A modelling approach to locating explosions

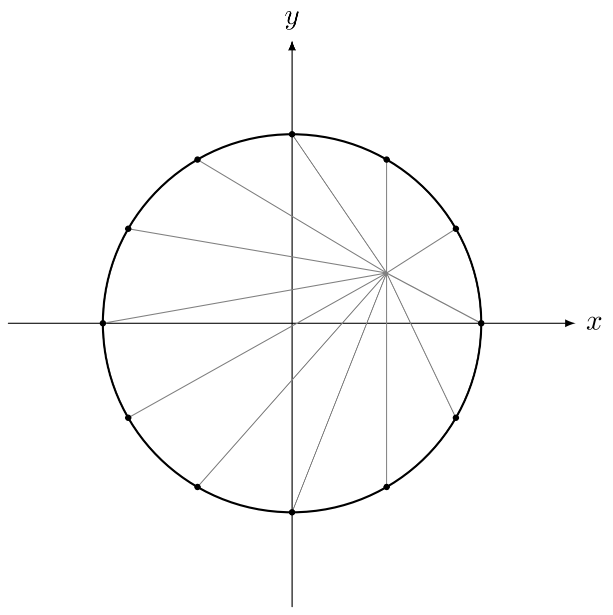
David Barber
University College London

October 14, 2017

1 Making a model

We consider a modified and simplified form of Stuart Russell's 'earthquake/nuclear blast' detection problem (see NIPS paper). We assume that there is an 'explosion event' somewhere within the earth and we wish to estimate where the explosion happened based on surface measurements of the explosion. For simplicity we assume a two dimensional earth.

There are N sensors evenly spread out on the surface of the earth with locations $(x_i, y_i), i = 1, \ldots, N$, indicated by the dots below:



The explosion happens at some (unknown) location (e_x, e_y) in the earth, from which a wave of energy propagates outwards and reaches the sensors on the surface. The amount that each sensor measures is given by

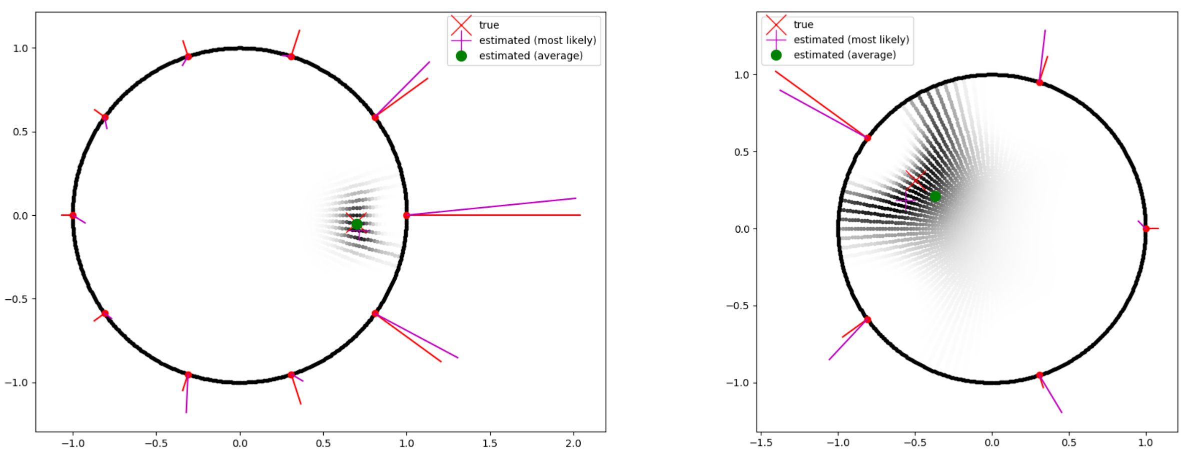
$$\frac{1}{d_i^2 + 0.1}$$

where d_i^2 is the squared distance from the explosion to sensor i and

$$d_i^2 = (x_i - e_x)^2 + (y_i - e_y)^2$$

This means that the signal strength of the explosion decreases as the distance from the explosion to a sensor increases.

The sensors are not able to perfectly detect the strength of the signal and the signal is measured with Gaussian noise with a standard deviation of σ. This means that the observed value v_i at sensor i follows a



(a) (b)

Figure 1: The posterior distribution for the location of the explosion (darker corresponds to a higher probability). This is based on a spiral coordinate system (see earthquake.jl) with $\sigma = 1$. The red points are the locations of the surface sensors. The observed (noisy) measurements at each sensor are represented by the magenta lines, and the true (unknown) blast values are denoted by the red lines. (a) Using 10 sensors. (b) Using 5 sensors. Note how the uncertainty in the posterior is larger in the case of having fewer sensors.

Gaussian distribution:

$$p(v_i|d_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}\left(v_i - \frac{1}{d_i^2+0.1}\right)^2}$$

Assuming that the observed sensor values are independent (given the explosion location), our simple generative model is

$$p(v_1, \ldots, v_N, e_x, e_y) = p(e_x, e_y) \prod_{i=1}^{N} p(v_i|d_i)$$

which has the belief network description below

$$\begin{tikzcd}
 & & e_x, e_y \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & & \\
v_1 & v_2 & v_3 & \cdots & v_N
\end{tikzcd}$$

2 Computing the posterior

Given an observed set of values $v_1, \ldots, v_N$, our interest is then the posterior distribution

$$p(e_x, e_y|v_1, \ldots, v_N)$$

For a uniform prior $p(e_x, e_y) = \mathsf{const.}$, then

$$p(e_x, e_y|v_1, \ldots, v_N) \propto \prod_{i=1}^{N} p(v_i|d_i)$$

In the code we plot a representation of the posterior and also plot the most likely point $\arg\max p(e_x, e_y|v_1, \ldots, v_N)$.

```
function earthquake()

# explosion detector (using spiral coorindate system)
# run this file from julia by typing:]
# julia> using PyPlot
# julia> include("earthquake.jl")
# julia> earthquake()

# define the coordinate system:
S=5000 # number of points on the spiral
x=zeros(S); y=zeros(S)
for s=1:S
theta=50*2*pi*s/S;  r=s/S
x[s]=r*cos(theta); y[s]=r*sin(theta)
end
plot(x,y,".")

# define the locations of the detection stations on the surface
# Also define what value on each sensor would be generated by an explostion at internal location s
N=10 # number of stations
x_sensor=zeros(N); y_sensor=zeros(N)
v=zeros(S,N)
for sensor=1:N
theta_sensor=2*pi*sensor/N
x_sensor[sensor]=cos(theta_sensor); y_sensor[sensor]=sin(theta_sensor)
for s=1:S
v[s,sensor]=value(x[s],y[s],x_sensor[sensor],y_sensor[sensor]) # explosion value
end
end


sd=1 # standard deviation of the Gaussian noise

# Make the explosion data:
true_s=randperm(S)[1] # true location of the explosion
theta=50*2*pi*true_s/S
r=true_s/S
x_true=r*cos(theta); y_true=r*sin(theta)

# Get the noisy sensor values that will be observed for this explosion:
val=zeros(N)
val_clean=zeros(N) # unknown clean values (just for interest)
for sensor=1:N
val_clean[sensor]=value(x_true,y_true,x_sensor[sensor],y_sensor[sensor])
val[sensor]=val_clean[sensor]+sd*randn()
end
figure()
plot(1:N,val,label="noisy observed sensor measurements")
plot(1:N,val_clean,label="clean (unknown) sensor measurements")
legend()

# Perform inference p(location|observed sensor values) given these sensor values
logp=zeros(S)
for s=1:S
for sensor=1:N
logp[s] += -0.5*(val[sensor]-v[s,sensor])^2/(sd^2) # Gaussian distribution
end
end
p=exp.(logp-maximum(logp)) # do exponentiation (and avoid over/underflow)
p=p/sum(p) # normalise


# plot the posterior and most likely location of the explosion:
maxp,maxind =findmax(p)
figure()
for s=1:S
plot(x[s],y[s],".",color=(1-(p[s]/maxp))*[1,1,1])
end
```

```
for theta=0:0.01:2*pi
plot(cos(theta),sin(theta),".",color=[0,0,0])
end
for sensor=1:N
plot(x_sensor[sensor],y_sensor[sensor],"o",color=[1,0,0])
end
plot(x_true,y_true,"rx",markersize=20,label="true")
plot(x[maxind],y[maxind],"m+",markersize=20,label="estimated (most likely)")
plot(sum(p.*x),sum(p.*y),"go",markersize=10,label="estimated (average)")
legend()

end

function value(x_true,y_true,x_sensor,y_sensor)
return 1/(0.1+ (x_true-x_sensor)^2 + (y_true-y_sensor)^2)
end
```