

Lecture 9 - Computing Harmonic Functions: February 21, 2008

*Lecturer: Dragomir Radev**Scribe: Nandini Bhardwaj*

1 Computing Harmonic Functions

1.1 Introduction

In this lecture, we discuss a few ways of computing harmonic functions, notably the Method of Relaxations and Monte Carlo methods. Computing a harmonic function in one dimension is quite simple but harmonic functions in arbitrary dimensions require more sophisticated methods. Figure 1 shows an example of a harmonic function in two dimensions.

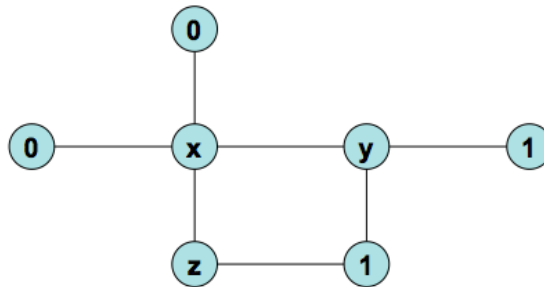


Figure 1: 2 D Harmonic Function

1.1.1 First attempt

We have 2 negative examples and 2 positive examples and 3 unlabeled examples. We assume that each value is only dependent on its nearest known values. This is similar to the k-nearest neighbour approach in Machine Learning. Since the two known nearest neighbors of x are labelled 0, we assign a 0 to x. Similarly, we assign 1 to y. The only known neighbor of z is 1. Therefore, we assign a value of 1 to z also. This goes further into the area of semi-supervised learning of harmonic functions on graphs.

1.1.2 The Dirichlet Problem

In this problem, we are given a sheet of metal. A square portion has been off from the center of this metal sheet. The problem is to formulate an expression for the distribution of temperature at all points on the metal sheet under the condition that the outer edge is heated uniformly - each point on the outer edge is held at $T=1$ and each point on the inner edge is held at $T=0$ (see Figure 2).

This problem requires the use of second order gradients, and Laplace's differential equation in particular, for its solution.

$$\nabla^2 u = u_{xx} + u_{yy} = 0$$

This equation is a special (steady-state) case of the (transient) heat equation given by

$$k\nabla^2 u = u_t$$

In general, the solutions to this equation are called harmonic functions.

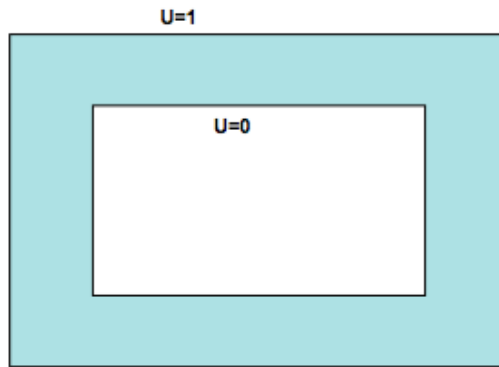


Figure 2: The Dirichlet Problem

1.2 Learning Harmonic Functions

In this section, we explore several ways of finding the solution of the two-dimensional harmonic function represented by the graph in Figure 1.

1.2.1 Linear Equation method

The second approach is to assume that the values of the unknown nodes depends on all neighbors, known and unknown. We find the value of each by solving a system of linear equations. These equations are formed by assuming harmonic nature of the graph, and the value of a node is the average of its neighbors.

$$x = \frac{0 + 0 + y + z}{4} \quad (1)$$

$$y = \frac{1 + 1 + x}{3} \quad (2)$$

$$z = \frac{x + 1}{2} \quad (3)$$

Substituting (2) and (3) in (1) and solving for x, we get

$$x = \frac{7}{19} = 0.37 \quad (4)$$

From (4), we find the values of y and z.

$$y = \frac{15}{19} = 0.79 \quad (5)$$

$$z = \frac{13}{19} = 0.69 \quad (6)$$

Note that in such problems, the values of the already labeled boundary nodes/conditions don't change, and the set B, of all boundary nodes, is defined as the set of all labeled nodes.

1.2.2 The method of relaxations

The method of relaxations is a discrete approximation to Laplace's (continuous) differential equation. We begin by assigning fixed values to the boundary points. Next, we assign arbitrary values to all other points.

At each iteration we adjust their values to be the average of their neighbors. We repeat till some criteria for convergence is achieved.

Following is an illustration of this method for the 2D harmonic function represented in Figure 1. We start assuming uniform distribution and assign the value 0.5 to each of x, y and z. As more and more information propagates through the graph, we adjust the values to values closer to the actual ones according to equations 1, 2 and 3.

Iteration	x	z	y
t = 0	0.5	0.5	0.5
t = 1	0.25	0.75	0.83
t = 2	0.4	0.62	0.75
t = 3	0.34	0.7	0.8
t = 4	0.38	0.67	0.78

We can see that these values are rather close to the values that we computed earlier. In the first iteration, we take account of the information one hop away, and so on, until all the information is factored in and the function converges.

However, bipartite graphs do not converge and exhibit oscillations.

1.2.3 Monte Carlo method

In this approach, we perform several random walks on the discrete representation. Now, we probabilistically compute the value of the harmonic function on that node, based on the similarity measure revealed by the random walk.

Xiaojin Zhu of Wisconsin Madison, [5] used the Cedar database, which is one of the largest databases of OCR data, to show the relationship between harmonic functions and Semi-supervised learning on such graphs. He built a graph of labeled and unlabeled OCR digits (from 0-9), with the labeled data constituting the set B of boundary nodes and the unlabeled data constituting the set D. The graph is created on the basis of this assumption A graph is given on the labeled and unlabeled data and that instances connected by heavy edge tend to have the same label.

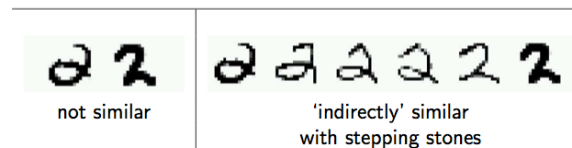


Figure 3: Cedar Database

The tutorial shows shows several interpretations of this graph such as an electrical network representation, a harmonic function interpretation, and a random walk interpretations as well. The idea was to learn 10 harmonic functions, one for every digit, to recognize the digits 0-9.

In this graph, random walks on this graph will attain an error-rate of 0.01 on conducting several runs, of the order 10,000.

Following is an example where Monte Carlo methods are employed for estimation. Let there be a dart board - shaped like a unit circle. The area of the square region encompassing the scoring area in the dart board has an area of 4 units. Therefore, assuming that all darts fall within the square, the percentage of darts that fall inside the circle = Area of unit circle/Area of enclosing square = $\pi/4$.

If we were able to estimate this percentage, we can calculate the value of π . This estimation can be done on the basis of a large number of trials as suggested by the law of large numbers in statistics and the law of averages in Physics.

1.2.4 Eigenvector method

The Eigenvector method looks at the stationary distribution of a random walk.

An eigenvector is an implicit "direction" for a matrix

$$A\vec{v} = \lambda\vec{v} \quad (7)$$

where \vec{v} (eigenvector) is non-zero, though λ (eigenvalue) can be any complex number in principle.

We compute eigenvalues by following the procedure shown below.

$$A = \begin{pmatrix} -1 & 3 \\ 2 & 0 \end{pmatrix} \quad (8)$$

$$A - \lambda I = \begin{pmatrix} -1-\lambda & 3 \\ 2 & -\lambda \end{pmatrix} \quad (9)$$

$$\text{Det}(A - \lambda I) = (-1 - \lambda) * (-\lambda) - 3 * 2 = 0 \quad (10)$$

$$\lambda + \lambda^2 - 6 = 0 \quad (11)$$

$$\lambda = 2, -3 \quad (12)$$

For $\lambda_1 = 2$:

$$\begin{pmatrix} -3 & 3 \\ 2 & -2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 0 \quad (13)$$

All vectors satisfying the property $x_1 = x_2$ are solutions.

1.3 Some Definitions

1.3.1 Stochastic Matrices

Stochastic matrices: each row (or column) adds up to 1 and no value is less than 0. Example:

$$A = \begin{pmatrix} 3/8 & 5/8 \\ 1/4 & 3/4 \end{pmatrix}$$

The largest eigenvalue of a stochastic matrix E is real: $\lambda_1 = 1$. For λ_1 , the left (principal) eigenvector is p, the right eigenvector = I (Identity matrix). In other words, $G^T p = p$.

1.3.2 Markov Chains

A homogeneous Markov chain is defined by an initial distribution x and a Markov kernel E. A 'path' is a sequence (x_0, x_1, \dots, x_n) . The probability of a path can be computed as a product of probabilities for each step i using the following formula:

$$X_i = x_{i-1} * E$$

We conduct several random walks to compute X_j , given x_0 , E, and j.

1.3.3 Stationary Solutions

The fundamental Ergodic Theorem for Markov chains [Grimmett and Stirzaker 1989] says that the Markov chain with kernel E has a stationary distribution p under three conditions: it must be stochastic, irreducible and aperiodic. To make sure that these conditions are true, all the rows of E add up to 1 (and no value is negative), E is strongly connected, and that E is not bipartite. This fact is used in the PageRank paper to prove that PageRank, which is this very stationary solution, actually exists.

1.3.4 Ergodic Chains

Loosely speaking, a graph is ergodic if it is possible to reach every node from every other node. A Markov chain is called an ergodic chain if it is possible to go from every state to every state (not necessarily in one move). This means that the chain must be recurrent. Ergodic Markov chains are also called irreducible. A Markov chain is called a regular chain if some power of the transition matrix has only positive elements [2].

This definition talks only about reachability. It says nothing about graphs like Tadpole graphs [4], or lollipop graphs [3], in which some regions are, probabilistically speaking, easy to get into, but difficult to get out of.

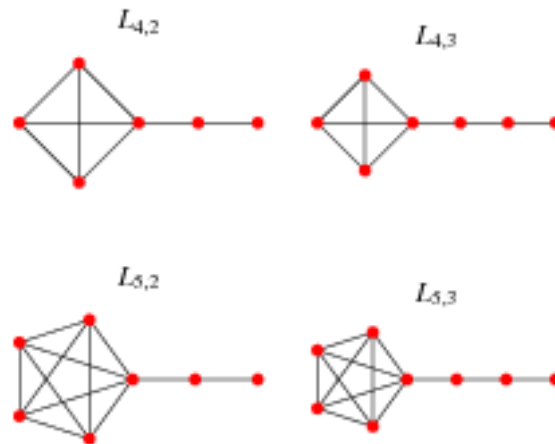


Figure 4: Lollipop Graphs

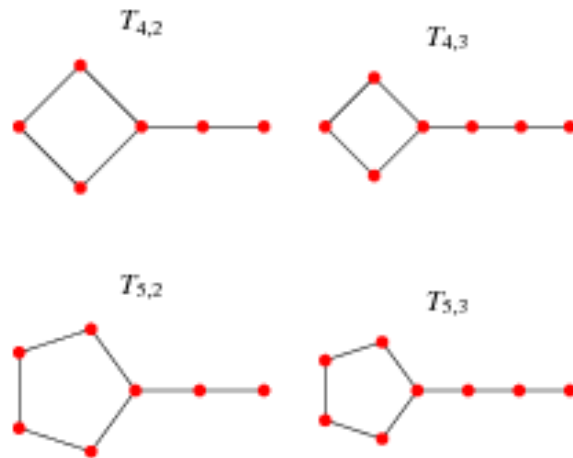


Figure 5: Tadpole Graphs

2 Readings

Pages 1-14, 29-37 (required), 37-51 (optional) of [1].

References

- [1] Peter G. Doyle and J. Laurie Snell. "Random Walks and Electric Networks". In: (2000). "1-14,29-37,37-51". URL: <http://www.citebase.org/abstract?id=oai:arXiv.org:math/0001057>.
- [2] Sergi Elizalde. *Ergodic Markov Chains*. URL: <http://www.math.dartmouth.edu/~m20x06/Lecture15.pdf>.
- [3] Eric W. Weisstein. "Lollipop Graph." *From MathWorld—A Wolfram Web Resource*. URL: <http://mathworld.wolfram.com/LollipopGraph.html>.
- [4] Eric W. Weisstein. "Tadpole Graph." *From MathWorld—A Wolfram Web Resource*. URL: <http://mathworld.wolfram.com/TadpoleGraph.html>.
- [5] Xiaojin Zhu. *Semi-Supervised Learning, ICML 2007 tutorial*. URL: <http://pages.cs.wisc.edu/~jerryzhu/pub/sslicml07.pdf>.