A Recursive Algorithm for Spatial Cluster Detection

Xia Jiang, MS, Gregory F. Cooper, MD, PhD
Department of Biomedical Informatics, University of Pittsburgh, Pittsburgh, PA

Abstract

Spatial cluster detection involves finding spatial subregions of some larger region where clusters of some event are occurring. For example, in the case of disease outbreak detection, we want to find clusters of disease cases so as to pinpoint where the outbreak is occurring. When doing spatial cluster detection, we must first articulate the subregions of the region being analyzed. A simple approach is to represent the entire region by an $n \times n$ grid. Then we let every subset of cells in the grid represent a subregion. With this representation, the number of subregions is equal to $2^{n^2} - 1$. If n is not small, it is intractable to check every subregion. The time complexity of checking all the subregions that are rectangles is $\theta(n^4)$. Neill et al.[8] performed Bayesian spatial cluster detection by only checking every rectangle. In the current paper, we develop a recursive algorithm which searches a richer set of subregions. We provide results of simulation experiments evaluating the detection power and accuracy of the algorithm.

Introduction

Spatial cluster detection consists of finding spatial subregions of some larger region where clusters of some event are occurring. For example, in the case of disease outbreak detection, we want to find clusters of disease cases so as to pinpoint where the outbreak is occurring. Other applications of spatial cluster detection include mining astronomical data, medical imaging, and military surveillance. When doing spatial cluster detection, we must first articulate the subregions of the region being analyzed. A simple approach is to represent the entire region by an $n \times n$ grid. Then we let every subset of cells in the grid represent a subregion. This is the approach taken in this paper. With this representation, the number of subregions is equal to $2^{n^2} - 1$. If n is not small, it is intractable to check every subregion. The time complexity of only checking every subregion that is a rectangle is $\theta(n^4)$. Neill et al.[8] performed Bayesian spatial cluster detection by only checking every rectangle. In the current paper, we develop an algorithm which searches a richer set of subregions. The algorithm can be used in any application of spatial cluster detection. However, we test it specifically in the context of disease outbreak detection. So next we describe disease outbreak detection.

Disease Outbreak Detection: Le Strat and Carrat[6] define an epidemic as the occurrence of a number of cases of a disease, in a given period of time in a given population that exceeds the expected number. A disease outbreak is an epidemic limited to localized increase, e.g., in a town or institution. If we can recognize an outbreak and its potential cost early, we can take appropriate measures to control it. Monitoring a community in order to recognize early the onset of a disease outbreak is called disease outbreak detection.

Often the count of some observable event increases during an outbreak. For example, since *Cryptosporidium* infection causes diarrhea, the count of over-the-counter (OTC) sales of antidiarrheal drugs ordinarily increases during a *Cryptosporidium* outbreak. Typically, during an outbreak, the number of new outbreak cases increases each day of the outbreak until a peak is reached, and then declines. Accordingly, the count of the observable event also increases during the outbreak. Therefore, a number of classical time-series methods have been applied to the detection of an outbreak based on the count of the observable event. Wong and Moore[9] review many such methods. Jiang and Wallstrom[4] describe a Bayesian network model for outbreak detection that also looks at daily counts.

Cooper et al.[1] took a different approach when developing PANDA. Rather than analyzing data aggregated over the entire population, they modeled each individual in the population. PANDA consists of a large Bayesian network that contains a set of nodes for each individual in a region. These nodes represent properties of the individual such as age, gender, home location, and whether the individual visited the ED with respiratory symptoms. By modeling each individual, we can base our analysis on more information than that contained in a summary statistic such as over-the-counter sales of antidiarrheal drugs. PANDA is theoretically designed specifically for the detection of non-contagious outbreak diseases such as airborne anthrax or West Nile encephalitis. Cooper et al.[2] extended the PANDA system to model the CDC Category A diseases, (See http://www.bt.

cdc.gov/agent/agentlist-category.asp). This augmented system, which is called PANDA-CDCA, takes as input a time series of 54 possible ED chief complaints, and it outputs the posterior probability of each CDC Category A disease and several additional diseases.

In a given region being monitored an outbreak may occur (or at least start) in some subregion of that region. For example, a *Cryptosporidium* outbreak might occur only in a subregion in close proximity to a contaminated water distribution. We want to determine that subregion, which can sometimes be accomplished by doing spatial cluster detection.

Spatial Cluster Disease Outbreak Detection: Traditional spatial cluster detection focuses on finding spatial subregions where the count of some observable event is significantly higher than expected. A frequentist method for spatial cluster detection is the spatial scan statistic developed by Kulldorff[5]. Neill et al.[8] developed a Bayesian version of the spatial scan statistic. In their experiments, they only considered the set of all subregions that are rectangles. This paper describes an algorithm that investigates a richer subset of subregions than the set of rectangles. We test the algorithm by using it to perform Bayesian spatial outbreak detection with PANDA-CDCA. Therefore, before describing the algorithm, we review PANDA-CDCA.

PANDA-CDCA

Figure 1 shows the Bayesian network in PANDA-CDCA. We briefly describe the nodes in the network. Node *O* represents whether an outbreak is currently taking place. Node *OD* represents which outbreak disease is occurring if there is an outbreak. Node *F* represents the hypothetical fraction of individuals in the population who are afflicted with the outbreak disease and go to the ED, given that an outbreak is occurring. This node indicates the extent of the outbreak, if one is occurring. For the sake of computational efficiency, we modeled F as a discrete variable. Furthermore, we assumed all outbreak types are equally likely to have the various levels of severity. This assumption is not necessary, and there could be an edge from OD to F. Node D_r represents whether an individual arrives in the ED with a particular disease. There is one such node for each individual *r* in the population. One value is *NoED*, which means the individual does not visit the ED. Node C_r represents each of the possible chief complaints the individual could have when arriving in the ED.

To do inference, we proceed as follows. On each day, we know the value of C_r for each individual *r* in the population. We call the set of all these values our *Data*. Using the network in Figure 1, we then compute $P(OD=none|Data)$ and for each outbreak disease *d*, $P(OD=d|Data)$.

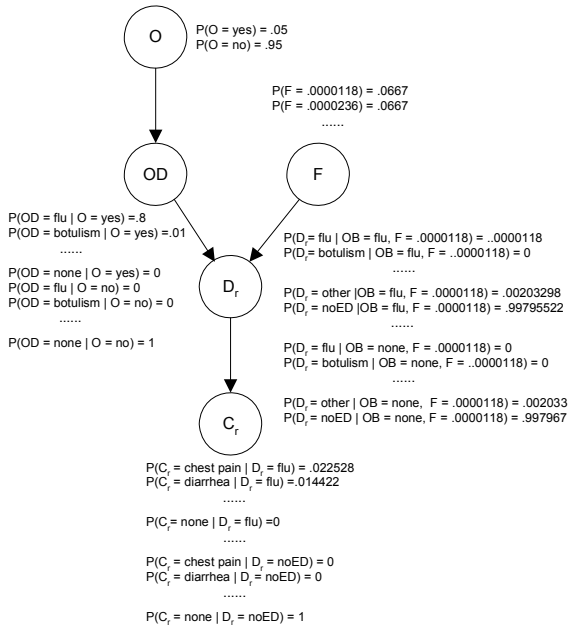

Figure 1. The PANDA-CDCA Bayesian network.

A Recurisve Algorithm for Spatial Cluster Detection of Complex Subregions

Next we develop a new algorithm for spatial cluster detection of complex subregions, and we apply the algorithm to outbreak detection using PANDA-CDCA. First we show how to compute the likelihood that a given subregion has an outbreak using PANDA-CDCA.

Computing the Likelihood of a Subregion: Let *OS* be a random variable, which represents the outbreak subregion, whose value is *none* if no outbreak is occurring, and whose value is *S* if an outbreak is occurring in subregion *S*. We want to compute $P(Data|OS=none)$ and for each subregion *S*, $P(Data|OS=S)$.

When $OS=none$ we assume the data is being generated according to the model shown in Figure 1 with *OD* set to *none*. Therefore, $P(Data|OS=none) = P(Data|OD=none)$, which is computed by doing inference in the network in Figure 1. When $OS=S$ we assume the data in subregion *S* is being generated according to the model in Figure 1 with *OD* set to one of the 13 diseases, and the data outside subregion *S* is

being generated by a separate model with *OD* set to *none*. Let $Data_{in}$ be the data concerning individuals in subregion *S* and $Data_{out}$ be the data concerning individuals outside of subregion *S*. Then $P(Data_{in}|OD=d,OS=S)$ and $P(Data_{out}|OD=d,OS=S)$ are each computed by doing inference in the network in Figure 1 with the instantiations just mentioned. We then compute

$$P(Data \mid OD = d, OS = S) =$$

$$P(Data_{in} \mid OD = d, OS = S) \times$$
$$P(Data_{out} \mid OD = d, OS = S).$$

Finally, we sum over *OD* to obtain the likelihood of subregion *S*.

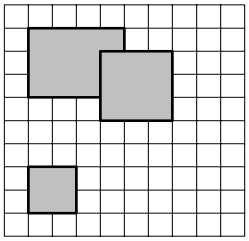

Figure 2. The shaded area is a possible subregion discovered by algorithm *refine*.

Finding a Likely Subregion: We can do Bayesian spatial cluster detection by only considering subregions that are rectangles, and assigning the same prior probability to all rectangles. Then, after computing the likelihoods discussed in the previous subsection, we use Bayes' Theorem to calculate $P(OS=none|Data)$ and $P(OS=R|Data)$ for every rectangle *R*. The posterior probability of an outbreak is then equal to $\sum_R P(OS=R|Data)$. We can then base the detection of an outbreak on this posterior probability, and report the posterior probability of each rectangle. The most probable rectangle is then considered to be the most likely subregion where the outbreak is occurring. The algorithms described next assume that we have done this. They then search for a more likely subregion than the most probable rectangle.

For subregion *S*, let $P(Data|OS=S)$ be the "score" of *S*. If we let $Score_{best}$ be the score of the most probable rectangle, we can possibly find a higher scoring subregion by seeing if we can increase the score by joining other rectangles to this rectangle. The following is an algorithm that repeatedly finds the rectangle that most increases the score and joins that rectangle to our current subregion. It does this until no rectangle increases the score. By $score(G,S)$ we mean the score of subregion *S* in grid *G*.

```
void refine (grid G; subregion& S_best)
    determine highest scoring rectangle R_best in G;
    S_best = R_best;
    Score_best = score(G, S_best);
    flag R_best;
    repeat
        found = false;
        for (each unflagged rectangle R in G) {
            S_try = S_best ∪ R;
            if (score(G, S_try)) > Score_best) {
                found = true;
                Score_best = score(G, S_try);
                T = R;}}
        if (found) {
            S_best = S_try ;
            flag T; }
    until (not found);
```

The algorithm would be called as follows (*G* is the entire grid): $refine(G, S_{best})$. The worst case time complexity of the algorithm is $O(n^8)$. Figure 2 shows a possible subregion discovered by algorithm *refine*. In order to model that more complex subregions have a lower prior probability than less complex ones, in each iteration of the **repeat** loop we multiplied the score by a penalty factor.

We might do better if, when we find a rectangle *R* in our grid *G* that increases the score, we treat *R* as grid, recursively call *refine* with *R* as the input grid, find the best subregion V_{best} in *R*, at the top level check if V_{best} increase the score in *G* more than *R*, and, if so, replace *R* by V_{best}. The algorithm that follows does this.

```
void refine2 (grid G; subregion& S_best, int level)
    if (level ≤ N) { // N is the recursion depth.
        determine highest scoring rectangle R_best in G;
        S_best = R_best;
        Score_best = score(G, S_best);
        flag R_best;
        if (level < N) {
            refine2(S_best, V_best, level + 1);
            if (score(G, V_best) > Score_best) {
                S_best = V_best;
                Score_best = score(G, V_best); }}
        repeat
            found = false;
            for (each unflagged rectangle R in G)
                S_try = S_best ∪ R;
```

```
if (score(G, S_try)) > Score_best ) {
    if (level < N) {
        refine2(R, V_best, level + 1);
        if (score(G, S_best ∪ V_best)
            > score(G, S_try) )
            S_try = S_best ∪ V_best; }
        found = true;
        Score_best = score(G, S_try);
        T = R; } }
    if (found) {
        S_best = S_try ;
        flag T; }
until (not found); }
```

The top-level call is as follows: $refine2(G, S_{best}, 0)$.

If the rectangles recursively become sufficiently small, algorithm *refine2* can detect an outbreak of any shape.

Experiments

Method: We simulated a region covered by a 10×10 grid. Using a Poisson distribution with mean 9500, we randomly generated the number of people in each cell of the grid. Next, using this simulated population, the Bayesian network in PANDA-CDCA with the outbreak node O instantiated to *no*, and logic sampling[7] we simulated ED visits during a one year period in which no outbreak was occurring. For each cell, we determined the mean and standard deviation σ of the number of ED visits for that cell. We simulated 3 types of 30-day influenza outbreaks: mild, moderate, and severe. To simulate a mild outbreak in a given cell, which reaches its peak on the 15th day, we assumed that 15σ extra ED visits (due to patients with influenza) occurred in the first 15 days in the cell, and then we solved

$$\Delta + 2\Delta + \cdots + 15\Delta = 15 \times \sigma$$

for Δ. We next injected Δ new ED visits in the cell on day 1, 2Δ on day 2,…, and tΔ on day t. We did this for 12 days. (Outbreaks were always detected by the 12th day.) To simulate moderate and severe outbreaks, we repeated this procedure with values of 2σ and 3σ. The following table shows the average value of Δ for each type of outbreak:

Outbreak Type	Stand. Deviations	Avg. Δ
mild	σ	.443
moderate	2σ	.886
severe	3σ	1.329

The number of injected ED visits must be an integer. We rounded down when $t\Delta < .5$, and up otherwise. Figure 3 shows a simulated outbreak in one cell.

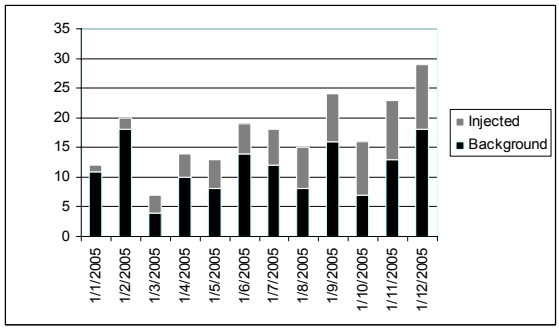

Figure 3. A simulated moderate outbreak.

We simulated outbreaks in six different types of subregions. The first was a T-shaped subregion, the second L-shaped, the third a cross, and the last three were three different separated rectangles. Figure 4 shows the T-shaped subregion and one of the separated-rectangles subregions. For each outbreak type, for each of the six subregion types, we did 12 simulations at different times during the one year background period. This made a total of 72 simulations for each of the three outbreak types. We used Algorithm *refine2* with a recursion depth of 5 to determine the outbreak subregion.

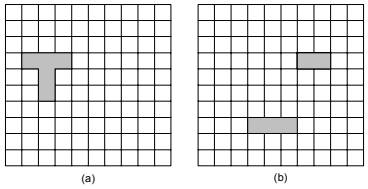

Figure 4. The injected T-subregion is shown in (a), and one of the injected separated-rectangles subregions is shown in (b).

Results: To measure detection power, we used AMOC curves[3]. In such curves, the annual number of false positives is plotted on the x-axis and the mean day of detection on the y-axis. Figure 5 shows AMOC curves for each of the outbreak types. To measure detection accuracy, we used the following function: $similarity(S_1,S_2) = \#(S_1 1 S_2) \;/\; \#(S_1 \chi S_2)$, where # returns the number of cells in a subregion. This function is 0 if and only if two subregions do not intersect, while it is 1 if and only if they are the same subregion. For each outbreak type, we determined the mean of the similarities between the detected subregions and the injected subregions on each day of the outbreaks. The graphs of these relationships appear in Figure 6. The mean similarity for mild outbreaks is about 0 on day 1, and for moderate and

severe outbreaks it has about the same value on day 1. This may be due to rounding. For example, since for mild outbreaks the average Δ=.443, no ED visits were often injected on the first day of such outbreaks.

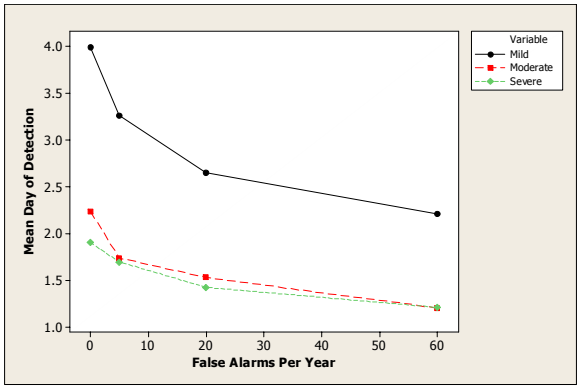

Figure 5. AMOC curves.

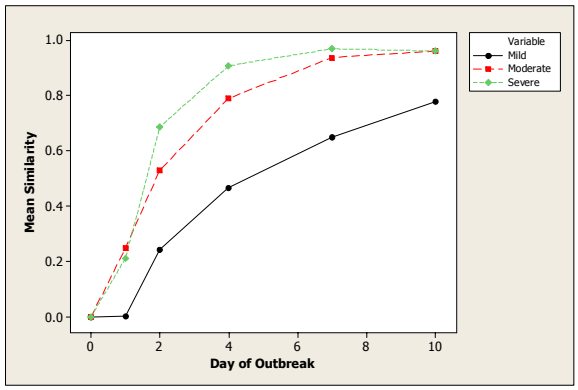

Figure 6. Mean similarities between detected subregion and injected subregion.

Discussion and Conclusions

The results are encouraging. They indicate that, on the average, we can detect 30-day severe, moderate, and mild outbreaks in complex subregions, respectively about 1.9, 2.2, and 4.0 days into the outbreak. Furthermore, the similarity between the detected subregion and the outbreak subregion averages about .7 by the 2nd, 3rd, and 8th days respectively of severe, moderate, and mild outbreaks.

We presented a recursive algorithm for detecting outbreaks in complex subregions. The results reported here provide support that the algorithm is a promising method for detecting such outbreaks.

In this preliminary evaluation, we used simulated data in order to test the inherent detection capability of the algorithm under well controlled conditions. Given that the results were promising, we next plan to evaluate the algorithm using real data and compare its results to that of other approaches.

Acknowledgements: This work was funded by a grant from the National Science Foundation Grant (IIS-0325581).

References

1. Cooper, G.F., Dash, D.H., Levander, J.D., Wong, W.K., Hogan, W.R., Wagner, M.M. 2004. Bayesian Biosurveillance of Disease Outbreaks. *Proceedings of 20th Conference on Uncertainty in Artificial Intelligence*. Arlington, VA.
2. Cooper, G.F., Dowling, J.N., Lavender, J.D., Sutovsky, P. 2006. A Bayesian Algorithm for Detecting CDC Category A Outbreak Diseases from Emergency Department Chief Complaints. *Proceedings of Syndromics 2006*, Baltimore, MD.
3. Fawcett, T., Provost, F. 1999. Activity Monitoring: Noticing Interesting Changes in Behavior. In *Proceedings of the Fifth SIGKDD Conference on Knowledge Discovery and Data Mining*. San Diego, CA: ACM Press.
4. Jiang, X.,Wallstrom, G.L. 2006. A Bayesian Network for Outbreak Detection and Prediction. In *Proceedings of AAAI-06*, Boston, MA.
5. Kulldorff, M. 1997. A Spatial Scan Statistic. *Communications in Statistics: Theory and Methods* 26, 6.
6. Le Strat, Y., Carrat, F. 1999. Monitoring Epidemiological Surveillance Data using Hidden Markov Models. *Statistics in Medicine* 18.
7. Neapolitan, R.E. 2004. *Learning Bayesian Networks*. Upper Saddle River, NJ: Prentice Hall.
8. Neill, D.B., Moore, A.W., Cooper, G.F. 2005. A Bayesian Spatial Scan Statistic. *Advances in Neural Information Processing Systems* 18.
9. Wong, W.K., Moore, A. 2006. Classical Time Series Methods for Biosurveillance. In Wagner, M. ed. *Handbook of Biosurveillance*, New York, NY: Elsevier.