Belief contagion simulation

Christopher Brown

April 22, 2013

1 Exploratory graph sampling methods

Closed form analysis of a social network can become difficult or intractable as we consider a vast number of different variables, particularly individual (and directed) relationships between nodes on the graph. We want to discover the proper distributions and effects that occur naturally; to perform this inference, we must use approximate sampling techniques.

One goal of this project is to discover the similarities and differences between this model and a epidemiological disease contagion model. While there are a lot of similarities,

Belief Contagion	Epidemiology
Reliability of person	Physical proximity
Believability	Infectiousness
Period of tentative believability	Period of contagiousness
Cultural norms / common knowledge	Social structure
Daily information habits	Daily social habits
'Going viral'	$R_0 > 1$

There are also some notable differences—aspects of each model that don't seem to have a direct correlate in the other:

Belief Contagion	Epidemiology
Asymmetric reliability	??
Information source independence	??
??	Morbidity
??	Immunity
??	Recovery period

To explore this way of thinking about idea / information / belief propagation, we will use sampling techniques, which are inexact, but tractable, and sufficient for the claims we will be making.

Our Bayesian belief model predicts a significant difference in models: the effect of hearing the same belief from two independent sources will have a much greater change if the belief is *a priori* unlikely, than if it were more probably. Verifying this prediction is one of our current tasks.

2 Agent-based modeling with Akka

Akka is a Scala library for message-oriented programming. Message passing is a common way to abstract over thread-sharing and memory-sharing, and Akka provides a high-level interface so that I, the programmer, do not have to work directly with the MPI (Message Passing Interface) specification, or with network details. In Akka, every entity is an "actor," i.e., an object that is built to respond to incoming messages, with the ability to send outgoing messages.

This is a rich interface for the simulations we have been discussing because it replicates human autonomy to some degree. For example: I have contact information for my friends, and I can send them messages, and they can send me messages, but I can't directly control them. Passing messages around an Akka network feels natural; every actor receives messages and can choose how to respond based on its own internal state. But the actor does not have access to the senders' internal state, which would be a disadvantage in some modeling scenarios, but it is an apt simplification in this one.

However, there are a few complications. Everything is evented—based on message passing. There is no way to say, "finish sending all your messages and let me know when you're done." For example, to simulate a succession of days (avoiding messages suddenly getting received 'days' after they were sent), you must create an artificial bottleneck. There must be a controller to send out N *wake-up* messages (for N believer-simulators), and wait for N *done* messages, only proceeding to the next step after receiving all messages. Akka provides two types of message passing: normal send-and-forget messages (`!`) and futures (`?`). I only used the basic send-and-forget type of message. Using futures may alleviate some of the accounting that has to be kept to enforce some level of synchrony. I think the overall consensus is that for the smaller-scale models we're exploring at the moment, Akka is overkill, and the black-box effects of message-passing programming are not worth the performance / scalability benefits.

3 Visualization tools

The webpage project has two main goals. First, to visualize the different structures that are popular in the literature (e.g., Watts-Strogatz's 'small-world', BarabásiâĂŞAlbert). Second, to allow changing variables and immediately seeing the effect on the way "beliefs" or "diseases" propagate on the graph. While this isn't the appropriate framework for inferring parameters characterizing some specific population given a transcript of messages passed around the network and measures of adoption, I believe that developing intuitions by experimentation is important.

The tool is currently hosted at:

`http://beliefs.lingsa.org/`

And the open source repository can be found at `https://github.com/chbrown/beliefs`.

Currently, there are many more aspects that I hope to implement. Foremost, and in progress, are distributions for each variable instead of static values. That is, I want to be able to parameterize

the believability of a generated idea on the graph according to a beta, instead of just giving a static value like ‘0.333.’

Second, I want to support different models of infection / belief adoption. Right now there is a repeat-exposure threshold; at each iteration, a node will add the believability of a received message to its recorded believability of that message, which starts at zero. Once it is sent that message enough times, it will go over the threshold, and that node will adopt the belief.

In this implementation, nodes never ‘get better.’ It will be important to build in recovery rates, i.e., information becoming stale, outdated, or obsolete. But in that case, it will be important to define stopping criteria and how new ideas emerge; it will be much more organic, and thus more complicated.

See Figures 1 and 2 for screenshots of the interface.

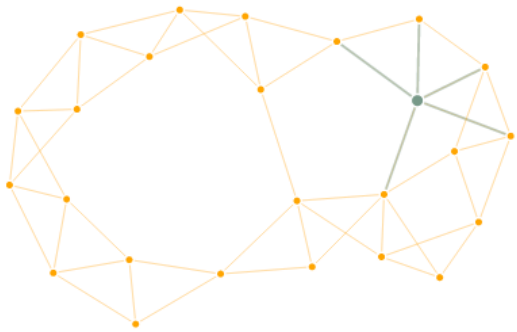

Figure 1: A simple “small world” graph that is predominantly regular, with four edges per node, but with a few irregular links that produce the small world effect. The blue node is “infected,” while the orange nodes are not.

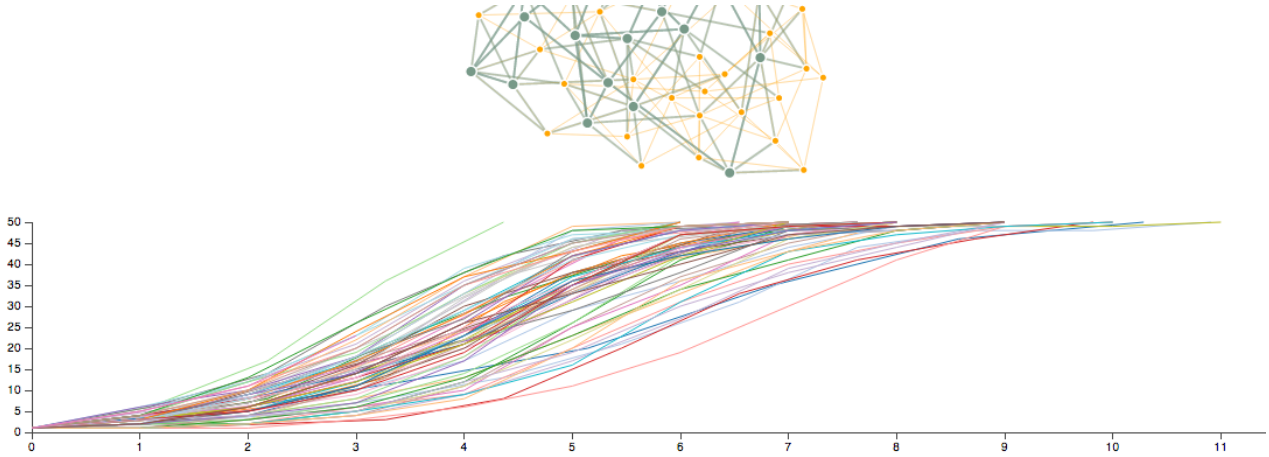

Figure 2: Several iterations plotted together: each line is one iteration. The horizontal axis displays discrete time steps; the vertical axis counts the total number of nodes infected at that time step.