k-1} \to \cdots \to u_0 = x)$. There is a one-to-one correspondence between the (x,y) walks and the (y,x) walks. Thus to prove the result, we must show that $R_y \cdot P(W) = R_x \cdot P(\bar{W})$. Write:

$$R_y \cdot P(W) = R_y \cdot P(u_0 \to u_1) \cdot P(u_0 \to u_1) \cdots P(u_{i-1} \to u_k)$$

Let $C_{u_i} = 1/R_{u_1}$ be the total conductance of node u_i and recall that $P(u_i \to u_{i+1}) = w(u,u_{i+1})/C_{u_i}$ and then we can express $P(W)$ as

$$R_y \cdot P(W) = P(W) \cdot R_y = \frac{w(x,u_1)}{C_x} \cdot \frac{w(u_1,u_2)}{C_{u_1}} \cdots \frac{w(u_{k-2},u_{k-1})}{C_{u_{k-2}}} \cdot \frac{w(u_{k-1},y)}{C_{u_{k-1}}} \cdot \frac{1}{C_y}$$

which we can rewrite by simple "shifting" the denominators to the left

$$R_y \cdot P(W) = \frac{1}{C_x} \cdot \frac{w(x,u_1)}{C_{u_1}} \cdot \frac{w(u_1,u_2)}{C_{u_2}} \cdots \frac{w(u_{k-2},u_{k-1})}{C_{u_{k-1}}} \cdot \frac{w(u_{k-1},y)}{C_y}$$

and then $w(u_i,u_{i+1}) = w(u_{i+1},u_i)$ by definition and reordering gives the required result

$$R_y \cdot P(W) = \frac{1}{C_x} \cdot \frac{w(u_1,x)}{C_{u_1}} \cdot \frac{w(u_2,u_1)}{C_{u_2}} \cdots \frac{w(u_{k-1},u_{k-2})}{C_{u_{k-1}}} \cdot \frac{w(y,u_{k-1})}{C_y} = R_x \cdot P(\bar{W})$$

□

5.3 S_{REP} as Electrical Currents

We proposed a similarity function based on the quantity

$$P(x \to^* y/x,y,s)$$

we can use the relationships between random walks and electrical current to compute this quantity. We can compute $P(x \to^* y/y,s)$ by using theorem 2 (place a battery across y and s and measure the voltage drop at x). This is not the required probability because it also includes walks that pass through x one or more times before reaching y. Thus, we can separate the x to x loops from the required walk and write:

$$\begin{aligned} P(x \to^* y/y,s) &= \left(\sum_{i=0}^{\infty} P(x \to^* x/x,y,s)^i\right) \cdot P(x \to^* y/x,y,s) \\ \Rightarrow P(x \to^* y/x,y,s) &= (1 - P(x \to^* x/x,y,s)) \cdot P(x \to^* y/y,s) \end{aligned}$$

Let $E_{y,s}(x)$ be the voltage drop at x with the battery across y and s. Let $E_{y:x,s}(u)$ be the voltage drop at u when the battery is attached to y and grounded at x, s. Then, we can define the probability of a loop from x to x as by taking one step and using theorem 2 again:

$$P(y \rightarrow^* y/x, y, s) = \sum_{u \sim y} P(y \rightarrow u) \cdot P(u \rightarrow^* y/x, y, s)$$
$$P(y \rightarrow^* y/x, y, s) = \sum_{u \sim y} \frac{w(y,u)}{C(y)} \cdot E_{y:x,s}[u]$$

That is, we can define

$$S_{REP}(x, y) = E_{y:s}[x] \cdot \sum_{u \sim y} \frac{w(y,u)}{C(y)} \cdot E_{y:x,s}[u]$$

and the only non-trivial computation that we explain in the next section is $E_{S_1:S_2}[u]$.

5.4 Kirchhoff Relaxation Algorithm for Voltages

We need to compute $E_{S_1:S_2}$ which is the set of voltage drops between all nodes in the circuit when the nodes in set S_1 are fixed at 1 volt and the nodes in S_2 are fixed at 0 volts (ground). We assume that S_1 and S_2 are disjoint. For brevity of notation, let $V_i[u]$ be the i^{th} approximation to $E_{S_1:S_2}[u]$. We initialize the relaxation algorithm by setting

$$V_0[u] = 1 \text{ iff } u \in S_1$$

and then at each step of the relaxation algorithm we update

$$\begin{cases} V_i[u] = 1 & \text{if } u \in S_1 \\ V_i[u] = 0 & \text{if } u \in S_2 \\ V_i[u] = \sum_{v \sim u} \frac{w(u,v)}{C(u)} \cdot V_0[v] & \text{otherwise} \end{cases}$$

where the last sub-equation is simply the basic identify $I = V/R$ subject to Kirchhoff's law (conservation of current):

$$\begin{aligned} \sum_{v \sim u} \frac{V[u]-V[v]}{R(u,v)} &= 0 \\ \Rightarrow (V[u] \cdot \sum_{v \sim u} 1/R(u,v)) - \sum_{v \sim u} \frac{V[v]}{R(u,v)} &= 0 \\ \Rightarrow V[u] \cdot C[u] &= \sum_{v \sim u} \frac{V[v]}{R(u,v)} \\ \Rightarrow V[u] &= \sum_{v \sim u} \frac{C(u,v)}{C(u)} \cdot V[v] \end{aligned}$$

It is easy to verify that these approximation follow a monotone increasing relationship $0 \leq V_0[u] < \cdots < V_i[u] < \cdots < V_l[u] = E_{S_1:S_2}[u]$ (exercise 1.2.5 of [4]) and then converge to the true value. We truncate the approximation sequence when a suitably accurate result is found.

5.5 Running Time of *REP*

There are two phases to our algorithm. First, a graph is constructed from a table of categorical values. This requires the addition of $O(n \cdot m^2)$ edges for a table with n rows and m columns. Assuming reasonable hashing, this can be done in $O(n \cdot m^2)$ time.

Computing S_{REP} requires that we use the relaxation algorithm described in the previous section. In that algorithm, the running time to compute V_0 is $O(n)$ where n is the number of nodes. Each improvement step computes V_i from V_{i-1} and requires $O(\#$ edges) time. Since a graph from a categorical table has one node for each distinct attribute value (and the sink) it has $M + 1$ nodes. The worst case bound on the number of edges is $O(M^2)$, which is rarely reached in practice. Thus, the time to compute S_{REP} is the time to do two calls to the relaxation algorithm and is thus $O(M^2)$ in the worst case.

It is very interesting is that the time to compute the similarities is independent of the number of rows in the table and that the pre-computation grows linearly with the number of rows in the input table. We will explore this in the next section.

6 Experiments

Since the $D_{fr,P}$ algorithm represents the "state of the art" in computing distances for categorical data, the following experiments attempt to compare REP to the $D_{fr,P}$ algorithm. To do so, we define a distance function:

$$d_{REP}(x, y) = 1/S_{REP}(x, y)$$

We use clustering as a means of visualizing the similarity function to see that REP produces superior distances over three data sets. Define the distance of a vector of categorical values as:

$$d_{REP}(< x_1, x_2, \cdots, x_k >, y) = || < d_{REP}(x_1, y), d_{REP}(x_2, y), \cdots, d_{REP}(x_k, y) > ||$$

We can use this distance function to do nearest neighbour classification on five data sets to see that REP offers excellent classification.

6.1 Clustering

The purpose of the clustering experiments is to visualize the distance function over different attributes and different data sets. We will find that REP provides similarity functions that match our expectations better than the distance functions computed by $D_{fr,P}$. Since $D_{fr,P}$ has been previously evaluated using single link hierarchical clustering, that is the algorithm that we will use here [3]. The three data sets we evaluated are:

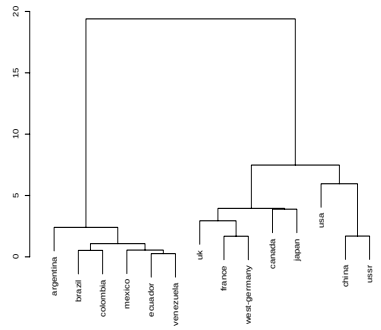
1. **Adult** - a selection of fields from the 1994 census data collected in the United States. There are 32,561 training examples and 16,281 test examples with 6 numeric fields and 8 categorical fields. In the experiments that follow, we treat the numeric fields as categorical fields (that is, each value is a category). We also performed experiments using only the 8 categorical fields which were similar to those reported here. This data set is available from the UC Irvine repository [1].
2. **Autos** - (imports-85) a data base of automobile specifications and insurance risk measures. There are 205 instances with 16 numeric fields and 10 categorical fields. Most of the numeric fields are actually drawn from a small ranges and it is appropriate to treat them as categorical fields. This data set is available from the UC Irvine repository [1].
3. **Reuters** - we extracted the subject keywords from the standard *Reuters-21578* text collection and used each keyword as a binary attribute. There are 19,716 instances with 445 boolean fields. This is the same data used in [3].

Figure 5 shows four different clusterings. The left column is always REP and the right column is always $D_{fr,P}$. Overall, REP appears to match our understand of the data better than $D_{fr,P}$. In particular, in parts (a) and (b), we see that REP creates clusters for the Latin American countries which are not well represented by $D_{fr,P}$. In parts (c) and (d) where we have clustered by maximum level of education attained, our REP results are basically perfect (represents the real hierarchy of education levels) while $D_{fr,P}$ is reasonably good but failed to identify the post-high school degree vs. high school degree split seen in part (c). In parts (e) and (f) where we have clustered by a person's occupation type, we see that REP creates three clusters: manual labour, lower level occupations and senior occupations. Conversely, $D_{fr,P}$ has left *Private house servant* as an outlier, combined *Clerical* with *Other service* and combined *Sales* and *Technical support*. The final pair of clusterings in parts (g) and (h) show the makes of cars in the Auto data set. The comparison here is more subtle, but the REP clustering has a more natural looking structure and three very distinct clusters for the luxury cars, the family cars and the imports. $D_{fr,P}$ on the other hand has combined *Mercury* with the *Alfa Romeo* and the *Porsche* which is somewhat surprising.

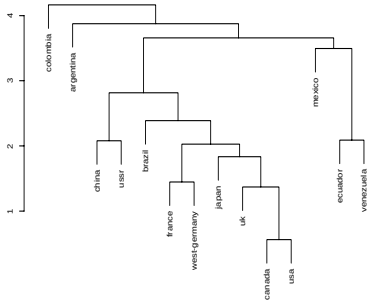
Overall, we see that even with very different data sets and different choices of the attribute on which to cluster, the REP distance function appears both more natural and more "correct" than the $D_{fr,P}$ distance function.

6.2 Classification

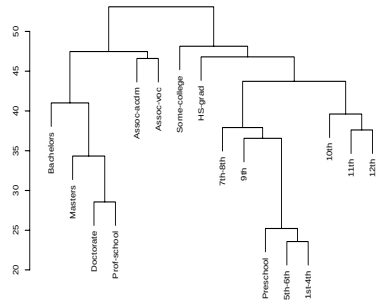
Now that we see that REP appears to be producing good distance measures. Now, we quantitatively evaluate its performance using the vector definition of our distance function to do nearest neighbour classification. This nearest neighbour classification relies on a good distance function. For comparison, we use C4.5 [10] which provides an



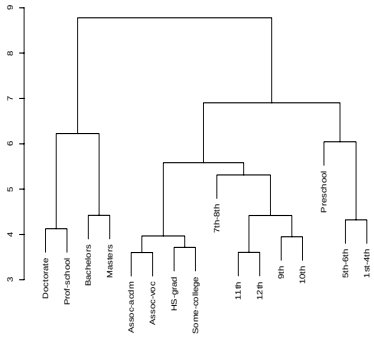
(a) REP clustering of Reuters countries



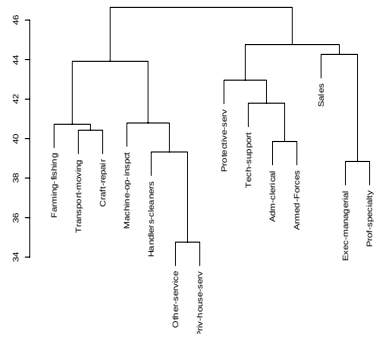
(b) $D_{fr,P}$ clustering of Reuters countries



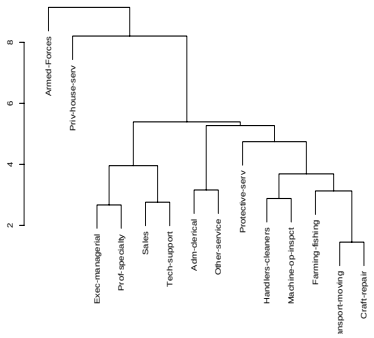
(c) REP clustering of Adult maximum education



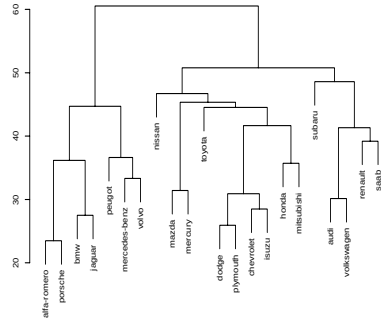
(d) $D_{fr,P}$ clustering of Adult maximum education



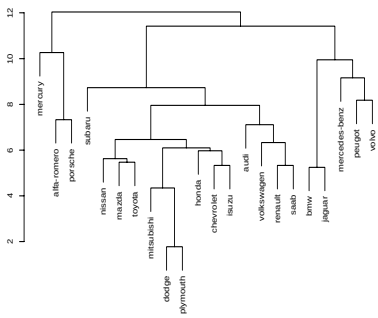
(e) REP clustering of Adult occupation type



(f) $D_{fr,P}$ clustering of Adult occupation type



(g) REP clustering of Autos car makes



(h) $D_{fr,P}$ clustering of Autos car makes

Fig. 5. Comparison of REP and $D_{fr,P}$ by using clustered output

Data set name	*REP*	C4.5	NN-hamming
Adult	15.1 %	13.6 %	21.5%
Audiology	23.1 %	15.4 %	30.8%
Letter recognition	28.2 %	32.4 %	30.5%
Mushrooms	0.8 %	0.2 %	0%
Optical recognition of handwritten digits	12.2 %	43.2 %	14.4 %

Table 1. Error rates show *REP* has classification error similar to C4.5 and better than NN with hamming distance

excellent benchmark of quality. For further comparison, we also ran a NN algorithm using the hamming distance between instances. We hope to see performance that is similar to C4.5 algorithm and is better than the simple hamming distance function. We ran this classification task for 5 data sets available from the UCI collection [1]. These results are summarized in Table 1 where we record the percent error for each method.

REP offers performance similar to C4.5. It has lower error for two data sets, nearly identical error for two data sets and higher error for only one data set. Using nearest neighbour classification with hamming distance is actually surprisingly good on some of the simpler tasks but is quite poor for the adult data set.

Thus we have seen experimentally that our distance function, *REP* both provides distance matrices that appear reasonable based on our background knowledge and that provide reasonable classification accuracy when used for NN classification.

6.3 Sensitivity to *sink_p*

Our distance function has one parameter, *sink_p*, which is the transition probability from each node to the terminal sink node. The larger the value of *sink_p*, the less value will be assigned to longer paths. In Figure 6, we ran the classification task for many values of *sink_p* to test the sensitivity of our results.

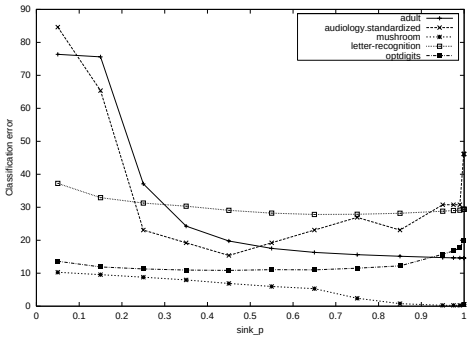


Fig. 6. Classification results are not very sensitive to *sink_p*

Here we see that the results can be completely useless for very small values of *sink_p*. For very small values of *sink_p*, the similarity between two values is essentially the number of walks between these two values. This gives high similarity to common values, independent of their distribution (because there are just more walks involving them). Most of the data sets exhibit increasing performance as *sink_p* grows from 0.5. The letter and digit recognition tasks are the exceptions which perform worse for very large values of *sink_p*.

Overall the results are quite stable for many different values of *sink_p*. We used 0.85 as a default value of the *sink_p* parameter which appears to be a good general purpose choice.

6.4 Scalability

The time to compute distances using *REP* has both a preprocessing component and a per-distance component. To realistically measure the running time, we chose to measure the time required to compute the distances to cluster the maximum education attained for the Adult data set. To vary the input size, we used the first x tuples of the

combined training and test sets. In Figure 7 we report the preprocessing time and the average time to compute each of the 16×7 required distances in the education distance array. These times are averaged over 3 runs.

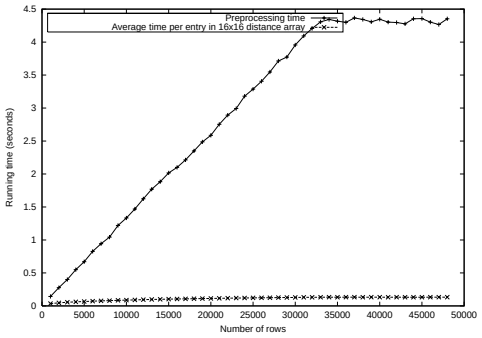


Fig. 7. Running time scales at most linearly with data set size

Our analysis claimed that the preprocessing time is linear in the number of edges in the graph and in terms of the number of input rows. We see this behaviour in our experiment and further see that it is the cost of the edges that are dominating in our experiment. The time becomes almost constant at the point where all pairs of attributes that will appear in the same tuple have appeared in the same time. The average time to compute a distance array element varies from .03 seconds to .13 and scales excellently with the input size!

7 Conclusion

In this paper we presented a node similarity function for graphs. We used this node similarity function to provide an external similarity function called S_{REP}. We found that this function is:

- better than the best existing external distance function,
- built upon a theoretical foundation (the existing approach is not), and
- allows cross attribute similarity computations which allows
- excellent nearest neighbour classification.

Our implementation was evaluated and we found that it

- provides excellent scalability with input size, and
- is not sensitive to its only parameter.

References

1. C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.
2. A. K. Chandra, P. Raghavan, W. L. Ruzzo, and R. Smolensky. The electrical resistance of a graph captures its commute and cover times. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 574–586, 1989.
3. Gautam Das, Heikki Mannila, and Pirjo Ronkainen. Similarity of attributes by external probes. In *Knowledge Discovery and Data Mining*, pages 23–29, 1998.
4. Peter G. Doyle and J. Laurie Snell. Random Walks and Electric Networks.
5. Venkatesh Ganti, Johannes Gehrke, and Raghu Ramakrishnan. CACTUS - clustering categorical data using summaries. In *Knowledge Discovery and Data Mining*, pages 73–83, 1999.
6. David Gibson, Jon M. Kleinberg, and Prabhakar Raghavan. Clustering categorical data: An approach based on dynamical systems. *VLDB Journal: Very Large Data Bases*, 8(3–4):222–236, 2000.
7. Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. ROCK - a robust clusering algorith for categorical attributes. In *Proceedings of IEEE International Conference on Data Engineering*, 1999.
8. G. Jeh and J. Widom. Simrank: A measure of structural-context similarity, 2002.
9. D. J. Klein and M. Randic. Resistance distance. *Journal of Mathematical Chemistry*, 1993.
10. Ross Quinlan. C4.5 decision tree generator.