Comparing priors over binary matrices within latent feature models framework

Supervisor: Lorenzo Ghilotti

V. Iapaolo, L. Maci, Z. Mehboob, V. Montefusco,
G.T. Morra, A. Napolitano

Department of Mathematical Engineering
Politecnico di Milano

Group project, 9 January 2024

The Beta-Bernoulli model: Gibbs Sampler

The aim is to sample binary matrices Z according to the Beta-Bernoulli Model.
Let $A \in \mathbb{R}^{\widetilde{N}xD}$,$X \in \mathbb{R}^{NxD}$,$\alpha < 0$,$\theta > -\alpha$:

Algorithm 1: GibbsSamplerbetabernoulli

Data: $X, A, \sigma_x, \sigma_A, BurnInIterations, Iterations$
Result: collection of Z matrices
for $iter \leftarrow 1$ **to** $BurnInIterations + Iterations$ **do**
    for $i \leftarrow 1$ **to** N **do**
        $K \leftarrow$ number of non-empty columns of Z_{-i};
        for $j \leftarrow 1$ **to** K **do**
            Sample already observed features;
        end
        Sample new features;
    end
    Compute $E[A|X, Z]$ and log[P(X,Z)] $\leftarrow$Eq. (12)+(21)
    if $iter > BurnInIterations$ **then**
        Add Z to the collection;
    end
end

Sampling observed features:update $z_{i,j}$

Let K be the number of non-null columns of Z_{-i}. Let $m \in \mathbb{R}^K$ be the vector that tells us how many subjects (except i) have feature j, $j \leq K$

Algorithm 2: Sample observed features

Data: $X, A, Z, \sigma_x, \sigma_A, i, j, m_j$
Result: Z

$z_{i,j} \leftarrow 1$;
$P(X|Z) \leftarrow$ see eq. (21);
$P(z_{ij} = 1 \,|\, z_{-i}) \leftarrow \frac{m_j - \alpha}{\theta + (n-1)}$;

$z_{i,j} \leftarrow 0$;
$P(X|Z) \leftarrow$ see eq. (21);
$P(z_{ij} = 0 \,|\, z_{-i}) \leftarrow 1 - P(z_{ij} = 1 \,|\, z_{-i})$;

$P(z_{il} = 1 \,|\, X, Z_{-i,j}) \propto P(X|Z) \cdot P(z_{ij} = 1 \,|\, z_{-i})$;
$P(z_{il} = 0 \,|\, X, Z_{-i,j}) \propto P(X|Z) \cdot P(z_{ij} = 0 \,|\, z_{-i})$;
Normalize the probabilites;

$prob\_param \leftarrow P(z_{il} = 1 \,|\, X, Z_{-i,j})$;
$z_{ij} \sim \text{Bernoulli}(prob\_param)$

Sampling of new features

Let w be the number of new features observed by subject i. In this step we sample w:

Algorithm 3: Sample new features

Data: $X, A, Z, \sigma_X, \sigma_A, i$
Result: Z
$W \leftarrow \widetilde{N} - K$;
$prob\_param = 1 - \frac{\theta+\alpha+n-1}{\theta+n-1}$;
Initialize **prob_vector** of dimension W to 0;
for $h \leftarrow 0, 1, \ldots, W$ **do**
    $prob\_vector_h \leftarrow BinomialProbability(W, prob\_param, K = h)$;
    update Z;
    Calculate $P(X|Z)$;
    $P(w = h \,|\, X, Z) \propto P(X|Z) \cdot prob\_vector_h$;
end
Normalize **prob_vector**;
Sample $w \sim \text{Discrete}(\textbf{prob\_vector})$;
Update Z;

The Indian Buffet Process: Gibbs Sampler

The aim is to sample binary matrices Z according to the IBP.
Let $A \in \mathbb{R}^{KxD}$,$X \in \mathbb{R}^{NxD}$ (K unbounded), $0 \leq \alpha \leq 1$,$\theta \geq \alpha$,$\gamma \geq 0$:

Algorithm 4: GibbsSamplerIBP

Data: $X, A, \sigma_x, \sigma_A, BurnInIterations, Iterations$
Result: collection of Z matrices
for $iter \leftarrow 1$ **to** $BurnInIterations + Iterations$ **do**
  for $i \leftarrow 1$ **to** N **do**
    $K \leftarrow$ number of non-empty columns of Z_{-i};
    for $j \leftarrow 1$ **to** K **do**
      Sample already observed features;
    end
    Sample new features, IBP;
  end
  Compute $E[A|X, Z]$ and log[P(X,Z)] $\leftarrow$Eq. (12)+(21)
  if $iter > BurnInIterations$ **then**
    Add Z to the collection;
  end
end

IBP:Sampling new features

Let w be the number of new features observed by subject i. In this step we sample w:

Algorithm 5: Sample new features,IBP

Data: $X, A, Z, \sigma_x, \sigma_A, \alpha, \theta, \gamma$
Result: Z

$W \leftarrow \text{UB} - K$; // $UB = 10$ if $K > 20$;5 otherwise
$prob\_param = \gamma \cdot \frac{(\theta+\alpha)_n}{(\theta+1)_n}$;
Initialize **prob_vector** of dimension W to all 0;
for $h \leftarrow 0, 1, \ldots, W$ **do**
    $prob\_vector_h \leftarrow$ PoissonProbability($prob\_param, k = h$);
    Update Z;
    Calculate $P(X|Z)$;
    $P(w = h \,|\, X, Z) \propto P(X|Z) \cdot prob\_vector_h$;
end
Normalize **prob_vector**;
Sample $w \sim$ Discrete(**prob_vector**);
Update Z;

Metropolis-Hastings Step for σ_x

Algorithm 6: Metropolis-Hastings step for updating σ_x

Data: $Z, X, A, \sigma_A, \sigma_x^{(\text{current})}$, proposal variance, prior variance
Result: Updated σ_x
Generate proposal $\sigma_x^{(\text{new})}$ from $\mathcal{N}(\sigma_x^{(\text{current})}, \sqrt{\text{proposal variance}})$;
if $\sigma_x^{(new)} \leq 0$ **then**
  return $\sigma_x^{(\text{current})}$; ; // Ensure proposed σ_x is positive
end
Compute $M^{(\text{current})}$ and $M^{(\text{new})}$ matrices;
Calculate log-likelihoods, $\log P(X|Z, \sigma_x^{(\text{current})}, \sigma_A)$, with $\sigma_x^{(\text{current})}$;
Calculate log-likelihoods, $\log P(X|Z, \sigma_x^{(\text{new})}, \sigma_A)$, with $\sigma_x^{(\text{new})}$;
Calculate log-priors, $\log \left(e^{-\frac{1}{2} \frac{\sigma_x^{(\text{current})^2}}{\text{prior variance}}} \right)$, for current $\sigma_x^{(\text{current})}$;
Calculate log-priors $\log \left(e^{-\frac{1}{2} \frac{\sigma_x^{(\text{new})^2}}{\text{prior variance}}} \right)$ for new $\sigma_x^{(\text{new})}$;
Compute log acceptance ratio $\log \left(\frac{\text{new\_log\_likelihood} \times \text{new\_log\_prior}}{\text{current\_log\_likelihood} \times \text{current\_log\_prior}} \right)$;
Sample from uniform distribution;
if $\log(\textit{uniform}()) < \log(\textit{acceptance ratio})$ **then**
  return $\sigma_x^{(\text{new})}$;
else
  return $\sigma_x^{(\text{current})}$;
end

Metropolis-Hastings Step for σ_a

Algorithm 7: Metropolis-Hastings step for updating σ_a

Data: $Z, X, A, \sigma_x, \sigma_a^{(\text{current})}$, proposal variance, prior variance
Result: Updated σ_a
Generate proposal $\sigma_a^{(\text{new})}$ from $\mathcal{N}(\sigma_a^{(\text{current})}, \sqrt{\text{proposal variance}})$;
if $\sigma_a^{(new)} \leq 0$ **then**
    return $\sigma_a^{(\text{current})}$; ;   `// Ensure proposed` σ_a `is positive`
end
Compute $M^{(\text{current})}$ and $M^{(\text{new})}$ matrices;
Calculate log-likelihoods, $\log P(X|Z, \sigma_a^{(\text{current})}, \sigma_x)$, with $\sigma_a^{(\text{current})}$;
Calculate log-likelihoods, $\log P(X|Z, \sigma_a^{(\text{new})}, \sigma_x)$, with $\sigma_a^{(\text{new})}$;
Calculate log-priors, $\log\left(e^{-\frac{1}{2}\frac{\sigma_a^{(\text{current})2}}{\text{prior variance}}}\right)$, for $\sigma_a^{(\text{current})}$;
Calculate log-priors, $\log\left(e^{-\frac{1}{2}\frac{\sigma_a^{(\text{new})2}}{\text{prior variance}}}\right)$, for $\sigma_a^{(\text{new})}$;
Compute log acceptance ratio $\log\left(\frac{\text{new\_log\_likelihood} \times \text{new\_log\_prior}}{\text{current\_log\_likelihood} \times \text{current\_log\_prior}}\right)$;
Sample from uniform distribution;
if $\log(\textit{uniform}()) < \log(\textit{acceptance ratio})$ **then**
    return $\sigma_a^{(\text{new})}$;
else
    return $\sigma_a^{(\text{current})}$;
end

Updating σ_x and σ_a within the Gibbs Sampler

During each iteration of the Gibbs Sampler, the hyperparameters σ_x and σ_a are updated. This is achieved by calling the Metropolis-Hastings update functions described in the previous slides.

- The proposal variance for σ_x is calculated as $0.1 \times \sigma_x$.
- The proposal variance for σ_a is calculated as $0.1 \times \sigma_a$.
- The function `metropolis_step_sigma_x()` updates σ_x based on the calculated proposal variance and the prior variance.
- Similarly, `metropolis_step_sigma_a()` updates σ_a.

These update steps are a crucial part of ensuring the Gibbs Sampler explores the parameter space effectively.

The Beta-Bernoulli model: sampling matrix A

We add the update of the matrix A:

Algorithm 8: GibbsSamplerbetabernoulli

Data: $X, A, \sigma_x, \sigma_A, \mathit{BurnInIterations}, \mathit{Iterations}$
Result: collection of Z matrices
for $\mathit{iter} \leftarrow 1$ **to** $\mathit{BurnInIterations} + \mathit{Iterations}$ **do**
    for $i \leftarrow 1$ **to** N **do**
        $K \leftarrow$ number of non-empty columns of Z_{-i};
        for $j \leftarrow 1$ **to** K **do**
            Sample already observed features;
        end
        Sample new features;
    end
    Update σ_x;
    Update σ_a;
    Update A ;
    Compute E[$A|X, Z$] and log[P(X,Z)] $\leftarrow$Eq. (12)+(21)
    if $\mathit{iter} > \mathit{BurnInIterations}$ **then**
        Add Z to the collection;

Gaussian Prior for A

We then generalized this model w.r.t. different possible priors of A.
At each iteration of the Gibbs Sampler, we update A and sample from it.
Let $\tau = \frac{1}{\sigma^2_{A_{n+1}}}$:

Algorithm 9: Update A

Data: Z, X, σ_x, σ_A
Result: updated A
$\tau = \frac{1}{\sigma^2_{A_n}} I_k + \frac{1}{\sigma^2_X} Z^T Z$;
$\sigma^2_{A_{n+1}} = \frac{1}{\tau}$; // posterior variance of A;
$\mu_{A_{n+1}} = \sigma^2_{A_{n+1}} Z^T X \frac{1}{\sigma^2_x}$; // posterior mean of A;
for $k \leftarrow 0, 1, \ldots, K$ **do**
  for $d \leftarrow 0, 1, \ldots, D$ **do**
    $A_{k,d} \sim \mathcal{N}(\mu_{A_{n+1}}(k, d), \sigma_{A_{n+1}}(k, k))$ // updated values of A
  end
end

Alternative Prior for A

Now we consider the following prior for the i-th row of A:

$$A_i \mid \mu_i, \sigma^2 \sim \mathcal{N}(\mu_i, \sigma^2),$$

$$\text{with } \mu_i \sim \mathcal{N}(0, c\,\sigma^2) \text{ and } \sigma^2 \sim invgamma(a, b)$$

Algorithm 10: Update A and σ_A

Data: Z, X, a, b, c
Result: updated A
Update a and b;
$\sigma^2 \sim invgamma(a, b)$; // posterior variance
for $d \leftarrow 0, 1, \dots, D$ **do**
  $\mu_d \sim \mathcal{N}(0, c\,\sigma^2)$; // posterior mean, with c: multiplicative constant
end
for $k \leftarrow 0, 1, \dots, K$ **do**
  for $d \leftarrow 0, 1, \dots, D$ **do**
    $A_{k,d} \sim \mathcal{N}(\mu_d, \sigma^2)$; // updated values of A
  end
end

Domande

- Aggiungere qua le domande
- Come scegliere i valori iniziali di a e b
- Come scegliere c
- In generale, è corretto usare Metropolis-Hastings per l'aggiornamento di σ_x e σ_a, oppure è possibile fare sampling dalle full-conditionals $P(\sigma_x|\text{resto})$ e $P(\sigma_a|\text{resto})$ e fare update con Gibbs sampler? Oppure posso ricavare una distribuzione nota della posterior dal prodotto likelyhood*prior?
- Se Metropolis-Hastings è appropriato, assumo come prior su σ_x e σ_a una normale con media zero e varianza 1?
- Come metodo per generare la proposal sigma, è corretto quello indicato nell'algoritmo?