Seismic 2-D

Nimar S. Arora
Bayesian Logic, Inc.
Union City, CA 94587

Stuart Russell
Dept. of Computer Science
Berkeley, CA 94720

March 8, 2015

Abstract

The goal of this problem is to detect and localize seismic events on a simulated two-dimensional world (the surface of a perfect sphere) given signals collected over a fixed time interval at a number of seismic stations. This is a simplification of the real problem (Arora et al., 2013), and is designed as a challenge problem for research in probabilistic programming languages.

1 Generative Model

1.1 Events

Events are generated by a homogenous space-time Poisson process over the surface of the earth with rate parameter λ_e. We assume that the earth is a perfect sphere with radius R, and we are only interested in events that occur in an episode of length T in time. In other words, the number of events in any episode has a Poisson distribution with rate $\lambda_e 4\pi R^2 T$. Further, it follows that the time of each event is uniformly distributed in $[0, T]$, and the location is uniformly distributed over the surface of the earth. If locations are represented by longitude and latitude then one can equivalently state that longitudes are uniformly distributed over $[-180, 180]$ and the *sin* of the latitude is uniformly distributed over $[-1, 1]$.

More formally, if e is the set of events in an episode, then

$$|e| \sim \text{Poisson}(\,\cdot \mid \lambda_e 4\pi R^2 T),$$

and for each event e^i with time, longitude, and latitude given by e^i_t, e^i_{l1}, and e^i_{l2} respectively,

$$\begin{aligned} e^i_t &\sim \text{Uniform}(\,\cdot \mid 0,\ T) \\ e^i_{l1} &\sim \text{Uniform}(\,\cdot \mid -180,\ 180) \\ \sin(e^i_{l2}) &\sim \text{Uniform}(\,\cdot \mid -1,\ 1). \end{aligned}$$

The event magnitude, e^i_m is distributed as per an exponential distribution with scale θ_m, a minimum value of μ_m and a maximum of γ_m, i.e.,

$$e^i_m \sim \text{Exponential}(\,\cdot \mid \mu_m,\ \theta_m,\ \gamma_m).$$

This magnitude model is based on the well known Gutenberg-Richter law (Wikipedia, 2015d). The maximum magnitude is a result of a phenomenon known as magnitude saturation.

1.2 True Detections

The seismic energy from an event travels radially outwards in distinct phases, each of which may or may not be detected by a station depending on local noise levels. For example, the energy can travel via different modes of transmission (compression waves, shear waves) and along different layers of the earth, the so called *body* waves or along the surface, refer to International Seismological Centre (2011) for a full list. In this work we only consider the first arriving phase, the P phase.

We define a true detection Λ^{ik} as the moment of first arrival of the energy from an event i at a seismic station k. Various signal processing algorithms are applied to the raw waveforms to detect an arrival, and then station processing algorithms collect various attributes of the detection such as time, azimuth, slowness, and amplitude referenced by Λ^{ik}_t, Λ^{ik}_z, Λ^{ik}_s, and Λ^{ik}_a respectively. Time is quite obviously the detection time of the energy, azimuth refers to the geographical direction of the incoming seismic waves, and amplitude is the height of the initial peak. Slowness is a more peculiar term, it refers to the inverse of the apparent surface speed of the waves, which will become clearer shortly.

1.2.1 Detection Probability

The probability that an event e^i is detected at station s^k is a function of the event magnitude and the great-circle distance, Δ_{ik}, between the event and the station. If $\Lambda^{ik} = \zeta$ represents a *mis*-detection then,

$$P(\Lambda^{ik} \neq \zeta \mid e^i, s^k) = \text{logistic}(\mu^k_{d0} + \mu^k_{d1} e^i_m + \mu^k_{d2}\Delta_{ik})\ .$$

1.2.2 Detection Time

The theoretical travel time of a seismic wave at a distance of δ is given by the travel time function,

$$I_T(\delta) = -.023 \times \delta^2 + 10.7 \times \delta\ + 5.$$

The detection time has a Laplacian distribution centered near the theoretical detection time,

$$\Lambda^{ik}_t \sim \text{Laplacian}(\,\cdot \mid e^i_t + I_T(\Delta_{ik}) + \mu^k_t\ ,\ \theta^k_t).$$

Note that the detections with time greater than T will not be available in the episodic data.

1.3 Detection Azimuth

The difference of the detection azimuth, Λ_z^{ik} from the theoretical station-to-event azimuth, $G_z(s_l^k, e_l^i)$ is distributed as a Laplacian,

$$\psi(G_z(s_l^k, e_l^i), \Lambda_z^{ik}) \sim \text{Laplacian}(\ \cdot \mid \mu_z^k,\ \theta_z^k\ ).$$

Here s_l^k and e_l^i refer to the locations (pair of longitude and latiude) of the station and the event respectively. See Appendix A for a definition of G_z and ψ.

1.4 Detection Slowness

The slowness at distance δ given by $I_S(\delta)$ is simply the derivative of the travel time. In other words, slowness measures the time that the seismic wave takes to travel between two points very close to the station in the direction of the event-to-station azimuth. The theoretical slowness is given by the formula,

$$I_S(\delta) = -.046 \times \delta + 10.7\,.$$

Note that I_S is always positive since $\delta \in [0, 180]$.

The detection slowness is a Laplacian centered near the theoretical slowness,

$$\Lambda_s^{ik} \sim \text{Laplacian}(\ \cdot \mid I_S(\Delta_{ik}) + \mu_s^k\ ,\ \theta_s^k).$$

1.5 Detection Amplitude

The log of the detection amplitude has a Gaussian distribution with a mean determined by the event magnitude and travel time.

$$\log(\Lambda_a^{ik}) \sim \text{Gaussian}(\ \cdot \mid \mu_{a0}^k + \mu_{a1}^k e_m^i + \mu_{a2}^k I_T(\Delta_{ik})\ ,\ \sigma_a^k\ ).$$

1.6 False Detections

Each station k has its own time-homogenous Poisson process generating false detections with rate λ_f^k. In other words, if ξ^k is the set of false detections in an episode,

$$|\xi^k| \sim \text{Poisson}(\ \cdot \mid \lambda_f^k T),$$

and the detection time ξ_t^k is uniformly distributed,

$$\xi_t^k \sim \text{Uniform}(\ \cdot \mid 0,\ T).$$

The azimuth and slowness are also uniformly distributed with their allowed ranges as follows:

$$\begin{aligned} \xi_z^k &\sim \text{Uniform}(\ \cdot \mid 0,\ 360), \\ \xi_s^k &\sim \text{Uniform}(\ \cdot \mid I_S(180),\ I_S(0)). \end{aligned}$$

However, the log-amplitude is distributed as Cauchy,

$$\log \xi_a^k \sim \text{Cauchy}(\ \cdot \mid \mu_f^k,\ \theta_f^k).$$

2 Hyperpriors and Constants

$$\begin{aligned}
T &= 3600\, s \\
R &= 6371\, km \\
\lambda_e &\sim \text{Gamma}(\ \cdot \mid 6.0, \frac{1}{4\pi R^2 T}) \\
\mu_m &= 3.0 \\
\theta_m &= 4.0 \\
\gamma_m &= 6.0 \\
\begin{bmatrix} \mu_{d0}^k \\ \mu_{d1}^k \\ \mu_{d2}^k \end{bmatrix} &= \text{MVarGaussian}\left(\cdot \ \middle|\ \begin{bmatrix} -10.4 \\ 3.26 \\ -0.0499 \end{bmatrix}, \begin{bmatrix} 13.43 & -2.36 & -0.0122 \\ -2.36 & 0.452 & 0.000\,112 \\ -0.0122 & 0.000\,112 & 0.000\,125 \end{bmatrix} \right) \\
\mu_t^k &= 0 \\
\theta_t^k &\sim \text{InvGamma}(\ \cdot \mid 120,\ 118) \\
\mu_z^k &= 0 \\
\theta_z^k &\sim \text{InvGamma}(\ \cdot \mid 5.2,\ 44) \\
\mu_s^k &= 0 \\
\theta_s^k &\sim \text{InvGamma}(\ \cdot \mid 6.7,\ 7.5) \\
\begin{bmatrix} \mu_{a0}^k \\ \mu_{a1}^k \\ \mu_{a2}^k \end{bmatrix} &= \text{MVarGaussian}\left(\cdot \ \middle|\ \begin{bmatrix} -7.3 \\ 2.03 \\ -0.001\,96 \end{bmatrix}, \begin{bmatrix} 1.23 & -0.227 & -0.000\,175 \\ -0.227 & 0.0461 & 0.000\,024\,5 \\ -0.000\,175 & 0.000\,024\,5 & 0.000\,000\,302 \end{bmatrix} \right) \\
(\sigma_a^k)^2 &\sim \text{InvGamma}(\ \cdot \mid 21.1,\ 12.6) \\
\lambda_f^k &\sim \text{Gamma}(\ \cdot \mid 2.1,\ 0.0013) \\
\mu_f^k &\sim \text{Gaussian}(\ \cdot \mid -0.68,\ 0.68) \\
\theta_f^k &\sim \text{InvGamma}(\ \cdot \mid 23.5,\ 12.45)
\end{aligned}$$

3 Stations

Abbreviation	Station Number	Longitude	Latitude
ASAR	0	133.9	-23.7
CMAR	1	98.9	18.5
FINES	2	26.1	61.4
ILAR	3	-146.9	64.8
MKAR	4	82.3	46.8
SONM	5	106.4	47.8
STKA	6	141.6	-31.9
TORD	7	1.7	13.1
WRA	8	134.3	-19.9
ZALV	9	84.8	53.9

4 Data and Evaluation

The data for this problem is several hundred megabytes, and has to be downloaded separately. The key point worth noting is that we are providing separate training and test data sets that are generated from the same underlying physics. The file `test.data` contains the fully labeled test data, and the file `training.data` is the fully labeled training data. In order to avoid accidental peeking, we are also providing an unlabeled copy of the test data in `test.blind`. The evaluation function is described in detail in the following sections. However, we are providing a simple script, `evaluate.py` to perform the evaluation.

A short data set of 100 episodes has been included for the purpose of evaluation on the challenge problem. A larger data set of 10000 episodes has also been included for further research.

4.1 File Format

Each data file consists of a number of episodes that are separated by a blank line. Each episode has subsections for the events, detections, and associations that took place in one episode of T seconds. The format of the blind data is identical, however it has no events or associations.

```
Episodes:

Events:
<longitude> <latitude> <magnitude> <time>
...
Detections:
<station number (zero-based)> <time> <azimuth> <slowness> <amplitude>
...
```

```
Assocs:
<event number (zero-based)> <detection number (zero-based)>
...
```

4.2 Evaluation

In each episode, the predicted events are matched against the ground truth events using min-weight max-cardinality matching. Only events that are within $W_T = 50$ seconds and $W_D = 5$ degrees of each other are considered as potential matches. The weight of a matched pairs of events e and e' is,

$$\frac{|e_t - e'_t|}{W_T} + \frac{dist(e, e')}{W_D}.$$

Given a matching, we can compute the precision, recall, and F-1 score in each episode as well as over the entire data set. Additionally, we will report the errors in magnitude estimates, distance, and time for the matched events.

There is just one caveat here. Some of the ground truth events don't have at least two associations, and thus can't be located reasonably. These events have to be excluded from the matching and the subsequent recall score.

The provided script `evaluate.py` may be used to automatically evaluate a solution bulletin. The format of the solution that is expected by this script is identical to that used for the other data files. The list of detections may be left out. It is highly recommended though that the list of associations should not be left out because future versions of the script may match associations as well.

The following shows the output from the script when run against a partial bulletin with only four episodes solved.

```
./evaluate.py short_data/test.data mysolution.data
Guess data has fewer episodes than gold data!!
10 matchable events, 12 guess events, and 4 matched
Precision 33.3 % , Recall 40.0 % , F1 36.4
Time Errors mean 8.5 std 4.7
Dist Errors mean 1.4 std 0.6
Mag Errors mean 0.2 std 0.1
```

4.3 Metric and Submission

This problem does not require a performance profile. Teams should report the metrics output by the evaluation script at the point where their solution halts. Teams should also report the CPU time consumed by their solution, in ms.

Appendices

A Spherical Geometry Functions

A.1 Azimuth

The azimuth of location $b = (lon_2, lat_2)$ as observed from location $a = (lon_1, lat_1)$ is given by the function $G_z(a, b) \in [0, 360)$. Where 0 is due north, 180 is due south, and 270 is due west. If $dlon = lon_2 - lon_1$ then we define

$$G_z(a,b) = [G'_z(a,b) + 360] \mod 360,$$
$$\text{where } G'_z(a,b) = \operatorname{atan2}\left(\sin(dlon)\ ,\ \cos(lat_1)\tan(lat_2) - \sin(lat_1)\cos(dlon) \right).$$

See Wikipedia (2015c) for an explanation of $G'_z(a, b)$.

A.2 Azimuth Difference

The difference between two azimuths $\psi(z_1, z_2)$ measures whether z_2 is clockwise from z_1, i.e. $\psi \in [0, 180]$ or counter-clockwise, $\psi \in [0, -180]$.

$$\psi(z_1,z_2) = \begin{cases} \psi'(z_1,z_2) - 360 & \text{if } \psi'(z_1,z_2) > 180 \\ \psi'(z_1,z_2) & \text{otherwise} \end{cases}$$

where,

$$\psi'(z_1,z_2) = [z_2 - z_1 + 360] \mod 360.$$

A.3 Great-Circle Distance

From Wikipedia (2015b):

> The great-circle distance is the shortest distance between two points on the surface of a sphere measured along the surface of the sphere.

We use the Vincenty version of the formula. If a and b are two points on the surface of a sphere described by longitude, latitude pairs (lon_1, lat_1) and (lon_2, lat_2) respectively, and $dlon$ is the difference in the longitudes, then

$$dist(a,b) = \operatorname{atan2}(y,x),$$

where

$y = \sqrt{(\cos(lat_2)\sin(dlon))^2 + (\cos(lat_1)\sin(lat_2) - \sin(lat_1)\cos(lat_2)\cos(dlon))^2}$, and
$x = \sin(lat_1)\sin(lat_2) + \cos(lat_1)\cos(lat_2)\cos(dlon)$.

Note that $atan2$ (Wikipedia, 2015a) has range $[0, 180]$ degrees when the first argument is non-negative. Hence $dist$ has range $[0, 180]$.

B Statistical Distributions and Functions

B.1 Cauchy

The Cauchy distribution with location μ and scale θ has probability density

$$\text{Cauchy}(x \mid \mu,\ \theta) = \frac{1}{\pi\theta}\ \frac{1}{1 + (\frac{x-\mu}{\theta})^2}$$

defined over all $x \in \mathbb{R}$.

B.2 Exponential

The Exponential distribution with location (or minimum value) μ, scale θ, and maximum value γ_m has the probability density

$$\text{Exponential}(x \mid \mu,\ \theta,\ \gamma) = \frac{1}{\theta}\ \frac{1}{1 - e^{-\frac{\gamma-\mu}{\theta}}}\ e^{-\frac{x-\mu}{\theta}}$$

defined over all $x \in \mathbb{R}$, $x \geq \mu$, and $x < \gamma$.

B.3 Gaussian

The Gaussian distribution with mean μ and standard deviation σ has probability density

$$\text{Gaussian}(x \mid \mu,\ \sigma) = \frac{1}{\sqrt{2\pi}\sigma}\ e^{-\frac{(x-\mu)^2}{2\sigma^2}}\ ,$$

defined over all $x \in \mathbb{R}$.

B.4 Gamma

The Gamma distribution with shape α and scale θ has probability density

$$\text{Gamma}(x \mid \alpha,\ \theta) = \frac{1}{\Gamma(\alpha)\ \theta^\alpha}\ x^{\alpha-1}e^{-\frac{x}{\theta}}\ ,$$

defined over all $x \in \mathbb{R}_{>0}$.

B.5 Inverse-Gamma

The Inverse-Gamma distribution with shape α and scale θ has probability density

$$\text{InvGamma}(x \mid \alpha,\ \theta) = \frac{\theta^\alpha}{\Gamma(\alpha)}\ x^{-\alpha-1}e^{-\frac{\theta}{x}}\ ,$$

defined over all $x \in \mathbb{R}_{>0}$.

B.6 Logistic

The logistic function is defined as,

$$\text{logistic}(x) = \frac{1}{1 + e^{-x}} \; .$$

B.7 Laplacian

The Laplacian distribution with location μ and scale θ has probability density

$$\text{Laplacian}(x \mid \mu,\ \theta) = \frac{1}{2\theta} \; e^{-\frac{|x-\mu|}{\theta}} \; ,$$

defined over all $x \in \mathbb{R}$.

B.8 Multi-variate Gaussian

The Muti-variate Gaussian distribution with mean vector $\mu \in \mathbb{R}^k$, and covariance matrix $\Sigma \in \mathbb{R}^{k\times k}$ has probability density

$$\text{MVarGaussian}(x \mid \mu,\ \Sigma) = \frac{1}{\sqrt{(2\pi)^k |\Sigma|}} \; e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)}$$

defined over all $x \in \mathbb{R}^k$.

B.9 Poisson

The Poisson distribution with rate λ has the probability density

$$\text{Poisson}(n \mid \lambda) = e^{-\lambda} \; \frac{\lambda^n}{n!} \; ,$$

defined over all $n \in \mathbb{Z}_{\geq 0}$.

B.10 Uniform

The Uniform distribution with parameters $a, b \in \mathbb{R}$ (and $a < b$) has the probability density

$$\text{Uniform}(x \mid a,\ b) = \frac{1}{b-a} \; \mathbb{1}_{x>a} \mathbb{1}_{x<b} \; ,$$

defined over all $x \in \mathbb{R}$.

C Supplied Python Scripts

All of the supplied scripts are briefly described in the provided `README.txt` file. Here is a more detailed description.

C.1 generate.py

This script first draws a sample from the hyperpriors. This sample represents the physics of a hypothetical earth. Next it samples a number of episodes of seismic events and detections based on this underlying physics. The episodes are divided into two sets, one for training and the other for testing, and then saved. The test episodes are further stripped of all events and associations and saved in the blind test file. The arguments to the script are as follows.

- The number of episodes to sample for each of the training and test files.
- The file name to save the physics.
- The file name to save the training data.
- The file name to save the test data.
- The file name to save the blind test data.

The following sample output is generated when the script is run. First we create a directory to save the data, and then invoke `generate.py`.

```
$> mkdir data

$> ./generate.py 100 data/physics.data data/training.data data/test.data data
   /test.blind
263 events generated
58.9 % events have at least two detections
316 events generated
57.9 % events have at least two detections

$> ls data
physics.data test.blind test.data training.data
```

The format of the `physics.data` is very simple. It simple lists each attribute of the physics in the order specified in Section 2 followed by an equal sign and the value as represented in Python. For example, the following are the first few lines of this file.

```
T = 3600
R = 6371
lambda_e = 2.44749420963e-12
mu_m = 3.0
theta_m = 4.0
gamma_m = 6.0
mu_d0 = [-15.525711059357944, -13.02388685474455, -7.7621843877940453,
   -11.133601793604004, -4.3077944888906963, -6.4595999202909073,
   -6.0893863902811285, -11.5243613488537, -10.974148036814023,
   -13.319638578401111]
```

C.2 util.py

This script has a number of utilities that are described below.

- `compute_travel_time` is the function I_T.
- `compute_slowness` is the function I_S.
- `invert_slowness` is the function I_S^{-1}.
- `write_namedtuple` writes out a Python *named tuple* to file. This is used to write the physics out to file.
- `read_namedtuple` clearly reads a Python *named tuple* from file. This function is handy to read the physics data.
- `write_episodes` writes out a list of episodes to file.
- `write_single_episode` writes out a single episode to an existing file stream.
- `read_episodes` reads a list of episodes from a file.
- `iterate_episodes` sets up a Python iterator for iterating episodes in a file.
- `compute_distance` is the *dist* function.
- `compute_azimuth` is the G_z function.
- `invert_dist_azimuth` computes a location by traveling a certain amount of distance in a given azimuth from a fixed location. This problem is also called *reckoning*. The use of this function is critical to the sample solver.
- `compute_degdiff` is the ψ function.
- `mvar_norm_fit` for fitting data to a multivariate normal.
- `mvar_norm_sample` for sampling from a multivariate normal. This function is used in `generate.py` to generate the detection probability coefficient, for example.

C.3 solve.py

This sample solver is in a very rudimentary stage as of this writing. This version cheats by looking directly at the physics rather than learning the physics from the training data. The arguments to the script are as follows.

- The name of the physics data file.
- The name of the blind data file.

- The name of the file where the solutions are to be written out.

Here is a sample command to run this script.

```
$> ./solve.py data/physics.data data/test.blind data/test.solution
```

The script keeps printing out to the terminal the events that it has located while populating the solution file as well. The solution file is flushed at the end of each episode.

A brief summary of the algorithm is as follows. For each detection we invert the slowness to get a distance estimate. Next invert the distance estimate and azimuth to get a potential event location, and finally the event time and magnitude is computed from the location. Now, repeat this process with perturbed slowness and azimuth values to get a number of candidates events from each detection. A large number of candidates, roughly a hundred from each detection, is thus generated. A candidate is assigned a score by attempting to associate the best set of detections from the available pool. An event score is the log likelihood ratio of the associated detections being generated by the event versus the same detections being generated by noise.

Once a best candidate is identified, its associated detections are removed from the pool and the process is repeated. Once the score of a candidate event is below a threshold the process stops.

Of course, this algorithm makes a number of simplifications. For example, it ignores the event prior and doesn't consider splitting or merging events. Please refer to Arora et al. (2013) for the actual deployed algorithm.

C.4 evaluate.py

This script takes only two arguments. The data file with the correct bulletin, the so-called *gold* data, and the guess data file. The guess file could have fewer episodes than the gold data. In this case only the episodes in the guess data file are evaluated. Here is a sample output from the script.

```
$> ./evaluate.py data/test.data data/test.solution
Guess data has fewer episodes than gold data!!
115 matchable events, 147 guess events, and 84 matched
Precision 57.1 % , Recall 73.0 % , F1 64.1
Time Errors mean 6.7 std 5.6
Dist Errors mean 1.4 std 0.9
Mag Errors mean 0.2 std 0.1
```

C.5 mwmatching.py

This utlity script implements a max-weight max cardinality matching algorithm that is used for the evaluation.

References

Arora, N. S., S. Russell, and E. Sudderth (2013, April). NET-VISA: Network processing vertically integrated seismic analysis. *Bulletin of the Seismological Society of America 103 no. 2A*, 709–729.

International Seismological Centre (2011). IASPEI standard phase list. http://www.isc.ac.uk/standards/phases/.

Wikipedia (2015a). atan2. http://en.wikipedia.org/wiki/Atan2. [Online; accessed 3-March-2015].

Wikipedia (2015b). Great-circle distance. http://en.wikipedia.org/wiki/Great-circle_distance. [Online; accessed 3-March-2015].

Wikipedia (2015c). Great-circle navigation. http://en.wikipedia.org/wiki/Great-circle_navigation. [Online; accessed 3-March-2015].

Wikipedia (2015d). Gutenberg-Richter law. http://en.wikipedia.org/wiki/Gutenberg-Richter_law. [Online; accessed 3-March-2015].