MCMC strategies for Task-Specific Region Partition Problem

Hua Wei
College of IST, Penn State University

Learning Task-Specific City Region Partition

Task: Crime Prediction

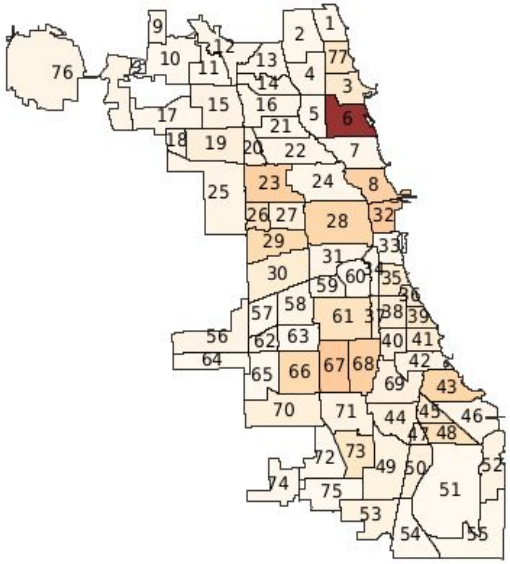
Given:

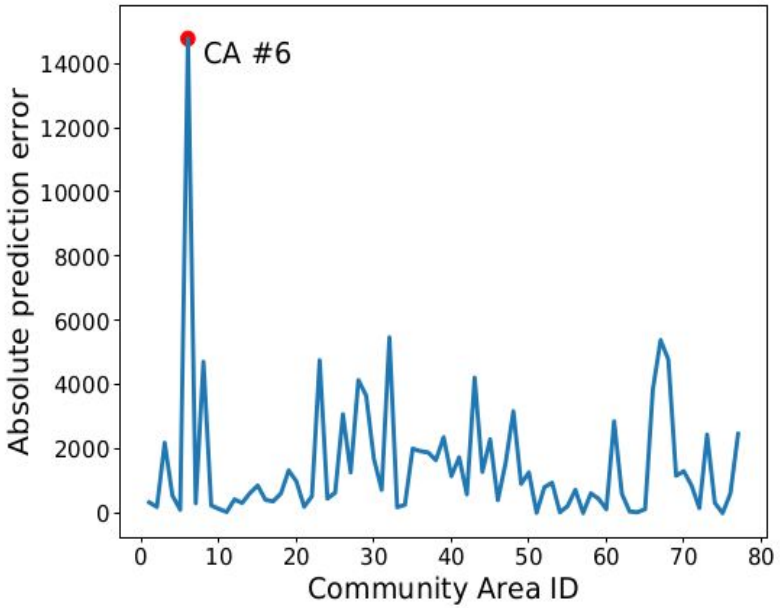
- tract features and crime number in 2010
- tract features in 2011
- Communities consist of tracts

Task:
predict crime number for every community in 2011

Easy solution:

- Aggregate tract-level data into community-level features and crime number as training data
- Fit regression model





Left: crime prediction error at community level.

Right: community #6 is an outlier.

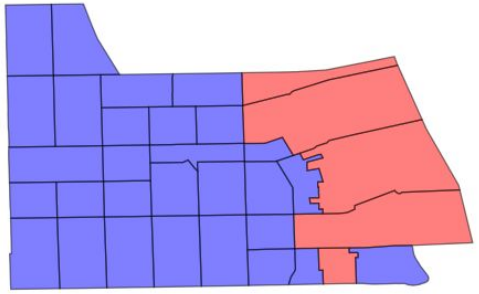
Learning Task-Specific City Region Partition

Explain the crime prediction error.

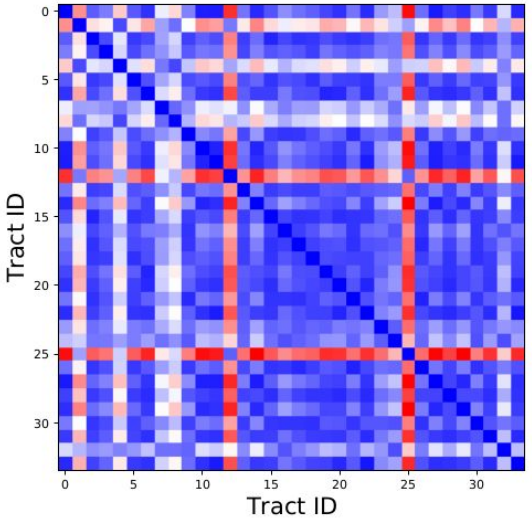
The eastside of Community #6 is different from others.

Question:

How do we obtain appropriate partitions?



(a) Census tracts of community #6.



(b) Tract similarity.

Appropriate partitions are needed!

- **Misalignment problem**: different observations are not in the same scale.
  - *Crime reports at point level, demographics at block level.*
- The existing administrative boundaries are **human-defined** and **static.**
  - *Chicago administrative boundaries are defined by university scholars and keep being used for over 100 years.*
- **Task dependent partition**: for different prediction tasks, the partition may not be the same.
  - *In State College the Beaver stadium is a hotspot for traffic but not for crime incidents.*

Problem definition: task-specific region partition

Input:

- Target property observed at tract level: ***y***.
- Other properties (e.g., demographics) available for all tracts: ***x***.
- Spatial adjacency of all tracts: **G**.

Output:

- Partition tracts into community areas with **Y** and **X**, such that

$$\arg\min_{\mathcal{Z},f} \sum_{j=1}^{m} \left(||Y_j - f(X_j)||_2 + G(Z_j) \right) \quad \text{(Eq. 1)}$$

Task prediction error

Constraint on partition, e.g. variance of population

$\mathcal{Z}$: partition. (mapping from tracts to community)

f: task function on community (regression model, etc.)

Given a fixed partition $\mathcal{Z}$, the optimal task function f can be easily learned.

The challenge lies in searching through the partition space.

A partition over City $\boldsymbol{T}$ is denoted as $\boldsymbol{Z} = \{\boldsymbol{Z1}, \boldsymbol{Z2}, \cdots, \boldsymbol{Zm}\}$, satisfying the following:
(1) (subset) $\forall \boldsymbol{j}, \boldsymbol{Zj} \subset \boldsymbol{T}$; (2) (non-overlapping) $\forall \boldsymbol{p,q}, \boldsymbol{Zp} \cap \boldsymbol{Zq} = \varnothing$; (3) (completeness) $\boldsymbol{U}\{\boldsymbol{Z\,j}\} = \boldsymbol{T}$;
(4) (spatial-continuity) $\forall \boldsymbol{j}, \boldsymbol{Pj}$ defines a polygon with exact one connected component.

This problem is challenging

1. It's difficult to calculate the maximum likelihood of objective function.
   a. Gradients on discrete variables are non available.
   b. Region properties (features and target variable) and model coefficients are high dimensional
   c. region properties change simultaneously when we change the region partition
   d. Complicated constraints should be met.
2. It is a NP-hard *combinatorial optimization* problem.
   a. Brute Force Solution: Enumerate all the possible combinations
      - O(M^N) - M is the number of tracts, N is the number of communities

Solution: Techniques like Simulated Annealing to approximate global optimization

- Approximate the global optimum with the Markov Chain Monte Carlo (MCMC) method

Markov Chain Monte Carlo (Metropolis-Hastings)

Motivation: Obtaining a sequence of random samples from a distribution for which direct sampling is difficult. It constructs a Markov chain that will ultimately converge through stochastic sampling. We only **care about the last state**.

In our case:

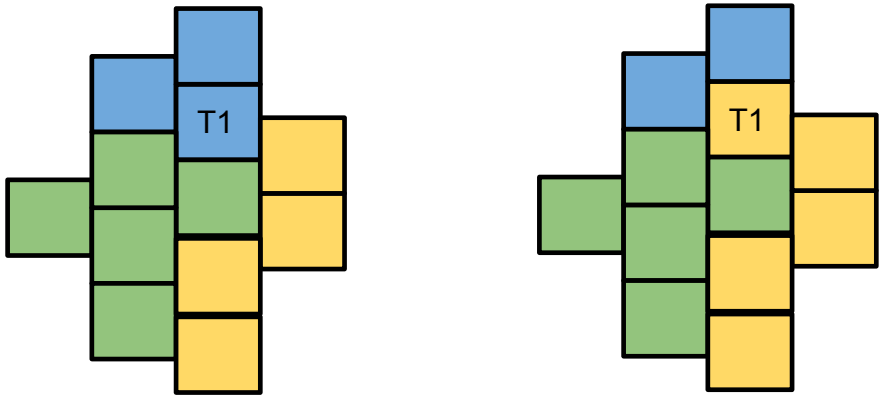
- state space is all possible partitions *Z*
- quality measure $\mathcal{F}(Z)$, a partition Z with lower value is more likely to be optimal
- proposal function $q(Z'|Z)$, defines the transition probability
- $p(Z)$ is the Boltzmann distribution $p(Z) = \frac{e^{-\mathcal{F}(Z)/T}}{P}$
- Stopping criteria:
  - Number of iterations/the standard deviation of $\mathcal{F}(Z)$

Algorithm 1 MCMC method to search $\mathcal{Z}$.

$\mathcal{Z} \leftarrow \mathcal{Z}_0$
while not satisfies stopping criteria **do**
  Sample $u \leftarrow \mathcal{U}_{[0,1]}$
  Sample $\mathcal{Z}' \leftarrow q(\mathcal{Z}'|\mathcal{Z})$
  $\gamma = \min\left[1, \frac{p(\mathcal{Z}')q(\mathcal{Z}|\mathcal{Z}')}{p(\mathcal{Z})q(\mathcal{Z}'|\mathcal{Z})}\right]$
  if $u < \gamma$ **then**
    $\mathcal{Z} \leftarrow \mathcal{Z}'$
  end if
end while

Naive MCMC

- Start from administrative boundary
- **Randomly** select one tract to **flip its assignment**.
- Decide whether to accept the new partition with training error on prediction task f.



Flip the assignment of a tract.

$$q(\mathcal{Z}'|\mathcal{Z}) = q(\mathcal{Z}|\mathcal{Z}') = \frac{1}{|\mathcal{T}_b| \cdot |Adjacent(t_i)|}$$

$$\gamma = \min[1, \frac{p(\mathcal{Z}')}{p(\mathcal{Z})}]$$

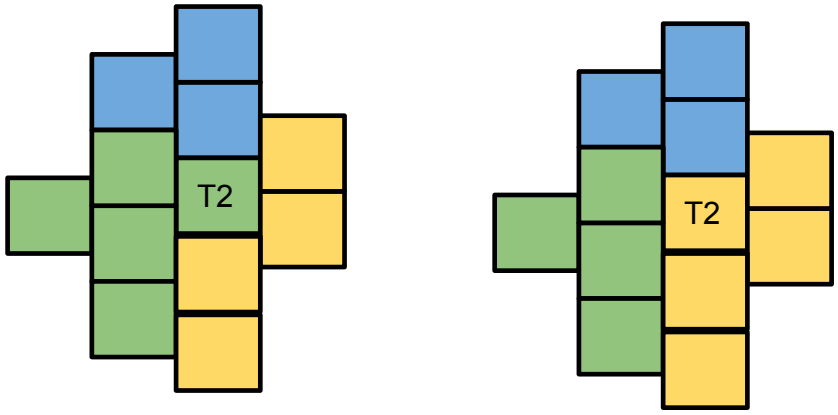
This could last a long time - since it's randomly select one tract to flip

Guided MCMC - Softmax MCMC

- Start from administrative boundary
- **Select tract from Community with the highest prediction error** (according to a softmax function), then **flip its assignment**.
- Decide whether to accept the new partition with training error on prediction task f.

$$Q(Z_j) = \frac{\exp(||Y_j - f(X_j)||_2)}{\sum_{k=1}^{m} \exp(||Y_k - f(X_k)||_2)}$$

$$q(\mathcal{Z}'|\mathcal{Z}) = \frac{Q(Z_j)}{|Z_j \cap \mathcal{T}_b| \cdot |Adjacent(t_i)|}$$

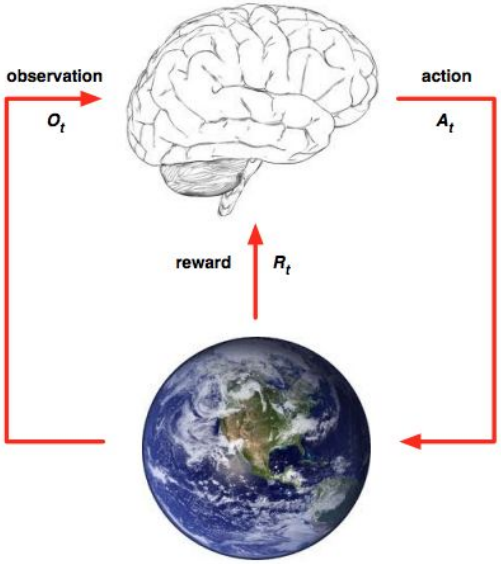


Flip the assignment of a tract *T2*.

Tract *T2* is selected because the green community has the highest prediction error.

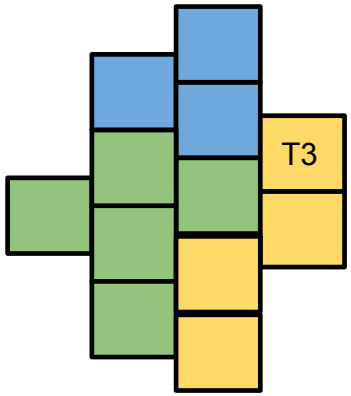
MCMC with reinforcement learning (RL)

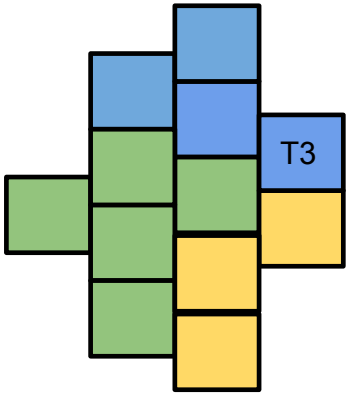
- Start from administrative boundary
- **Instead of pre-define sampling strategy, learn where to sample with RL**, then **flip its assignment**.



At each step t the agent:

- Executes action At
- Receives observation Ot
- Receives scalar reward Rt





Tract *T3* is selected because based on past experience, T3 is more likely to decrease $\mathcal{F}$ most

MCMC with reinforcement learning (RL)

- Start from administrative boundary
- **Instead of pre-define sampling strategy, learn where to sample with RL**, then **flip its assignment**.

Challenges:
- Huge state space
- Large and dynamic action space
- Training overhead is high

Learning

$$Q(\mathcal{Z}, \langle t^k, Z^k \rangle) = \Delta\mathcal{F} + \delta \cdot \sum_{a \in \{\langle t^{k+1}, Z^{k+1} \rangle\}} Q(\mathcal{Z}', a)$$

Q is a neural network
δ is set to 0

$\mathcal{Z}$: current partition state
<t, Z> : action (a tract to flip)

- Sampling: $\arg\max_{\langle t, Z \rangle} Q(\mathcal{Z}, \langle t, Z \rangle)$

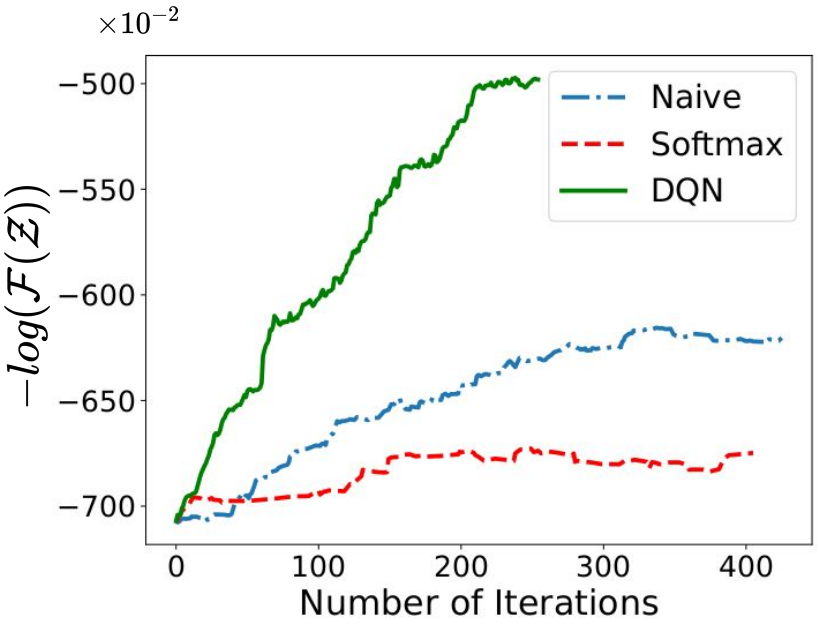
Take the max of 30 random samples

Results

Clustering first, then do prediction

Method	MAE	
	Crime	House price
Admin	1715.91	31.29
Agglomerative	72201.00	50.34
K-means	2887.83	32.40
Spectral	1440.57	29.66
Naive	1073.42(81.93)	25.73(2.76)
Softmax	1041.68(76.75)	27.13(2.98)
DQN	**746.13**(154.19)	**25.16**(1.30)

MAE of two prediction tasks.

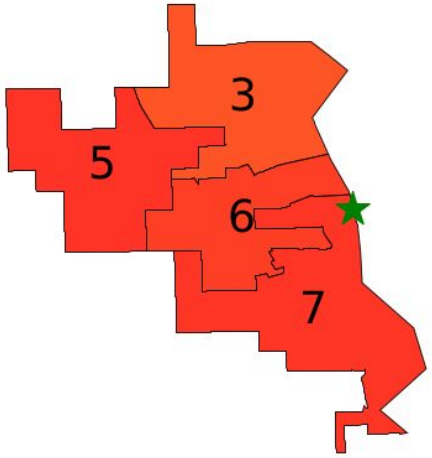


Convergence of three proposed methods.

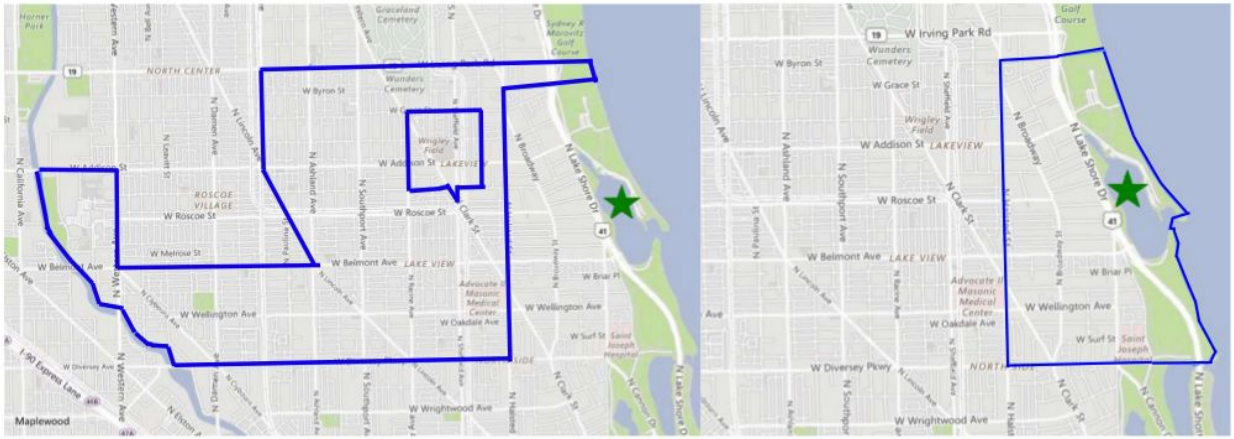
Dataset:
[1] https://data.cityofchicago.org/
[2] https://www.zillow.com/

Case study: Community #6

Community #6 from administrative boundary



DQN partition around Community #6



Zillow's self-defined region (Zillow.com: real estate website)

The green star marks the location of **Belmont Harbor** (one of Chicago's largest boating areas).

DQN community #7: contain leisure destinations such as the Lincoln Park Zoo, and numerous beach areas.

Suggestions?

Count Data Issues with Integrated Nested Laplace Approximations

Adam Walder

April 23rd, 2018

Background Information

Latent Gaussian Models consist of three layers.

- Likelihood: $\boldsymbol{y}|\boldsymbol{\eta}, \boldsymbol{\theta} \sim \prod_i p(y_i|\eta_i, \boldsymbol{\theta})$
- Latent Field: $\boldsymbol{\eta}|\boldsymbol{\theta} \sim p(\boldsymbol{\eta}|\boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \Sigma(\theta))$
- Hyper priors: $\boldsymbol{\theta} \sim p(\boldsymbol{\theta})$

A latent Gaussian field with a sparse precision matrix, Q,

$$\boldsymbol{\eta}|\boldsymbol{\theta} \sim p(\boldsymbol{\eta}|\boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, Q(\boldsymbol{\theta})^{-1})$$

is referred to as a **Gauss-Markov Random Field**.

- Benefit of sparse Q.
- Why LGMs?

INLA: Why, What and When

- **Why do we need INLA?**
  - MCMC alternative
- **What are we after with INLA?**
  - $\pi(\eta_i|\boldsymbol{y})$ and $\pi(\theta_i|\boldsymbol{y})$
- **When is INLA applicable?**
  - $|\boldsymbol{\theta}|$ is small.
  - $\boldsymbol{\eta}|\boldsymbol{\theta}$ is GMRF when dimension is high.
  - Each y_i depends on only one η_i.
- **Why is it called INLA?**
  1. Integrated: Numerical integration gives our estimates.
  2. Nested: We use an approximation to $\pi(\boldsymbol{\theta}|\boldsymbol{y})$ to get $\pi(\eta_i|\boldsymbol{y})$
  3. LA: Used in approximating $\pi(\boldsymbol{\theta}|\boldsymbol{y})$

INLA Task List

Goal: Obtain marginal distributions $\pi(\theta_i|\boldsymbol{y})$ and $\pi(\eta_i|\boldsymbol{y})$

Method: We obtain the marginals through numerical integration

$$\begin{aligned}\pi(\theta_i|\boldsymbol{y}) &\approx \sum_k \tilde{\pi}(\boldsymbol{\theta}_k^{(i)}|\boldsymbol{y}) \text{ x } \Delta(\boldsymbol{\theta}_k^{(-i)}) \\ \pi(\eta_i|\boldsymbol{y}) &\approx \sum_k \tilde{\pi}(\boldsymbol{\eta}_i|\boldsymbol{\theta}_k, \boldsymbol{y}) \text{ x } \tilde{\pi}(\boldsymbol{\theta}_k|\boldsymbol{y}) \text{ x } \Delta(\boldsymbol{\theta}_k)\end{aligned}$$

Preliminary Tasks:

1. Obtain approximation $\tilde{\pi}(\boldsymbol{\theta}|\boldsymbol{y})$
2. Explore $\log(\tilde{\pi}(\boldsymbol{\theta}|\boldsymbol{y}))$ to obtain $\Delta(\boldsymbol{\theta})$
3. Obtain approximation $\tilde{\pi}(\eta_i|\boldsymbol{\theta}, \boldsymbol{y})$

Step 1: Find $\tilde{\pi}(\boldsymbol{\theta}|\boldsymbol{y})$

The approximation for the marginal of the parameters is given by

$$\tilde{\pi}(\boldsymbol{\theta}|\boldsymbol{y}) \propto \frac{\pi(\boldsymbol{\eta}, \boldsymbol{\theta}, \boldsymbol{y})}{\pi_G(\boldsymbol{\eta}|\boldsymbol{\theta}, \boldsymbol{y})}|_{\boldsymbol{\eta}=\boldsymbol{\eta}^*(\boldsymbol{\theta})} \tag{1}$$

The Gaussian approximation in the denominator must be found as follows,

$$\tilde{\pi}(\boldsymbol{\eta}|\boldsymbol{y}, \boldsymbol{\theta}) \propto exp(-\frac{1}{2}\boldsymbol{\eta}' Q(\boldsymbol{\theta})\boldsymbol{\eta} + \sum \log(\pi(y_i|\boldsymbol{\theta}, \eta_i))) \tag{2}$$

We expand a second-order Taylor expansion about an initial modal guess $\boldsymbol{\mu}^{(0)}$. The result is

$$\log(\pi(y_i|\boldsymbol{\theta}, \eta_i)) \approx b_i(\mu_i^{(k)})\eta_i - \frac{1}{2}c_i(\mu_i^{(k)})\eta_i^2$$

Step 1: Find $\tilde{\pi}(\boldsymbol{\theta}|\boldsymbol{y})$ Cont.

To finish finding $\boldsymbol{\eta}^*(\boldsymbol{\theta})$ the following NR algorithm is used,

$$\boldsymbol{\eta}^{(k+1)} = (Q(\boldsymbol{\theta}) + diag(\boldsymbol{c}(\boldsymbol{\eta}^{(k)})))^{-1}\boldsymbol{b}(\boldsymbol{\eta}^{(k)})$$

The result produces the following Gaussian approximation to the posterior distribution

$$\pi_G(\boldsymbol{\eta}|\boldsymbol{\theta}, \boldsymbol{y}) \sim \mathcal{N}(\boldsymbol{\eta}^*(\boldsymbol{\theta}), [Q(\boldsymbol{\theta}) + \text{diag}(\boldsymbol{c}(\boldsymbol{\eta}^*(\boldsymbol{\theta})))]^{-1})$$

Note: Still have a *GMRF*.
Note: The algorithm involves iteratively solving a system.

STEP 2: EXPLORE $\log(\tilde{\pi}(\boldsymbol{\theta}|\boldsymbol{y}))$

(1) Find mode:

$\boldsymbol{\theta}^* = \text{argmax}_{\boldsymbol{\theta}} \log(\tilde{\pi}(\theta|\boldsymbol{y}))$

(2) z-parameterization

$\Sigma = H^{-1} = V\Lambda^{1/2}V'$

Define: $\boldsymbol{\theta}(\boldsymbol{z}) = \boldsymbol{\theta}^* + V\Lambda^{1/2}\boldsymbol{z}$

(3) Define δ_z and δ_π. Let k = dim($\boldsymbol{\theta}$)

for(i in 1:k)

$\boldsymbol{z} = (0, , .., 0) \qquad z_i = \delta_z$

while($|\log(\tilde{\pi}(\boldsymbol{\theta}(\mathbf{0})|\boldsymbol{y}) - \log(\tilde{\pi}(\boldsymbol{\theta}(\boldsymbol{z})|\boldsymbol{y})| < \delta_\pi$)

store $\boldsymbol{\theta}(\boldsymbol{z})$ and $\log(\tilde{\pi}(\boldsymbol{\theta}(\boldsymbol{z})|\boldsymbol{y})$

$z_i = z_i + \delta_z$

Repeat for the opposite direction.

We also need to compute all the intermediate combinations. In dimensions > 2, Box and Behnken design is used.

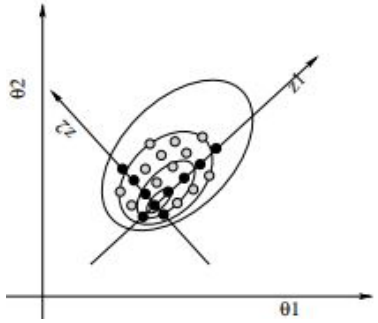
STEP 3: APPROXIMATE $\pi(\eta_i|\boldsymbol{\theta}, \boldsymbol{y})$ AND INTEGRATE

We can approximate the density as follows

$$\tilde{\pi}(\eta_i|\boldsymbol{\theta}, \boldsymbol{y}) \sim \mathcal{N}(\mu_i(\boldsymbol{\theta}), \sigma_i^2(\boldsymbol{\theta}))$$

- We already have $\mu_i(\boldsymbol{\theta})$ from step 2.
- We only need to obtain the marginal variances.
- Other options are LA and SLA.

Complete the Integration



Model of Interest

Generalized Latent Gaussian Model

$$\begin{aligned} y_i &\sim \text{Poisson}(\lambda_i) \\ \log(\lambda_i) &\sim x_i\beta + \eta_i \\ L\boldsymbol{\eta} &\sim \mathcal{N}(\mathbf{0}, \sigma_\eta^2 \boldsymbol{I}) \end{aligned}$$

The priors, $\boldsymbol{\theta} = (\beta, \sigma_\eta^2)$, are given by

$$\begin{aligned} \beta &\sim \mathcal{N}(0, \sigma_\beta^2) \\ \sigma_\eta^2 &\sim I.G.(u, v) \end{aligned}$$

- L = Q + κ^2I.
- Q is a finite difference matrix on a 2-D unit square.

Note: L*L is sparse, so we are dealing with a **GMRF**.

What Goes Wrong?

- The algorithm fails in the first step.
- We can not obtain a Gaussian approximation to the denominator.

$$\tilde{\pi}(\boldsymbol{\theta}|\boldsymbol{y}) \propto \frac{\pi(\boldsymbol{\eta}, \boldsymbol{\theta}, \boldsymbol{y})}{\pi_G(\boldsymbol{\eta}|\boldsymbol{\theta}, \boldsymbol{y})}|_{\boldsymbol{\eta}=\boldsymbol{\eta}^*(\boldsymbol{\theta})}$$

The NR algorithm is very sensitive to the initial points of optim.

Recall: NR requires iterating until convergence

$$\boldsymbol{\mu}^{(k+1)} = [\frac{1}{\sigma_\eta^2} L^2 + \text{diag}(\boldsymbol{c}^{(k)})]^{-1} \boldsymbol{b}^{(k)}$$

$$b_i^{(k)} = y_i - exp(x_i\beta + \mu_i^{(k)}) \quad \text{and} \quad c_i^{(k)} = -exp(x_i\beta + \mu_i^{(k)})$$

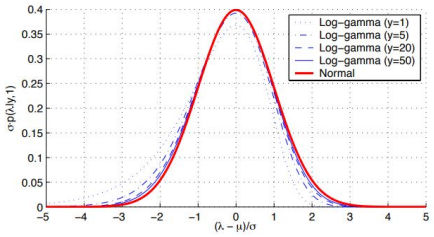
Normal Approximation to the Log-Gamma Distribution

The Log-Gamma distribution

$$\pi(v|a,b) = \frac{1}{\Gamma(a)b^a} e^{va} e^{-\frac{e^v}{b}} \approx \mathcal{N}(\log(ab), a^{-1})$$

We can now write rewrite $\pi(\boldsymbol{y}|\boldsymbol{\theta},\boldsymbol{\eta})$ as follows,

$$[\boldsymbol{y}|\boldsymbol{\lambda}] = \prod \frac{1}{y_i!}\lambda_i^{y_i} e^{-\lambda_i} = \prod \frac{1}{y_i}[\frac{1}{(y_i-1)!} e^{\log(\lambda_i)y_i} e^{-e^{\log(\lambda_i)}}] \approx \prod \frac{1}{y_i}\mathcal{N}(\log(y_i), y_i^{-1})$$



Adjusted Gaussian Approximation

The Gaussian approximation is now,

$$\begin{aligned}\tilde{\pi}(\boldsymbol{\eta}|\boldsymbol{y},\boldsymbol{\theta}) &\propto \exp(-\frac{1}{2}\boldsymbol{\eta}'Q(\boldsymbol{\theta})\boldsymbol{\eta} + \sum_i \log(\pi(y_i|\eta_i,\boldsymbol{\theta}))) \\ &\propto \exp(-\frac{1}{2}\boldsymbol{\eta}'Q(\boldsymbol{\theta})\boldsymbol{\eta} + \sum_i \log(\mathcal{N}_{\log(\lambda_i)}(\log(y_i), y_i^{-1})))\end{aligned}$$

Now we have

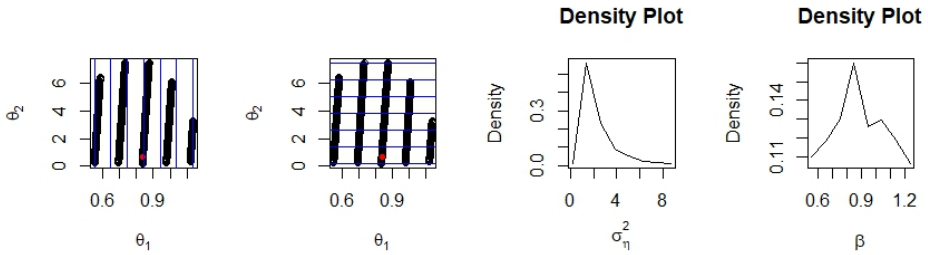
$$\tilde{\pi}(\boldsymbol{\eta}|\boldsymbol{y},\boldsymbol{\theta}) \sim \mathcal{N}(Q^*(\boldsymbol{\theta})^{-1}(\log(\boldsymbol{y}) - \boldsymbol{x}'\beta), Q^*(\boldsymbol{\theta})^{-1})$$

Where,

$$Q^*(\boldsymbol{\theta}) = \frac{1}{\sigma_\eta^2}L^2 + \text{diag}(\boldsymbol{y})$$

What was accomplished?

RESULTS



$\hat{\beta}$	95% CI: β	$\hat{\sigma}^2_\eta$	95% CI: σ^2_η
0.8479	(0.8896, 0.9314)	2.3889	(1.9000, 2.8778)

Concluding Thoughts

- **Topics Unmentioned**
  - INLA package, Issues extending to $|\theta| > 2$
- **Criticisms**
  - Lack of Clarity by authors
- **Caveat of "blackboxing"**
  - No justification/intuition for subtle decisions
- **Future Work**
  - Interested in how they chose δ_π, find new interpolant.

Kernel Adaptive Metropolis-Hastings:

Gaussian Process Classification using Pseudo-marginal MCMC

Nicholas Sterge

April 22, 2018

Adaptive Metropolis-Hastings

Metropolis-Hastings:

- Propose $\theta^* \sim q(\cdot|\theta)$
- Accept θ^* w.p. $\min\left\{1, \frac{\pi(\theta^*)q(\theta|\theta^*)}{\pi(\theta)q(\theta^*|\theta)}\right\}$

What is the best proposal, $q(\cdot|\theta)$?

- Adaptive M-H: learn covariance structure of target, adapt proposal accordingly
- Gaussian w/ covariance learned from chain history, $\mathbf{q_t}(\cdot|\theta^{(\mathbf{t})}) = \mathcal{N}(\cdot|\theta^{(\mathbf{t})}, \nu^{\mathbf{2}}\mathbf{\Sigma_t})$ Haario et al. (1999)
- Adaptive scaling, principal component updates Andrieu and Thoms (2008)

Adaptive Metropolis-Hastings

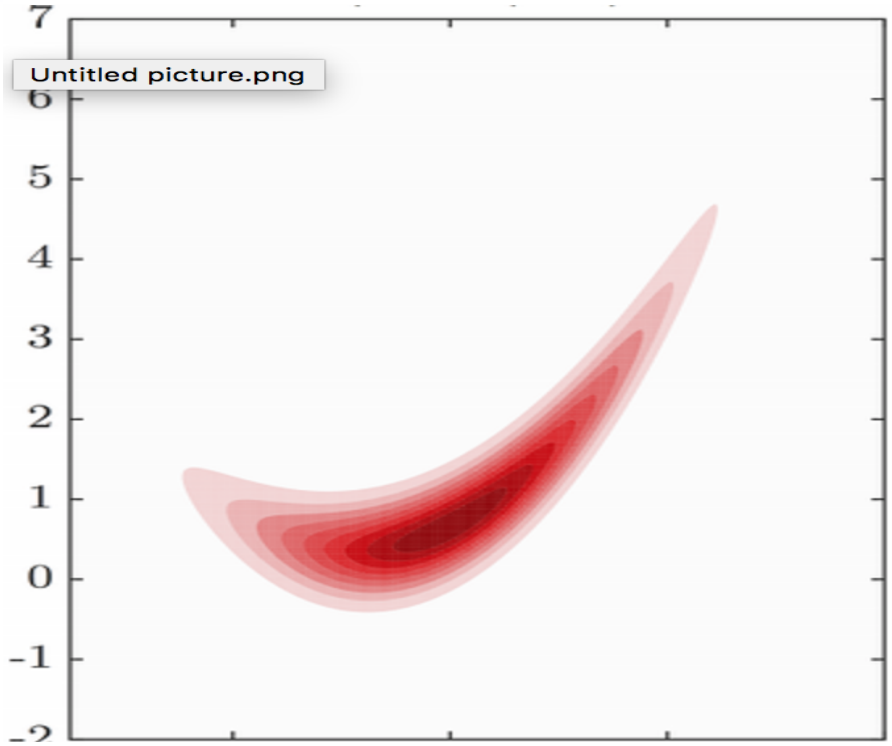
Why adapt?

- Limited information about target
- Multiple proposals to tune, e.g. Section 8 of Haario et al. (1999) (GOMOS)
- Adaptive burn-in for Metropolis algorithm

Shortcomings:

- Strongly non-linear targets, e.g. Banana target of Haario et al. (1999), Flower target of Sejdinovic et al. (2014)
- Directions of large variance depend on location of sampler

Banana



Kernel Adaptive Metropolis-Hastings

Motivation:

Principal Component Proposals

- Estimate $\Sigma_{\mathbf{z}}$ from subset of chain history, $\mathbf{z}$
- Eigenvectors/eigenvalues inform proposal, i.e. mixture of random walks down principal eigendirections

Kernelize $\rightarrow$ Kernel Principal Component Proposals

- Using kernel PCA, nonlinear principal directions can inform our proposal

Kernel Adaptive Metropolis-Hastings

Idea: Nonlinear support of target may be learned using Kernel PCA

- RKHS of functions, $\mathcal{H}$, $f : \mathcal{X} \to \mathbb{R}$, with RK $k(\cdot, \cdot)$
- Probability measure $\mathbb{P}$ on $\mathcal{X}$, covariance operator $C_{\mathbb{P}}$, empirical covariance operator $C_{\mathbf{z}}$ ($\mathbf{z} = \{z_i\}_{i=1}^n$)
- Kernel PCA is linear PCA on the covariance operator $C_{\mathbf{z}}$ [Schölkopf et al. (1998)]
- Principal eigendirections will be non-linear functions

How do we use Kernel PCA to construct a proposal?

1. Mixture of random walks down principal eigendirections
2. Consider the Gaussian measure on $\mathcal{H}$ induced by $C_{\mathbf{z}}$

Constructing the Proposal

Suppose current state y, subset of chain history $\mathbf{z} = \{z_i\}_{i=1}^n$

- Gaussian measure on $\mathcal{H}$:

$$\mathcal{N}(f|k(\cdot,y),\nu^2 C_{\mathbf{z}}) \propto \exp\left\{\frac{-1}{2\nu^2}\big(f-k(\cdot,y)\big)^T C_{\mathbf{z}}^{-1}\big(f-k(\cdot,y)\big)\right\}$$

- Samples have form $f = k(\cdot,y) + \sum_{i=1}^n \beta_i k(\cdot,z_i)$, where $\beta \sim \mathcal{N}(0, \frac{\nu^2}{n}\mathbf{I}_n)$

Moving from $\mathcal{H}$ to $\mathcal{X}$

- Obtain sample $f \in \mathcal{H}$, want $x \in \mathcal{X}$ s.t. $k(\cdot,x)$ 'near' f, i.e. $x = \arg\min \|k(\cdot,x) - f\|_{\mathcal{H}}^2$
- Minimization expensive $\rightarrow$ one iteration of gradient descent

Constructing the Proposal

Closed form:

$$q_{\mathbf{z}}(\cdot|\mathbf{y}) = \mathcal{N}(\cdot|y, \gamma^2 \mathbf{I}_n + \nu^2 \mathbf{M}_{\mathbf{z},y} \mathbf{H} \mathbf{M}_{\mathbf{z},y}^T)$$

where γ is a GD parameter, $\mathbf{H}$ centering matrix,
$\mathbf{M}_{\mathbf{z},y} = 2(\nabla_x k(x, z_1)|_{x=y}, ...)$

Update Schedule and Convergence:

- Proposal updated each time we update $\mathbf{z}$
- Adaptation probabilities $\{p_t\}_{t=1}^{\infty}$ s.t. $p_t \to 0$ and $\sum_t p_t = \infty$, guarantee convergence to correct target

Properties:

- Only requires evaluation of unnormalized target, locally adaptive in input space $\mathcal{X}$

Proposal Algorithm

At iteration $t+1$, current state is x_t

1. With probability p_t update random subsample, $\mathbf{z}$, of chain history
2. Sample proposed x^* from $q_{\mathbf{z}}(\cdot|x_t) = \mathcal{N}(\cdot|x_t, \gamma^2\mathbf{I}_n + \nu^2\mathbf{M}_{\mathbf{z},x_t}\mathbf{H}\mathbf{M}_{\mathbf{z},x_t}^T)$
3. Accept/Reject with M-H probability

$$x_{t+1} = x^* \quad w.p. \min\left\{1, \frac{\pi(x^*)q_{\mathbf{z}}(x_t|x^*)}{\pi(x_t)q_{\mathbf{z}}(x^*|x_t)}\right\}$$

$$x_{t+1} = x_t \quad w.p. 1 - \min\left\{1, \frac{\pi(x^*)q_{\mathbf{z}}(x_t|x^*)}{\pi(x_t)q_{\mathbf{z}}(x^*|x_t)}\right\}$$

Gaussian Process Classification

Inputs: $\mathbf{X} = \{x_1, ..., x_n\}$, $x_i \in \mathbb{R}^d$

Labels: $\mathbf{Y} = \{y_1, ..., y_n\}$, $y_i \in \{\pm 1\}$

Latent: $\mathbf{f} = \{f_1, ..., f_n\}$, $\mathbf{f} \sim \mathcal{N}(\mathbf{0}, K(\theta))$

$$\theta = \{\sigma, \tau_1^2, ...\tau_d^2\}, \quad K(\theta)_{ij} = \sigma \exp\left(\frac{-1}{2}(x_i - x_j)^T A(x_i - x_j)\right)$$

$$A^{-1} = diag\left(\tau_1^2, ..., \tau_d^2\right)$$

$$p(y_i = 1|f_i) = \Phi(y_i f_i)$$

Gaussian Process Classification

Prediction:

$$p(y_* = 1|\mathbf{Y}, \theta) = \int p(y_* = 1|f_*)p(f_*|\mathbf{f}, \mathbf{Y})p(\mathbf{f}, \theta|\mathbf{Y})df_* d\mathbf{f} d\theta$$

MCMC to approximate integration w.r.t. $p(\mathbf{f}, \theta|\mathbf{Y})$

$$p(y_* = 1|\mathbf{Y}, \theta) \approx \frac{1}{N}\sum_{i=1}^{N} \int p(y_*|f_*)p(f_*|\mathbf{f}^{(i)}, \theta^{(i)})df_*$$

Pseudo-marginalization

Elliptical slice sampling to draw $\mathbf{f}$ from $p(\mathbf{f}|\mathbf{Y}, \theta)$

Pseudo-marginal MCMC to draw θ from $p(\theta|\mathbf{Y})$

$$\tilde{z} = \frac{\tilde{p}(\mathbf{Y}|\theta^*)p(\theta^*)q_{\mathbf{z}}(\theta|\theta^*)}{\tilde{p}(\mathbf{Y}|\theta)p(\theta)q_{\mathbf{z}}(\theta^*|\theta)}$$

where

$$\tilde{p}(\mathbf{Y}|\theta) = \frac{1}{N_{imp}} \sum_{i=1}^{N_{imp}} \frac{p(\mathbf{Y}|\mathbf{f}_i)p(\mathbf{f}_i|\theta)}{r(\mathbf{f}_i|\mathbf{Y}, \theta)}$$

$r(\cdot|\mathbf{Y}, \theta)$ is importance function obtained via Laplace Approximation

C. Andrieu and J. Thoms. A tutorial on adaptive mcmc. *Statistics and Computing*, 18(4):343–373, 2008.

H. Haario, E. Saksman, and J. Tamminen. Adaptive proposal distribution for random walk metropolis. *Comput. Stat.*, 14(3): 375–395, 1999.

B. Schölkopf, A. Smola, and K. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

Dino Sejdinovic, Heiko Strathmann, Maria Lomeli Garcia, Christopher Andrieu, and Arthur Gretton. Kernel adaptive metropolis-hastings. In *Proceedings of the 31st International Conference on Machine Learning*, Beijing, China, 2014.