PPAML CP 2: Continent Scale Bird-Migration Modeling

Yusuf Bugra Erol

May 23, 2014

1 Bird Migration Problem Definition

The mathematical representation of the bird migration pattern on a $I \times J$ grid where $\xi_{t,t+1}(i,j)$'s are bird migration features, is as follows.

$$\begin{aligned} y_t &\sim \text{Poisson}(\mathbf{n}_t) \\ \theta_{t,t+1}(i,j) &= \beta^T \xi_{t,t+1}(i,j) \\ \mathbf{n}_{t,t+1}(i,:) &\sim \text{Multinomial}\left(\mathbf{n}_t(i); \theta_{t,t+1}(i,1), \ldots, \theta_{t,t+1}(i,J)\right), \forall i \in [1,I] \\ \mathbf{n}_{t+1}(j) &= \sum_i \mathbf{n}_{t,t+1}(i,j) \end{aligned} \tag{1}$$

2 Parameters

We use a uniform prior over the parameters $\beta_1, \beta_2, \beta_3, \beta_4$. The concatenation of the parameters form the parameter vector beta.

```
random Real beta1 ~ UniformReal(3, 13);
random Real beta2 ~ UniformReal(3, 13);
random Real beta3 ~ UniformReal(3, 13);
random Real beta4 ~ UniformReal(3, 13);
random RealMatrix beta(Timestep t)
        = transpose( __SCALAR_STACK(beta1 ,beta2 ,beta3 ,beta4) );
```

3 Defining the Features

Here we are implementing the following equations in BLOG.

$$\theta_{t,t+1}(i,j) = \beta^T \xi_{t,t+1}(i,j)$$
$$\left(\theta_{t,t+1}(i,1), \ldots, \theta_{t,t+1}(i,J)\right), \forall i \in [1,I]$$

```
random RealMatrix F1(Location src) ~ UniformVector(-2.5,2.5,-2.5,2.5,...
random RealMatrix F2(Location src) ~ UniformVector(-2.5,2.5,-2.5,2.5,...
random RealMatrix F3(Location src, Timestep t) ~ UniformVector(-2.5,2.5,-2.5,2.5,.
random RealMatrix F4(Location src) ~ UniformVector(-2.5,2.5, ...
// flow probabilities
random RealMatrix probs(Location src, Timestep t) =
        exp(beta(t) * vstack(F1(src),F2(src),F3(src,t),F4(src) );
```

We define features as

4 Model Dynamics

Here we are implementing the following equations in BLOG.

$$\begin{aligned}
\mathbf{n}_{t,t+1}(i,:) &\sim \text{Multinomial}\left(\mathbf{n}_t(i); \theta_{t,t+1}(i,1), \ldots, \theta_{t,t+1}(i,J)\right), \forall i \in [1, I] \\
\mathbf{n}_{t+1}(j) &= \sum_i \mathbf{n}_{t,t+1}(i,j) \qquad (2)
\end{aligned}$$

The BLOG representation of the network flow is:

```
random Integer birds(Location loc, Timestep t){
if t%20==@0 then = toInt(initial_value[loc_to_int(loc)])
    else = toInt(sum({ inflow(src, loc, Prev(t)) for Location src }))
};
// the vector of outflow from source(src) to all other locations
random RealMatrix outflow_vector(Location src,Timestep t) ~
                                    Multinomial(birds(src,t), probs(src,t)) ;
// inflow from source(src) to destination(dst)
random Integer inflow(Location src, Location dst, Timestep t) =
                             toInt(outflow_vector(src,t)[loc_to_int(dst)]);
```

- **birds(Location loc, Timestep t)**: The number of birds at Location loc and Timestep t.
  - For $t\%20 == @0$ (i.e. beginning of each year), it is equal to initial value.

    ```
    if t%20==@0 then = toInt(initial_value[loc_to_int(loc)])
    ```

  - For all the other time steps, number of birds is the sum of all birds that are flying from all sources (src) into the loc.

    ```
    else = toInt(sum({ inflow(src, loc, Prev(t)) for Location src }))
    ```

- **outflow_vector(Location src,Timestep t)** : The number of birds that flew from source (src) to all other locations, which mathematically corresponds to multinomial sampling. The elements of the vector sum up to $\mathrm{birds}(\mathrm{src}, \mathrm{t})$.

  ```
  ~ Multinomial(birds(src,t), probs(src,t)) ;
  ```

- **inflow(Location src, Location dst, Timestep t)**: The birds that flew from source (src) to destination (dst).

  ```
  = toInt(outflow_vector(src,t)[loc_to_int(dst)]);
  ```

Remark 1. *We do not support a Range type at the moment, so for indexing vectors/matrices loc which is of type Location, needs to be converted to an integer using the deterministic function loc_to_int.*

5 Observation Model

Here we are implementing the following equations in BLOG.

$$y_t \sim \text{Poisson}(\mathbf{n}_t) \tag{3}$$

The observations are defined as *Poisson* random variables. When there is no bird at a location, the corresponding observation should be zero as well. Due to random nature of particle filtering such scenarios may happen and it will lead to the collapse of the particle filter. To avoid particle filtering from collapsing, we add a small noise term to the observations.

```
random Integer NoisyObs(Location loc, Timestep t){
    if birds(loc,t) == 0 then ~ Poisson(0.01)
    else ~ Poisson(birds(loc,t))
};
```

If $\text{birds}(\text{loc}, \text{t})$ is non-zero the observations are Poisson random variables with $\lambda = \text{birds}(\text{loc}, \text{t})$, whereas when $\text{birds}(\text{src}, \text{t}) == 0$ then the observations are Poisson(0.01) instead of Poisson(0).

6 Observations

We are observing the birds only for the training period and whereas we observe the features for the whole training and testing period. The observation syntax is:

```
  obs NoisyObs(ℓ[2],@9) = 0;
```

7 Queries

We are asking for outflow_vector at each time step as well as parameter vector beta at the end of training period. The query syntax is:

```
  query outflow_vector(ℓ[6],@0);
  query outflow_vector(ℓ[11],@54);
```

8 Inference

We are using particle filtering which is a Sequential Monte Carlo (SMC) algorithm [1]. The simplest importance density is used (bootstrap filter). However, it is known that under the presence of static parameters, the particle filter degenerates quickly. To avoid such particle degeneracy we utilize the Liu-West filter which is a variant of particle filtering that adds random perturbations to the static parameter particles in such a way that the mean and the variance is preserved [2].

9 How to Run

The way to run is detailed in README.md.

10 Changes in BLOG to handle CP2

Short list of things that changed in BLOG engine to express the CP2 and do inference on it.

- Multinomial distribution that can take random expressions as inputs
- Linear algebra support for various matrix-vector operations
- TupleSet to handle expressions of the form:

  ```
  { inflow(src, loc, Prev(t)) for Location src }
  ```

- vstack for concatenating matrices/vectors
- scalar stack for concatenating reals to form vectors
- exp (exponential) function
- Arithmetic operations on Timestep type
- Solved various issues related to particle filter's uninstantiation of old time steps , and some memory problems
- Liu-West filter [2] to fight degeneracy problems in static parameter estimation via particle filtering.

References

[1] Arnaud Doucet and Adam M. Johansen. A tutorial on particle filtering and smoothing: fifteen years later. *The Oxford Handbook of Nonlinear Filtering*, pages 4–6, December 2011.

[2] Jane Liu and Mike West. Combined parameter and state estimation in simulation-based filtering. In *Sequential Monte Carlo Methods in Practice*. 2001.