Bayesian graph convolutional neural networks for semi-supervised classification (AAAI2019)

2019.5.30

GCN(Kipf and Welling 2017)

$$\mathbf{H}^{(1)} = \sigma(\mathbf{A}_{\mathcal{G}}\mathbf{X}\mathbf{W}^{(0)})$$
$$\mathbf{H}^{(l+1)} = \sigma(\mathbf{A}_{\mathcal{G}}\mathbf{H}^{(l)}\mathbf{W}^{(l)})$$

Bayesian neural networks

$$\mathbf{X} = \{x_1, ..., x_n\} \quad \mathbf{Y} = \{y_1, ..., y_n\} \qquad y = f(x)$$

Since W is not deterministic, the output of neural network is also a random variable. Prediction for a new input X can be formed by integrating with respect to the posterior distribution of W as follows:

$$p(y|x, \mathbf{X}, \mathbf{Y}) = \int p(y|x, W)p(W|\mathbf{X}, \mathbf{Y})\, dW$$

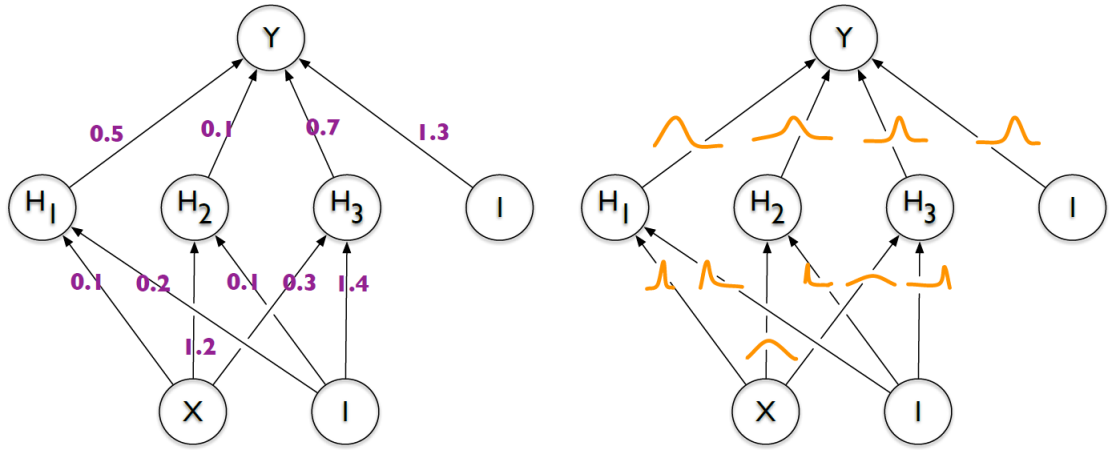
$$p(y|x, \mathbf{X}, \mathbf{Y}) = \int p(y|x, W)p(W|\mathbf{X}, \mathbf{Y})\, dW$$

The integral is in general intractable.

Markov Chain Monte Carlo

Variational inference

Expectation propagation



Deep Bayesian Active Learning with Image Data (NIPS2017)

Max Entropy $$\mathbb{H}[y|\mathbf{x}, \mathcal{D}_{\text{train}}] := -\sum_c p(y=c|\mathbf{x}, \mathcal{D}_{\text{train}}) \log p(y=c|\mathbf{x}, \mathcal{D}_{\text{train}})$$

$$\mathbb{I}[y, \boldsymbol{\omega}|\mathbf{x}, \mathcal{D}_{\text{train}}] = \mathbb{H}[y|\mathbf{x}, \mathcal{D}_{\text{train}}] - \mathbb{E}_{p(\boldsymbol{\omega}|\mathcal{D}_{\text{train}})}\left[\mathbb{H}[y|\mathbf{x}, \boldsymbol{\omega}]\right]$$

Choose pool points that are expected to maximize the information gained about the model parameters: maximise the mutual information between predictions and model posterior.

Approximate the acquisition function using approximate distribution $q_\theta^*(\boldsymbol{\omega})$

$$\begin{aligned}\mathbb{I}[y, \boldsymbol{\omega}|\mathbf{x}, \mathcal{D}_{\text{train}}] &:= \mathbb{H}[y|\mathbf{x}, \mathcal{D}_{\text{train}}] - \mathbb{E}_{p(\boldsymbol{\omega}|\mathcal{D}_{\text{train}})}\left[\mathbb{H}[y|\mathbf{x}, \boldsymbol{\omega}]\right] \\ &= -\sum p(y=c|\mathbf{x}, \mathcal{D}_{\text{train}}) \log p(y=c|\mathbf{x}, \mathcal{D}_{\text{train}}) \; + \mathbb{E}_{p(\boldsymbol{\omega}|\mathcal{D}_{\text{train}})}\left[\sum_c p(y=c|\mathbf{x}, \boldsymbol{\omega}) \log p(y=c|\mathbf{x}, \boldsymbol{\omega})\right]\end{aligned}$$

$$\mathbb{I}[y, \boldsymbol{\omega}|\mathbf{x}, \mathcal{D}_{\text{train}}] := \mathbb{H}[y|\mathbf{x}, \mathcal{D}_{\text{train}}] - \mathbb{E}_{p(\boldsymbol{\omega}|\mathcal{D}_{\text{train}})}\left[\mathbb{H}[y|\mathbf{x}, \boldsymbol{\omega}]\right]$$
$$= -\sum p(y = c|\mathbf{x}, \mathcal{D}_{\text{train}}) \log p(y = c|\mathbf{x}, \mathcal{D}_{\text{train}}) \; + \mathbb{E}_{p(\boldsymbol{\omega}|\mathcal{D}_{\text{train}})}\left[\sum_c p(y = c|\mathbf{x}, \boldsymbol{\omega}) \log p(y = c|\mathbf{x}, \boldsymbol{\omega})\right]$$

$$p(y = c|\mathbf{x}, \mathcal{D}_{\text{train}}) = \int p(y = c|\mathbf{x}, \boldsymbol{\omega}) p(\boldsymbol{\omega}|\mathcal{D}_{\text{train}}) \mathrm{d}\boldsymbol{\omega}$$

$$\mathbb{I}[y, \boldsymbol{\omega}|\mathbf{x}, \mathcal{D}_{\text{train}}] = -\sum_c \int p(y = c|\mathbf{x}, \boldsymbol{\omega}) p(\boldsymbol{\omega}|\mathcal{D}_{\text{train}}) \mathrm{d}\boldsymbol{\omega} \cdot \log \int p(y = c|\mathbf{x}, \boldsymbol{\omega}) p(\boldsymbol{\omega}|\mathcal{D}_{\text{train}}) \mathrm{d}\boldsymbol{\omega}$$
$$+ \mathbb{E}_{p(\boldsymbol{\omega}|\mathcal{D}_{\text{train}})}\left[\sum_c p(y = c|\mathbf{x}, \boldsymbol{\omega}) \log p(y = c|\mathbf{x}, \boldsymbol{\omega})\right]$$

$$\approx -\sum_c \int p(y = c|\mathbf{x}, \boldsymbol{\omega}) q_\theta^*(\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega} \cdot \log \int p(y = c|\mathbf{x}, \boldsymbol{\omega}) q_\theta^*(\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega} \; + \mathbb{E}_{q_\theta^*(\boldsymbol{\omega})}\left[\sum_c p(y = c|\mathbf{x}, \boldsymbol{\omega}) \log p(y = c|\mathbf{x}, \boldsymbol{\omega})\right]$$

$$\approx -\sum_c \left(\frac{1}{T} \sum_t \widehat{p}_c^t\right) \log \left(\frac{1}{T} \sum_t \widehat{p}_c^t\right) + \frac{1}{T} \sum_{c,t} \widehat{p}_c^t \log \widehat{p}_c^t =: \widehat{\mathbb{I}}[y, \boldsymbol{\omega}|\mathbf{x}, \mathcal{D}_{\text{train}}]$$

$$\approx -\sum_c \left(\frac{1}{T}\sum_t \widehat{p}_c^t\right) \log\left(\frac{1}{T}\sum_t \widehat{p}_c^t\right) + \frac{1}{T}\sum_{c,t} \widehat{p}_c^t \log \widehat{p}_c^t =: \widehat{\mathbb{I}}[y, \boldsymbol{\omega} | \mathbf{x}, \mathcal{D}_{\text{train}}]$$

$$\widehat{\mathbf{p}}^t = [\widehat{p}_1^t, ..., \widehat{p}_C^t] = \text{softmax}(\mathbf{f}^{\widehat{\boldsymbol{\omega}}_t}(\mathbf{x})) \qquad\qquad \widehat{\boldsymbol{\omega}}_t \sim q_\theta^*(\boldsymbol{\omega})$$

$$\widehat{\mathbb{I}}[y, \boldsymbol{\omega} | \mathbf{x}, \mathcal{D}_{\text{train}}] \xrightarrow[T\to\infty]{} \mathbb{H}[y|\mathbf{x}, q_\theta^*] - \mathbb{E}_{q_\theta^*(\boldsymbol{\omega})}\left[\mathbb{H}[y|\mathbf{x}, \boldsymbol{\omega}]\right] \approx \mathbb{I}[y, \boldsymbol{\omega} | \mathbf{x}, \mathcal{D}_{\text{train}}]$$

Weight Uncertainty in Neural Networks ICML2015

$$\mathbf{w}^{\text{MLE}} = \arg\max_{\mathbf{w}} \log P(\mathcal{D}|\mathbf{w}) = \arg\max_{\mathbf{w}} \sum_i \log P(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w})$$

$$\mathbf{w}^{\text{MAP}} = \arg\max_{\mathbf{w}} \log P(\mathbf{w}|\mathcal{D}) = \arg\max_{\mathbf{w}} \log P(\mathcal{D}|\mathbf{w}) + \log P(\mathbf{w})$$

Bayesian inference for neural networks calculates the posterior distribution of the weights given the training data

$$P(\hat{\mathbf{y}}|\hat{\mathbf{x}}) = \mathbb{E}_{P(\mathbf{w}|\mathcal{D})}[P(\hat{\mathbf{y}}|\hat{\mathbf{x}}, \mathbf{w})]$$

Thus taking an expectation under the posterior distribution on weights is equivalent to using an ensemble of an uncountably infinite number of neural networks.

*** suggested finding a variational approximation to the Bayesian posterior distribution on the weights. Variational learning finds the parameters of a distribution on the weights that minimizes the KL divergence with the true Bayesian posterior on the weights:

$$\begin{aligned}\theta^\star &= \arg\min_\theta \mathrm{KL}[q(\mathbf{w}|\theta)||P(\mathbf{w}|\mathcal{D})] \\ &= \arg\min_\theta \int q(\mathbf{w}|\theta) \log \frac{q(\mathbf{w}|\theta)}{P(\mathbf{w})P(\mathcal{D}|\mathbf{w})} \mathbf{dw} \\ &= \arg\min_\theta \mathrm{KL}\left[q(\mathbf{w}|\theta) \;||\; P(\mathbf{w})\right] - \mathbb{E}_{q(\mathbf{w}|\theta)}\left[\log P(\mathcal{D}|\mathbf{w})\right]\end{aligned}$$

$$\mathcal{F}(\mathcal{D},\theta) = \mathrm{KL}\left[q(\mathbf{w}|\theta) \;||\; P(\mathbf{w})\right] - \mathbb{E}_{q(\mathbf{w}|\theta)}\left[\log P(\mathcal{D}|\mathbf{w})\right]$$

The cost function is a sum of a **data-dependent part**, which we shall refer to as the **likelihood cost**, and a **prior-dependent part**, which we shall refer to as the **complexity cost**. The cost function embodies a **trade-off** between satisfying the complexity of the data D and satisfying the simplicity prior P(w).

$$\mathcal{F}(\mathcal{D}, \theta) = \mathrm{KL}\left[q(\mathbf{w}|\theta) \,||\, P(\mathbf{w})\right] - \mathbb{E}_{q(\mathbf{w}|\theta)}\left[\log P(\mathcal{D}|\mathbf{w})\right]$$

Proposition 1. *Let ϵ be a random variable having a probability density given by $q(\epsilon)$ and let $\mathbf{w} = t(\theta, \epsilon)$ where $t(\theta, \epsilon)$ is a deterministic function. Suppose further that the marginal probability density of $\mathbf{w}$, $q(\mathbf{w}|\theta)$, is such that $q(\epsilon)d\epsilon = q(\mathbf{w}|\theta)d\mathbf{w}$. Then for a function f with derivatives in $\mathbf{w}$:*

$$\begin{aligned}
\frac{\partial}{\partial\theta}\mathbb{E}_{q(\mathbf{w}|\theta)}[f(\mathbf{w}, \theta)] &= \frac{\partial}{\partial\theta}\int f(\mathbf{w}, \theta)q(\mathbf{w}|\theta)\mathrm{d}\mathbf{w} \\
&= \frac{\partial}{\partial\theta}\int f(\mathbf{w}, \theta)q(\epsilon)\mathrm{d}\epsilon \\
&= \mathbb{E}_{q(\epsilon)}\left[\frac{\partial f(\mathbf{w}, \theta)}{\partial\mathbf{w}}\frac{\partial\mathbf{w}}{\partial\theta} + \frac{\partial f(\mathbf{w}, \theta)}{\partial\theta}\right]
\end{aligned}$$

Apply Proposition 1 to the optimisation problem

$$\text{let } f(\mathbf{w}, \theta) = \log q(\mathbf{w}|\theta) - \log P(\mathbf{w})P(\mathcal{D}|\mathbf{w}) \qquad \arg\min_{\theta} \int q(\mathbf{w}|\theta) \log \frac{q(\mathbf{w}|\theta)}{P(\mathbf{w})P(\mathcal{D}|\mathbf{w})} \mathrm{d}\mathbf{w}$$

Using Monte Carlo sampling to **evaluate the expectations**, a **backpropagation**-like algorithm is obtained for variational Bayesian inference in neural networks-Bayes by Backprop – which uses unbiased estimates of gradients of the cost to learn a distribution over the weights of a neural network

Bayes by Backprop operates on **weights** (of which there are a **great many**), whilst most previous work applies this method to learning distributions on stochastic hidden units (of which there are far fewer than the number of weights).

Unlike previous work, we do not use the closed form of the complexity cost (or entropic part)

$$\mathcal{F}(\mathcal{D}, \theta) \approx \sum_{i=1}^{n} \log q(\mathbf{w}^{(i)}|\theta) - \log P(\mathbf{w}^{(i)}) - \log P(\mathcal{D}|\mathbf{w}^{(i)})$$

$$\mathcal{F}(\mathcal{D}, \theta) \approx \sum_{i=1}^{n} \log q(\mathbf{w}^{(i)}|\theta) - \log P(\mathbf{w}^{(i)}) - \log P(\mathcal{D}|\mathbf{w}^{(i)})$$

$\mathbf{w}^{(i)}$ denotes the ith Monte Carlo sample drawn from the variational posterior $q(\mathbf{w}^{(i)}|\theta)$

Note that every term of this approximate cost depends upon **the particular weights drawn from the variational posterior**: this is an instance of a variance reduction technique known as common random numbers.

Suppose that the variational posterior is a diagonal Gaussian distribution, then a sample of the weights **w** can be obtained by sampling a unit Gaussian, shifting it by a mean μ and scaling by a standard deviation σ.

$$\sigma = \log(1 + \exp(\rho)) \quad \theta = (\mu, \rho) \quad \mathbf{w} = t(\theta, \epsilon) = \mu + \log(1 + \exp(\rho)) \circ \epsilon$$

1. Sample $\epsilon \sim \mathcal{N}(0, I)$.

2. Let $\mathbf{w} = \mu + \log(1 + \exp(\rho)) \circ \epsilon$

3. Let $\theta = (\mu, \rho)$

4. Let $f(\mathbf{w}, \theta) = \log q(\mathbf{w}|\theta) - \log P(\mathbf{w})P(\mathcal{D}|\mathbf{w})$

5. Calculate the gradient with respect to the mean

$$\Delta_\mu = \frac{\partial f(\mathbf{w}, \theta)}{\partial \mathbf{w}} + \frac{\partial f(\mathbf{w}, \theta)}{\partial \mu}.$$

6. Calculate the gradient with respect to the standard deviation parameter ρ

$$\Delta_\rho = \frac{\partial f(\mathbf{w}, \theta)}{\partial \mathbf{w}} \frac{\epsilon}{1 + \exp(-\rho)} + \frac{\partial f(\mathbf{w}, \theta)}{\partial \rho}$$

7. Update the variational parameters:

$$\mu \leftarrow \mu - \alpha \Delta_\mu$$
$$\rho \leftarrow \rho - \alpha \Delta_\rho.$$

Note that the $\frac{\partial f(\mathbf{w},\theta)}{\partial \mathbf{w}}$ term of the gradients for the mean and standard deviation are shared and **are exactly the gradients found by the usual backpropagation algorithm on a neural network**.

Thus, to learn both the mean and the standard deviation we must simply **calculate the usual gradients found by backpropagation, and then scale and shift them as above**.

Monte Carlo dropout is equivalent to drawing samples of W from the approximate posterior

The uncertainty in the weights induces prediction uncertainty by marginalising over the approximate posterior using Monte Carlo integration:

$$p(y = c|\mathbf{x}, \mathcal{D}_{\text{train}}) = \int p(y = c|\mathbf{x}, \boldsymbol{\omega}) p(\boldsymbol{\omega}|\mathcal{D}_{\text{train}}) \mathrm{d}\boldsymbol{\omega} \approx \int p(y = c|\mathbf{x}, \boldsymbol{\omega}) q_\theta^*(\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega}$$

$$\approx \frac{1}{T} \sum_{t=1}^{T} p(y = c|\mathbf{x}, \widehat{\boldsymbol{\omega}}_t)$$

$$\widehat{\boldsymbol{\omega}}_t \sim q_\theta^*(\boldsymbol{\omega}) \qquad q_\theta(\boldsymbol{\omega}) \text{ is the Dropout distribution}$$

$$p(y|x, \mathbf{X}, \mathbf{Y}) = \int p(y|x, W) p(W|\mathbf{X}, \mathbf{Y}) \, dW \qquad p(y|x, \mathbf{X}, \mathbf{Y}) \approx \frac{1}{T} \sum_{i=1}^{S} p(y|x, W_i)$$

The goal is to compute the posterior probability of labels, which can be written as:

$$p(\mathbf{Z}|\mathbf{Y}_{\mathcal{L}}, \mathbf{X}, \mathcal{G}_{obs}) = \int p(\mathbf{Z}|W, \mathcal{G}, \mathbf{X}) p(W|\mathbf{Y}_{\mathcal{L}}, \mathbf{X}, \mathcal{G}) p(\mathcal{G}|\lambda) p(\lambda|\mathcal{G}_{obs})\, dW\, d\mathcal{G}\, d\lambda$$

Various parametric random graph generation models can be used to model $p(\lambda|\mathcal{G}_{obs})$

A Monte Carlo approximation

$$p(\mathbf{Z}|\mathbf{Y}_{\mathcal{L}}, \mathbf{X}, \mathcal{G}_{obs}) \approx \frac{1}{V} \sum_{v}^{V} \frac{1}{N_G S} \sum_{i=1}^{N_G} \sum_{s=1}^{S} p(\mathbf{Z}|W_{s,i,v}, \mathcal{G}_{i,v}, \mathbf{X})$$

V samples λ_v are drawn from $p(\lambda|\mathcal{G}_{obs})$

The N_G graphs $\mathcal{G}_{i,v}$ are sampled from $p(\mathcal{G}|\lambda_v)$

$W_{s,i,v}$ are sampled from $p(W|\mathbf{Y}_{\mathcal{L}}, \mathbf{X}, \mathcal{G}_{i,v})$ from the Bayesian GCN corresponding to the graph $\mathcal{G}_{i,v}$

Assortative mixed membership stochastic block model

Since G is often noisy and may not fit the adopted parametric block model well, sampling π_v and β_v can lead to high variance. Instead, replace the integration over π and β with a maximum a posteriori estimate.

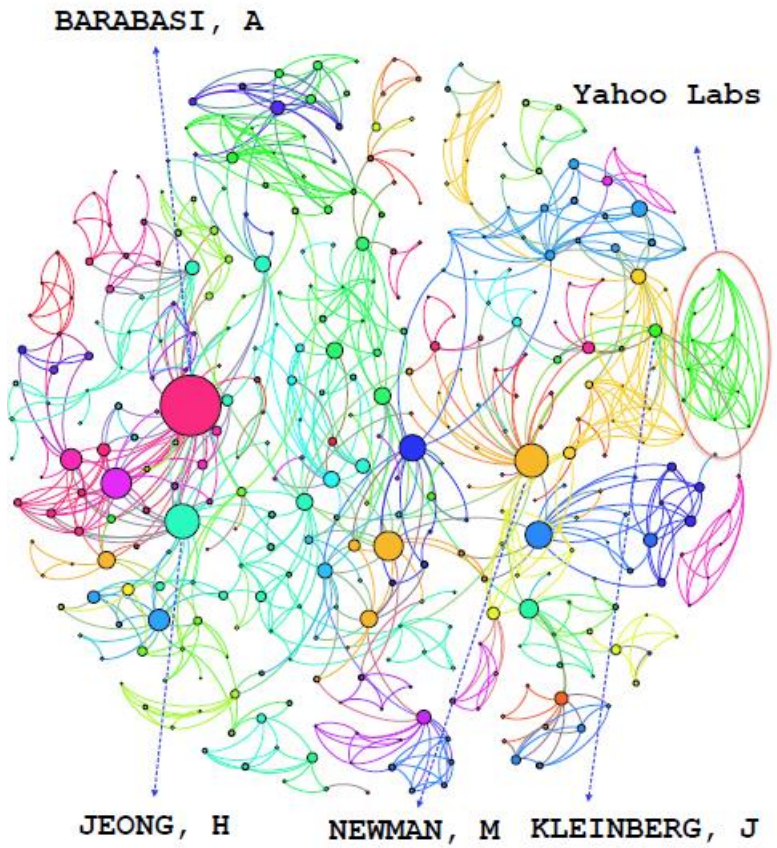
$$\{\hat{\pi}, \hat{\beta}\} = \arg\max_{\beta,\pi} p(\beta, \pi | \mathcal{G}_{obs})$$

$$p(\mathbf{Z}|\mathbf{Y}_{\mathcal{L}}, \mathbf{X}, \mathcal{G}_{obs}) \approx \frac{1}{V} \sum_{v}^{V} \frac{1}{N_G S} \sum_{i=1}^{N_G} \sum_{s=1}^{S} p(\mathbf{Z}|W_{s,i,v}, \mathcal{G}_{i,v}, \mathbf{X})$$

$$p(\mathbf{Z}|\mathbf{Y}_{\mathcal{L}}, \mathbf{X}, \mathcal{G}_{obs}) \approx \frac{1}{N_G S} \sum_{i=1}^{N_G} \sum_{s=1}^{S} p(\mathbf{Z}|W_{s,i}, \mathcal{G}_{i}, \mathbf{X})$$

Posterior inference for the MMSBM

$$\mathcal{G}_{obs} = \{y_{ab} \in \{0,1\} : 1 \leq a < b \leq N\} \qquad \pi_a = [\pi_{a1}, ...\pi_{aK}]^T$$



For any two nodes a and b

- Sample $z_{ab} \sim \pi_a$ and $z_{ba} \sim \pi_b$.
- If $z_{ab} = z_{ba} = k$, sample a link $y_{ab} \sim$ Bernoulli(β_k). Otherwise, $y_{ab} \sim$ Bernoulli(δ).

Here, $0 \leq \beta_k \leq 1$ is termed community strength of the k-th community and δ is the cross community link probability, usually set to a small value.

$$p(\pi, \beta | \mathcal{G}_{obs}) \propto p(\beta) p(\pi) p(\mathcal{G}_{obs} | \pi, \beta)$$

$$= \prod_{k=1}^{K} p(\beta_k) \prod_{a=1}^{N} p(\pi_a) \prod_{1 \leq a < b \leq N} \sum_{z_{ab}, z_{ba}} p(y_{ab}, z_{ab}, z_{ba} | \pi_a, \pi_b, \beta)$$

We use a Beta(η) distribution for the prior of β_k and a Dirichlet distribution, Dir(α), for the prior of π_a, where η and α are hyper-parameters.

Algorithm 1 Bayesian-GCNN

Input: $\mathcal{G}_{obs}$, $\mathbf{X}$, $\mathbf{Y}_{\mathcal{L}}$
Output: $p(\mathbf{Z}|\mathbf{Y}_{\mathcal{L}}, \mathbf{X}, \mathcal{G}_{obs})$

Initialization: train a GCNN to initialize the inference in MMSBM and the weights in the Bayesian GCNN.
for $i = 1 : N_G$ **do**
    Perform N_b iterations of MMSBM inference to obtain $(\hat{\pi}, \hat{\beta})$.
    Sample graph $\mathcal{G}_i \sim p(\mathcal{G}|\hat{\pi}, \hat{\beta})$.
    for $s = 1 : S$ **do**
        Sample weights $W_{s,i}$ via MC dropout by training a GCNN over the graph $\mathcal{G}_i$.
    end for
end for
Approximate $p(\mathbf{Z}|\mathbf{Y}_{\mathcal{L}}, \mathbf{X}, \mathcal{G}_{obs})$ using eq. (8).

Bayesian Semi-supervised Learning with Graph Gaussian Processes NIPS2018

Convolutional Gaussian Processes NIPS2017

Additive Gaussian Processes NIPS2011

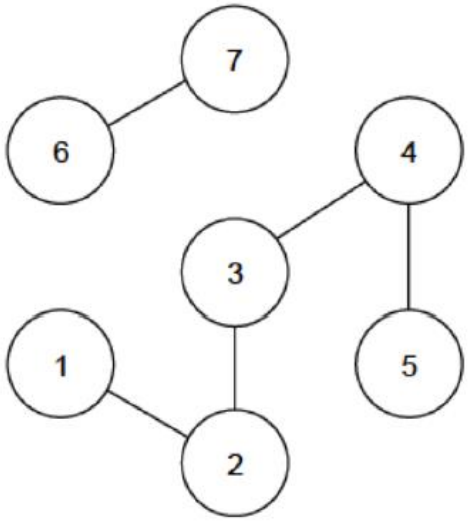
Inter-domain Gaussian Processes for Sparse Inference using Inducing Features NIPS2009

Variational Learning of Inducing Variables in Sparse Gaussian Processes AISTAS2009

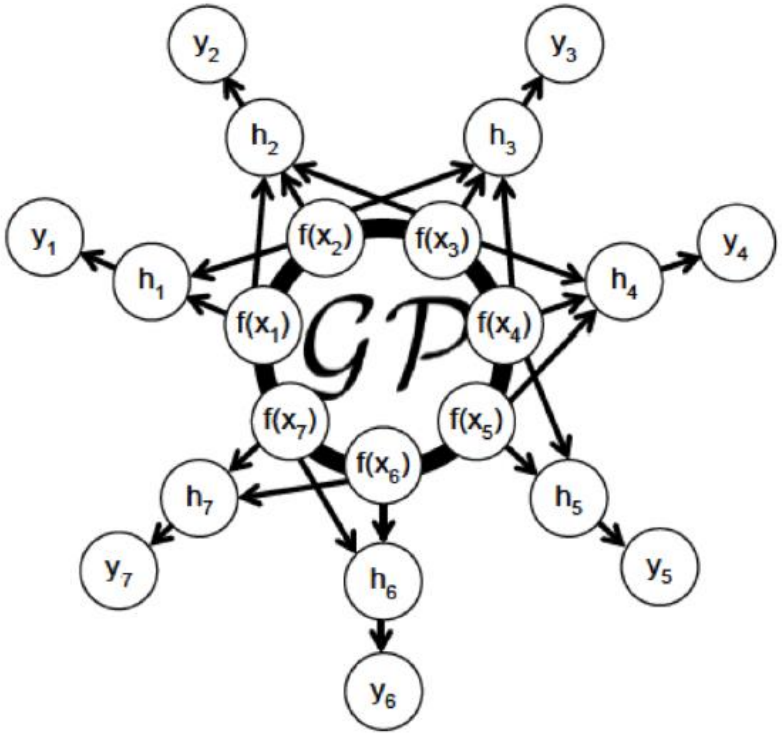
Bayesian Gaussian Process Latent Variable Model AISTAS2010

Graph Convolutional Gaussian Processes arXiv(2019.5.14) ICML2019

Bayesian Semi-supervised Learning with Graph Gaussian Processes NIPS2018



Observed Graph



$$p_\theta(\mathbf{Y}, \mathbf{h}|\mathbf{X}, \mathbf{A}) = p_\theta(\mathbf{h}|\mathbf{X}, \mathbf{A}) \prod_{n=1}^{N} p(y_n|h_n)$$

$$h_n = \frac{f(\mathbf{x}_n) + \sum_{l \in Ne(n)} f(\mathbf{x}_l)}{1 + D_n}$$

$$p_\theta(\mathbf{h}|\mathbf{X}, \mathbf{A}) = \mathcal{N}(\mathbf{0}, \mathbf{P}\mathbf{K}_{\mathbf{XX}}\mathbf{P}^\mathsf{T})$$

$$p(\mathbf{f}) \propto \exp\left(-\sum_{i,j} w_{ij}(\mathbf{f}_i - \mathbf{f}_j)^2 - \delta \sum_i \mathbf{f}_i^2\right) = \exp\left(-\mathbf{f}^\top(\mathbf{L} + \delta\mathbf{I})\mathbf{f}\right) \qquad \mathbf{K}^{-1} = \mathbf{L} + \delta\mathbf{I}$$

$$\mathbb{E}(\|\mathbf{f}_{\mathcal{S}^c} - \hat{\mathbf{f}}_{\mathcal{S}^c}\|^2) = \text{tr}(\mathbb{E}(\mathbf{f}_{\mathcal{S}^c} - \hat{\boldsymbol{\mu}}_{\mathcal{S}^c|\mathcal{S}})(\mathbf{f}_{\mathcal{S}^c} - \hat{\boldsymbol{\mu}}_{\mathcal{S}^c|\mathcal{S}})^\top) = \text{tr}(\hat{\mathbf{K}}_{\mathcal{S}^c|\mathcal{S}})$$

$$f(\mathbf{x}) \sim \mathcal{GP}\big(m(\mathbf{x}), k_\theta(\mathbf{x}, \mathbf{x}')\big) \qquad p_\theta(\mathbf{h}|\mathbf{X}, \mathbf{A}) = \mathcal{N}(\mathbf{0}, \mathbf{P}\mathbf{K}_{\mathbf{X}\mathbf{X}}\mathbf{P}^\top)$$

$$\mathbf{P}\Phi_{\mathbf{X}}\Phi_{\mathbf{X}}^\top\mathbf{P}^\top \qquad \mathbf{P} = (\mathbf{I} + \mathbf{D})^{-1}(\mathbf{I} + \mathbf{A})$$