# Notes on Geometric Learning Papers

Matthew Mo

2020 年 11 月 13 日

## 目录

# 1 NeVAE

**Idea** VAE on graph, node-wise repr, permutation invariant.

## 1.1 Encoder

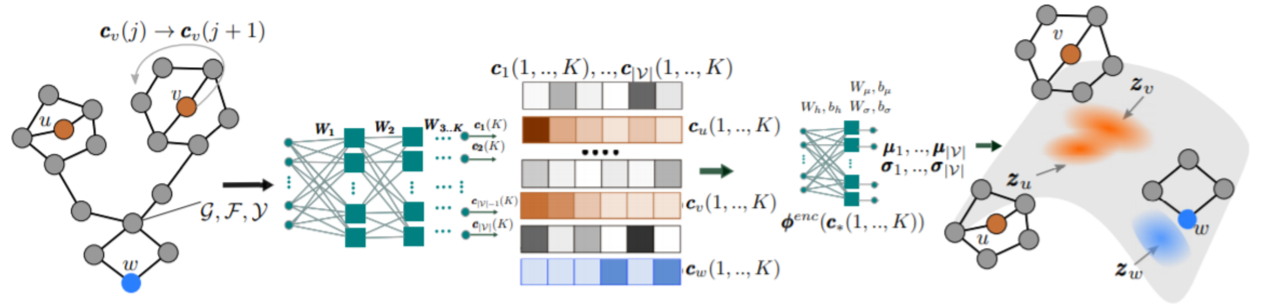



Figure 1: The encoder of our variational autoencoder for molecular graphs. From left to right, given a molecular graph $\mathcal{G}$ with a set of node features $\mathcal{F}$ and edge weights $\mathcal{Y}$, the encoder aggregates information from a different number of hops $j \leq K$ away for each node $v \in \mathcal{G}$ into an embedding vector $\mathbf{c}_v(j)$. These embeddings are fed into a differentiable function $\phi^{enc}$ which parameterizes the posterior distribution $q_\phi$, from where the latent representation of each node in the input graph are sampled from.

GNN-like message-passing on k-hops:

$$q_\phi(\boldsymbol{z}_u|\mathcal{V}, \mathcal{E}, \mathcal{F}, \mathcal{Y}) \sim \mathcal{N}(\boldsymbol{\mu}_u, Diag(\boldsymbol{\sigma}_u)) \tag{1}$$

$$[\boldsymbol{\mu}_u, \boldsymbol{\sigma}_u] = \phi^{enc}(\boldsymbol{c}_u(k)_{k=1..K}) \tag{2}$$

$$\boldsymbol{c}_u(k) = \begin{cases} \boldsymbol{r}(\boldsymbol{W}_k^{\mathcal{T}} \boldsymbol{t}_u + \boldsymbol{W}_k^{\mathcal{X}} \boldsymbol{x}_u), k = 1 \\ \boldsymbol{r}(\boldsymbol{W}_k^{\mathcal{T}} \boldsymbol{t}_u + \boldsymbol{W}_k^{\mathcal{X}} \boldsymbol{x}_u \odot \boldsymbol{\Lambda}(\{y_{uv} \boldsymbol{c}_v(k-1)\}_v \in N(u))), k > 1 \end{cases} \tag{3}$$

## 1.2 Decoder

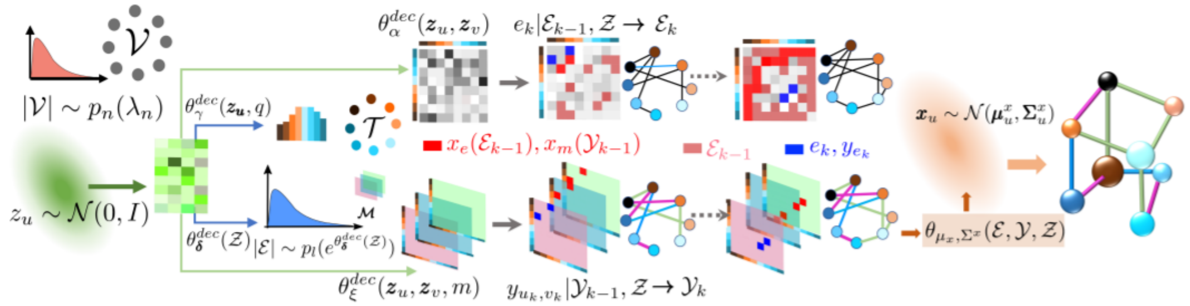



Figure 2: The decoder of our variational autoencoder for molecular graphs. From left to right, the decoder first samples the number of nodes $n = |\mathcal{V}|$ from a Poisson distribution $p_n(\lambda_n)$ and it samples a latent vector $\boldsymbol{z}_u$ per node $u \in \mathcal{V}$ from $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Then, for each node $u$, it represents all potential node feature values as an unnormalized log probability vector (or 'logits'), where each entry is given by a nonlinearity $\theta_\gamma^{dec}$ of the corresponding latent representation $\boldsymbol{z}_u$, feeds this logit into a softmax distribution and samples the node features. Next, it feeds all latent vectors $\mathcal{Z}$ into a nonlinear log intensity function $\theta_{\boldsymbol\delta}^{dec}(\mathcal{Z})$ which is used to sample the number of edges. Thereafter, on the top row, it constructs a logit for all potential edges $(u, v)$, where each entry is given by a nonlinearity $\theta_\alpha^{dec}$ of the corresponding latent representations $(\boldsymbol{z}_u, \boldsymbol{z}_v)$. Then, it samples the edges one by one from a soft max distribution depending on the logit and a mask $\beta_e(\mathcal{E}_{k-1})$, which gets updated every time it samples a new edge $e_k$. On the bottom row, it constructs a logit per edge $(u, v)$ for all potential edge weight values $m$, where each entry is given by a nonlinearity $\theta_\xi^{dec}$ of the latent representations of the edge and edge weight value $(\boldsymbol{z}_u, \boldsymbol{z}_v, m)$. Then, every time it samples an edge, it samples the edge weight value from a soft max distribution depending on the corresponding logit and mask $x_m(u, v)$, which gets updated every time it samples a new $y_{u_k v_k}$. Finally, for each atom $u$, it samples its coordinates $\boldsymbol{x}_u$ from a multidimensional Gaussian distribution whose mean $\mu_{\boldsymbol{x}}$ and variance $\Sigma_{\boldsymbol{x}}$ depends on the latent vectors of the corresponding atom and its neighbors and the underlying chemical bonds.

Decoder: gen. logits, softmax edges one-by-one, possible binary mask(for expert exp.).

$$\text{Nodes Count:}|\mathcal{V}| \sim p_l(\lambda_n) \tag{4}$$

$$\text{Latent Repr.:}\boldsymbol{z}_u \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{5}$$

$$\text{Node Feat.:}\boldsymbol{f}_u = softmax_u(\theta_{\boldsymbol\gamma}^{dec}(\boldsymbol{z}_u, q)), q \text{is atom type} \tag{6}$$

$$\text{Edges Count:}|\mathcal{E}| \sim p_l(e^{\theta_{\boldsymbol\delta}^{dec}(\mathcal{Z})}) \tag{7}$$

$$\text{Edges Gen.:}p(e = (u, v)|\mathcal{E}_{k-1}, \mathcal{V}) = \frac{\beta_e e^{\theta_{\boldsymbol\alpha}^{dec}(\boldsymbol{z}_u, \boldsymbol{z}_v)}}{\sum_{e'=(u',v')\notin\mathcal{E}_{k-1}} \beta_{e'} e^{\theta_{\boldsymbol\alpha}^{dec}(\boldsymbol{z}'_u, \boldsymbol{z}'_v)}} \tag{8}$$

$$\text{E. Feat. Gen.:}p(y_{uv} = m|\mathcal{Y}_{k-1}, \mathcal{V}) = \frac{\beta_m(u, v) e^{\theta_{\boldsymbol\xi}^{dec}(\boldsymbol{z}_u, \boldsymbol{z}_v, m)}}{\sum_{m' \neq m \beta'_m(u,v) e^{\theta_{\boldsymbol\xi}^{dec}(\boldsymbol{z}_u, \boldsymbol{z}_v, m')}}}, \text{note: not normal softmax?} \tag{9}$$

$$\text{Pos. Gen.:}p(\boldsymbol{x}_u|\mathcal{E}, \mathcal{Y}, \mathcal{Z}) = \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x) \tag{10}$$

$$[\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x] = [\theta_{\mu^x}(\boldsymbol{r}(u)), \theta_{\Sigma^x}(\boldsymbol{r}(u))\theta_{\Sigma^x}^T(\boldsymbol{r}(u))] \tag{11}$$

$$\boldsymbol{r}(u) = \boldsymbol{z}_u + \sum_{v \in N(u)} y_{uv} \boldsymbol{z}_v \tag{12}$$

## 1.3 Training

- **Prior**: $\mathcal{Z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
- maximize evidence lower bound(ELBO)+Poisson max-likelihood:

$$\max_{\phi,\theta,\lambda_n} \frac{1}{N} \sum_{i\in[N]} \mathbb{E}_{q_\phi(\mathcal{Z}_i|\mathcal{V}_i,\mathcal{E}_i,\mathcal{F}_i,\mathcal{Y}_i)}[\log p_\theta(\mathcal{Y}_i,\mathcal{E}_i,\mathcal{F}_i,|\mathcal{Z}_i)] - KL(q_\phi||p_z) + \log p_{\lambda_n}(n_i)) \tag{13}$$

- note term $E_{q_\phi}[\log p_\theta(\mathcal{Y}_i,\mathcal{E}_i,\mathcal{F}_i,|\mathcal{Z}_i)]$ need the edges sequence specified, use BFS with random tie breaking in child-sel. step, with random selected source node $s \sim \zeta_s$! Thus

$$\begin{aligned} E_{q_\phi}[\log p_\theta(\mathcal{Y}_i,\mathcal{E}_i,\mathcal{F}_i,|\mathcal{Z}_i)] &\approx E_{q_\phi}[\log \mathbb{E}_{s\sim\zeta_s} p_\theta(\mathcal{Y}_i,\mathcal{E}_i,\mathcal{F}_i,|\mathcal{Z}_i)] & (14) \\ &\geq E_{q_\phi,s\sim\zeta_s}[\log p_\theta(\mathcal{Y}_i,\mathcal{E}_i,\mathcal{F}_i,|\mathcal{Z}_i)] & (15) \end{aligned}$$

- **Theorem** If dist. $\zeta_s$ is independent to labels of nodes, then the learned model is permutation-invariant.
- **Proposition** Decoder defined is permutation-invariant.

## 1.4 Property Oriented Mol. Gen.

Train variational probablistic decoder, to maxmize some property of mol., ⇒train a supervised decoder $p^*$ on trained decoder $p_\theta$:

$$\begin{aligned} &\min_{p(\cdot|\mathcal{Z})} \mathbb{E}_{\mathcal{Z}\sim p_z(\cdot)} \mathbb{E}_{\mathcal{E},\mathcal{Y},\mathcal{F}\sim p(\mathcal{E},\mathcal{Y},\mathcal{F}|\mathcal{Z})}[l(\mathcal{E},\mathcal{Y},\mathcal{F}) + \rho \log \frac{p(\mathcal{E},\mathcal{Y},\mathcal{F}|\mathcal{Z})}{p_\theta(\mathcal{E},\mathcal{Y},\mathcal{F}|\mathcal{Z})}] & (16) \\ &\Rightarrow \min \mathbb{E}_{\mathcal{Z}\in p_z}[KL(p(\cdot|\mathcal{Z})||g_\theta(\cdot|\mathcal{Z}))] & (17) \\ &\text{where } g_\theta(\mathcal{E},\mathcal{Y},\mathcal{F}|\mathcal{Z}) = \frac{p_\theta(\mathcal{E},\mathcal{Y},\mathcal{F}|\mathcal{Z}) \exp\left(-\frac{l(\mathcal{E},\mathcal{Y},\mathcal{F})}{\rho}\right)}{\mathbb{E}_{\mathcal{E},\mathcal{Y},\mathcal{F}\sim p_\theta(\cdot|\mathcal{Z})}[p_\theta(\mathcal{E},\mathcal{Y},\mathcal{F}|\mathcal{Z}) \exp\left(-\frac{l(\mathcal{E},\mathcal{Y},\mathcal{F})}{\rho}\right)]} & (18) \end{aligned}$$

The above equations has a obvious solution $p^* \equiv g_\theta$, however sampling might be too slow for practical use ⇒A Stochastic Gradient Approach.

**Algorithm 1:** PropertyOrientedDecoder: it trains a parameterized property-oriented decoder.

```
Given: The loss function ℓ(·), parameter ρ, original decoder p_θ, # of iterations M, mini batch size B,
and learning rate γ
θ'_0 ← θ
for j = 1, ..., M do
    Z_j ~ p_z(·)
    D ← Minibatch(p_{θ'_j}(·|Z_j), B)
    ∇ ← 0
    for (E_i, Y_i, F_i) ∈ D do
        S ← ℓ(E_i, Y_i, F_i) + ρ log( p_{θ'_j}(E_i, Y_i, F_i|Z_j) / p_θ(E_i, Y_i, F_i|Z_j) )
        ∇ ← ∇ + (S + ρ) ∇_{θ'} log p_{θ'_j}(E_i, Y_i, F_i|Z_j)
    θ'_{j+1} ← θ'_j + γ ∇/B
Return θ'_M
```

Use SGD to update param. $\theta'$ of $p_{\theta'}$:

$$\Delta\theta' = \alpha\nabla_{\theta'}\mathbb{E}_{\mathcal{Z}\sim p_z(\cdot)}\mathbb{E}_{\mathcal{E},\mathcal{Y},\mathcal{F}\sim p(\mathcal{E},\mathcal{Y},\mathcal{F}|\mathcal{Z})}[l(\mathcal{E},\mathcal{Y},\mathcal{F}) + \rho\log\frac{p(\mathcal{E},\mathcal{Y},\mathcal{F}|\mathcal{Z})}{p_\theta(\mathcal{E},\mathcal{Y},\mathcal{F}|\mathcal{Z})}] \tag{19}$$

$$= \alpha\mathbb{E}_{\mathcal{Z}\sim p_z(\cdot)}\nabla_{\theta'}\mathbb{E}_{\mathcal{E},\mathcal{Y},\mathcal{F}\sim p(\mathcal{E},\mathcal{Y},\mathcal{F}|\mathcal{Z})}[l(\mathcal{E},\mathcal{Y},\mathcal{F}) + \rho\log\frac{p(\mathcal{E},\mathcal{Y},\mathcal{F}|\mathcal{Z})}{p_\theta(\mathcal{E},\mathcal{Y},\mathcal{F}|\mathcal{Z})}] \tag{20}$$

$$\text{(by log-deriv. trick)} = \alpha\mathbb{E}_{\mathcal{Z}\sim p_z(\cdot)}\mathbb{E}_{\mathcal{E},\mathcal{Y},\mathcal{F}\sim p(\mathcal{E},\mathcal{Y},\mathcal{F}|\mathcal{Z})}\left[(l(\mathcal{E},\mathcal{Y},\mathcal{F}) + \rho\log\frac{p(\mathcal{E},\mathcal{Y},\mathcal{F}|\mathcal{Z})}{p_\theta(\mathcal{E},\mathcal{Y},\mathcal{F}|\mathcal{Z})} + \rho)\nabla_{\theta'}\log p_{\theta'}\right] \tag{21}$$

by a unbiased MC estim.

$$\approx \frac{1}{M}\sum_{i\in[M]}\left[(l(\mathcal{E},\mathcal{Y},\mathcal{F}) + \rho\log\frac{p(\mathcal{E},\mathcal{Y},\mathcal{F}|\mathcal{Z})}{p_\theta(\mathcal{E},\mathcal{Y},\mathcal{F}|\mathcal{Z})} + \rho)\nabla_{\theta'}\log p_{\theta'}\right] \tag{22}$$

# 2 Seminar on Self/Un-Supervised Learning @ 2020/9/16

## 2.1 Self-Learning @ Video Learning

Supervised success: good & sufficient data, a way different from human! ⇒*Linda Smith, The Dev. of Embodied Cognition*

**Paragidims**:

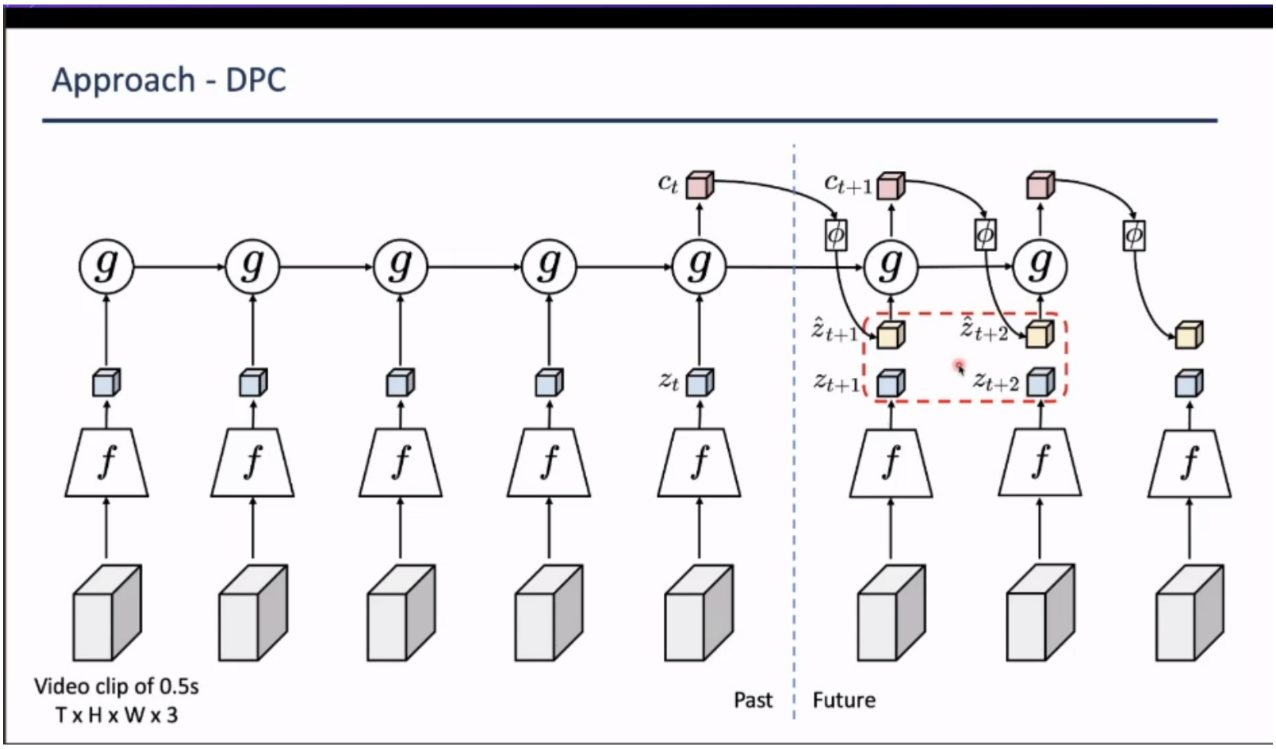
- Use proxy task(e.g. semantics repr.) for a repr., use linear probing for downstream task.
- Use proxy task(e.g. semantics repr.) for a repr., generalizable with *zero annotation*

**Why video-based self-supervised**: like what human percepts, rich info; might with audio.

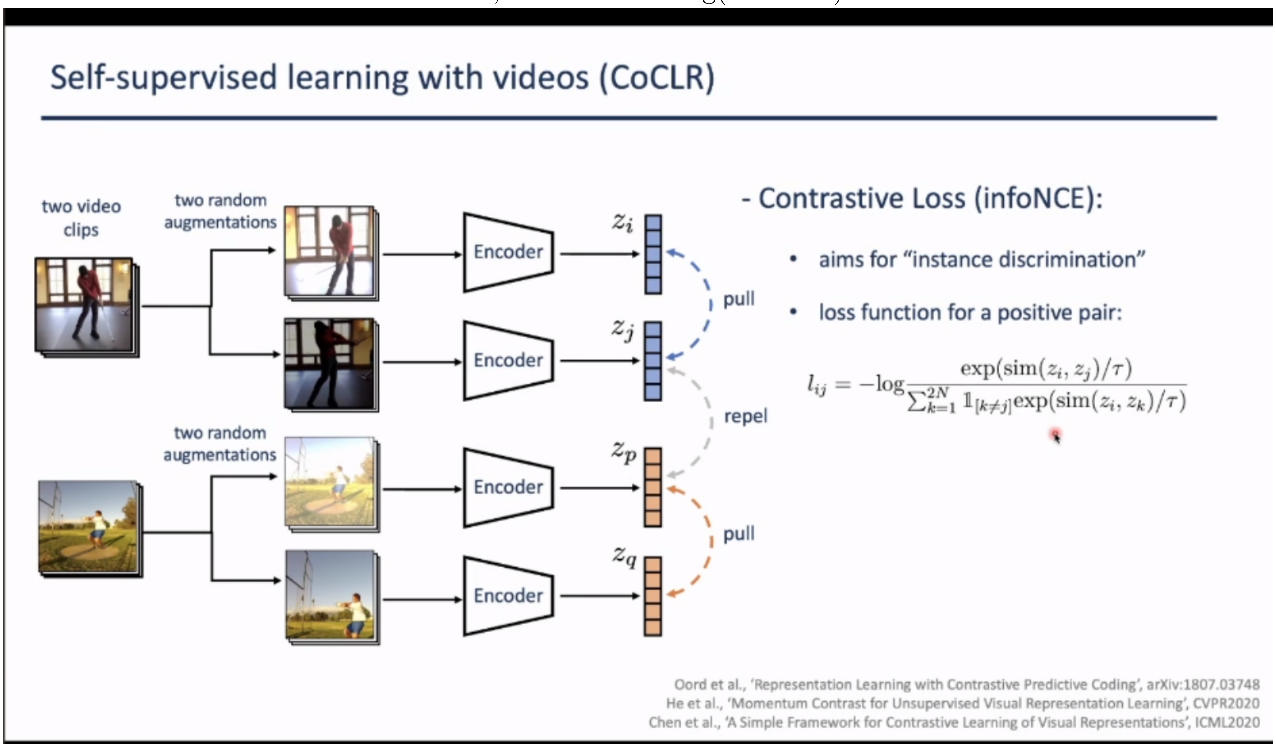
Proxy loss design: temporal info, spatial cohenrence, motions of obj., multimodal

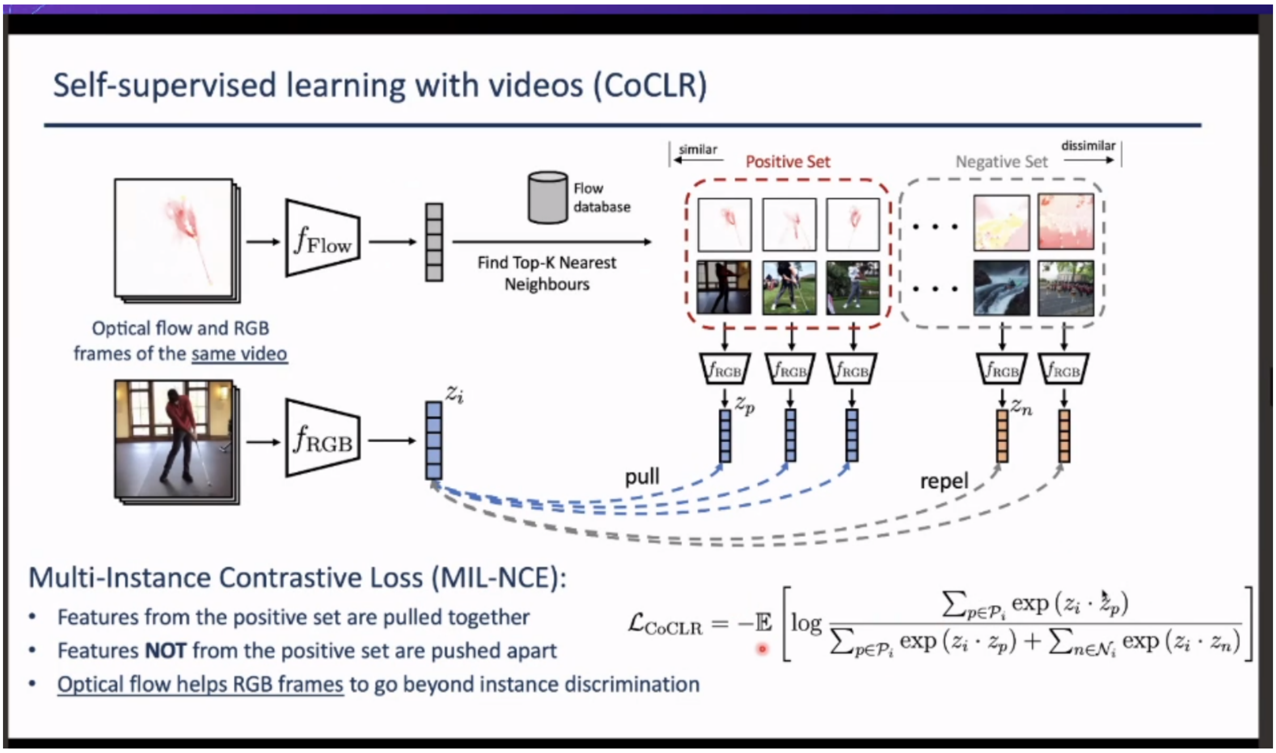
**Temporal**:

- shuffle & learn
- forward or backward?(arrow of time)
- SpeedNet: which speed(frame-rate) is normal/speed-up
- ===Weak, irrelative with downstream tasks===
- DPC: *learn repr. in predicting future in video*




DPC Arch.:encoder-decoder like, contrast learning(infoNCE)






CoCLR: SimCLR like(infoNCE), colearning multimodally with motion flow(MIL-NCE, with noise added)

Audio-Video Co-learning: train a net to check if image/audio clip are same-sourced! Get both video/audio repr.

MAST: self-supervised tracking, give 1st frame mask(seg.), predict sequantial segmentations

**Next**:

- More efficient learning
- Scale up model to uncurrated data, like GPT-1/2/3
- Design proxy task for obj-centric learing
- Design and understand **effective memory!!!**(for video task especially)
- Hand-crafted proxy task ⇒Auto proxy task design?
- Theoratic: small or negative improvement, in upstream task to downstream task.
- Are there difficult task for supervised learning but easy for SSL(e.g. unable to label)?

## 2.2 Transformation Equivariance vs. Invariance @ Visial Repr. Learning

**Contents**:

- TER(Transformation Equivariance Repr.)

- AET(AutoEncoding Transformation)
- AVT(Autoencoding Variational Transformation)
- SAT(Semi-supervised Autoencoding Transformation)

CNN = Translation Equivariant Repr. + FC Classifier. Go beyond: Tranformation Equivariant Repr. + Tranformation Invariant Classifier

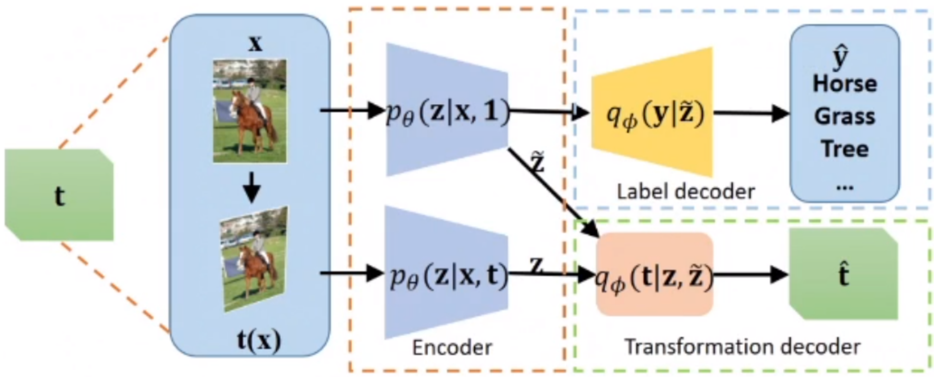
Trans. Equiv.: $E(\boldsymbol{t}(\boldsymbol{x})) = \boldsymbol{\rho}(\boldsymbol{t})[E(\boldsymbol{x})]$. Trans. Inv. is when $\boldsymbol{\rho} \equiv \mathbf{1}_E$.

Steerability: $\boldsymbol{\rho}$ is inpend. with sample $\boldsymbol{x}$.

Targets: Non-linear $\boldsymbol{\rho}$, General Transformation(e.g. recoloring)

## SAT: Semi-Supervised Autoencoding Transformation

- Adding a label decoder $q_\phi(\mathbf{y}|\tilde{\mathbf{z}})$to approximate the posterior $p_\theta(\mathbf{y}|\mathbf{x})$




## Variational Bound

- By introducing label decoder and transformation decoder, we have

$$I_\theta(\mathbf{y}, \mathbf{z}; \tilde{\mathbf{z}}, \mathbf{t}) \geq \mathbb{E}_{p_\theta(\mathbf{y},\mathbf{z},\tilde{\mathbf{z}})} \underbrace{\log q_\phi(\mathbf{y}|\mathbf{z}, \tilde{\mathbf{z}})}_{\text{Label decoder}} + \mathbb{E}_{p_\theta(\mathbf{t},\mathbf{z},\tilde{\mathbf{z}})} \underbrace{\log q_\phi(\mathbf{t}|\mathbf{z}, \tilde{\mathbf{z}})}_{\text{Transformation decoder}}$$

- Jointly maximizing over encoder $\theta$ and decoders $\phi$

$$\max_{\theta,\phi} \mathbb{E}_{p_\theta(\mathbf{y},\mathbf{z},\tilde{\mathbf{z}})} \log q_\phi(\mathbf{y}|\mathbf{z}, \tilde{\mathbf{z}}) + \mathbb{E}_{p_\theta(\mathbf{t},\mathbf{z},\tilde{\mathbf{z}})} \log q_\phi(\mathbf{t}|\mathbf{z}, \tilde{\mathbf{z}})$$

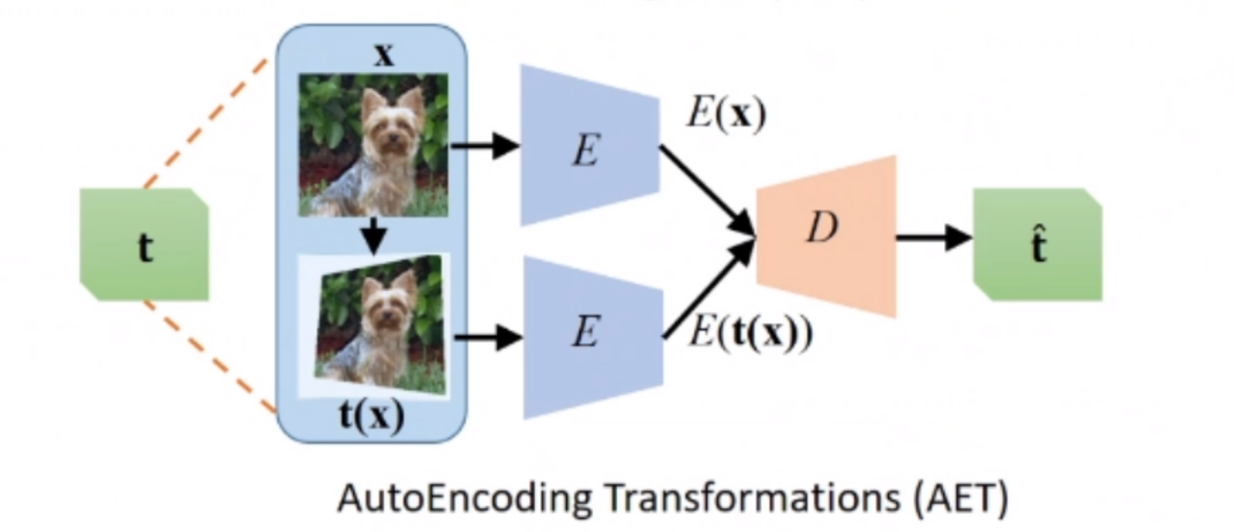
SAT:

- add a label decoder compared to AVT.
- variational surrogate ⇒cross-entropy loss on supervised data + AVT loss

Contrastive Learning: more utilized trans-invariant repr. Future: unifying trans-inv/equiv repr.

# 3 AET, AVT: Autoencoding Transformations

**Idea** autoencoders used in modeling transformations rather than images in order to learn general repr.




AET:

- use autoencoders to learn **transformations**
- trans. generated randomly for self-supervised learing
- use Siamese net as encoder backbone
- AET loss: parameterized, non-parametric, GAN-induced

Losses in AET:

- parameterized transformations: if trans. are parameterized $\mathcal{T} \in \{t_\theta | \theta \in \Theta\}$, loss can be defined as norm of param. diff.
$$l(t_\theta, t_{\hat{\theta}} = ||\theta - \hat{\theta}||.)$$
- for non-parametric trans., use expected distance on source domain
$$l(t, \hat{t}) = \mathbb{E}_{x \sim X}\{dist(t(x), \hat{t}(x)))\}$$

- GAN-induced trans.: image tranformed in form $G(x, z)$, we have loss

$$l(t_z, t_{\hat{z}} = ||z - \hat{z}||.)$$

**Idea of AVT** use prob. dist. to model trans., VAE like modeling!

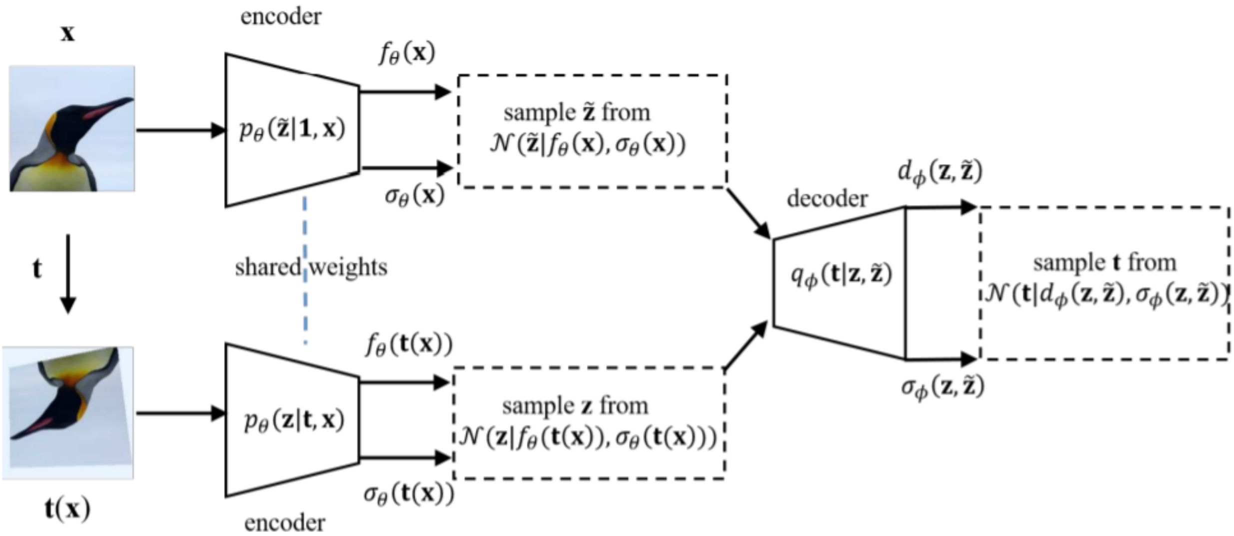



Figure 1: The architecture of the proposed AVT. The original and transformed images are fed through the encoder $p_\theta$ where $\mathbf{1}$ denotes an identity transformation to generate the representation of the original image. The resultant representations $\tilde{\mathbf{z}}$ and $\mathbf{z}$ of original and transformed images are sampled and fed into the transformation decoder $q_\phi$ from which the transformation $\mathbf{t}$ is sampled.

AVT:

- maximize mutual info $I(t; z|\tilde{z})$
- variational bound, introdicing a decoder $q_\phi(t|z, \tilde{z})$:

$$\begin{aligned}
I(t; z|\tilde{z}) &= H(t|\tilde{z}) - H(t|z, \tilde{z}) && (23)\\
&= H(t|\tilde{z}) + \mathbb{E}_{p_\theta(t,z,\tilde{z})}[p_\theta(t|z, \tilde{z})] && (24)\\
&= H(t|\tilde{z}) + \mathbb{E}_{p(t,z,\tilde{z})}[q_\theta(t|z, \tilde{z})] + \mathbb{E}_{p_{(z,\tilde{z})}}[D(p_\theta(t, z, \tilde{z})||q_\phi(t|z, \tilde{z}))] && (25)\\
&\geq H(t|\tilde{z}) + \mathbb{E}_{p(t,z,\tilde{z})}[q_\phi(t|z, \tilde{z})] \equiv \tilde{I}(t; z|\tilde{z}) && (26)\\
&\Rightarrow \max_{\theta,\phi} \mathbb{E}_{p(t,z,\tilde{z})}[q_\phi(t|z, \tilde{z})] && (27)
\end{aligned}$$

- specifically in batch-wise formulation:

$$\mathbb{E}_{p(t,z,\tilde{z})}[q_\phi(t|z,\tilde{z})] \approx \frac{1}{n}\sum_{i=1}^{n} \log \mathcal{N}(t^i|d_\phi(z^i,\tilde{z}^i),\sigma_\phi(z^i,\tilde{z}^i)) \tag{28}$$

$$\text{where } z^i = f_\theta(t^i(x^i)) + \sigma_\theta(t^i(x^i)) \odot \epsilon^i \tag{29}$$

$$\text{and } \tilde{z}^i = f_\theta(x^i) + \sigma_\theta(x^i) \odot \tilde{\epsilon}^i \tag{30}$$

$$\text{where } \epsilon^i, \tilde{\epsilon}^i \sim (\epsilon|0,I), t^i \sim p(t)(\text{predifined or so?}) \tag{31}$$

- trick: take 5 samples to full explore the distribution

# 4 Flow-Based Generative Models

## 4.1 Outline & Basics

Two random vector of same dim.:

$$X \sim P_X(x), z \sim \Pi_Z(z), \text{find mapping } f: Z \to X = x(z), \tag{32}$$

we have

$$\begin{cases} p_X(x) = \pi_Z(f^{-1}(x))\,|\det J(f)|^{-1} \\ \pi_Z(z) = p_X(f(z)))\,|\det J(f)| \end{cases} \tag{33}$$

use a simple dist. on $Z$ and invertibly generate $X \sim p_G(x)$: $x = G(z)$. train $G^{-1}$ as a discriminator. keys: invertible, easy-to-compute $G^{-1}$, easy-to-compute Jacobian determinant.

**"Coupling Layer"**:

$$\begin{cases} (\text{copy}) x_i = z_i, i \le d \\ (\text{affine}) x_i = \beta_i z_i + r_i, d < i \le D \end{cases} \tag{34}$$

$$\beta_{d+1,\ldots D} = F(z_{d+1,\ldots D}), \gamma_{d+1,\ldots D} = H(z_{d+1,\ldots D}) \tag{35}$$

$$J_G = \left[\begin{array}{c|c} \boldsymbol{I}_d & \boldsymbol{O} \\ \hline \boldsymbol{M}(\text{non-matter}) & \boldsymbol{D}(\text{diagonal}) \end{array}\right] \tag{36}$$

$$\det J_G = \prod_{k=d+1..D} \frac{\partial x_k}{\partial z_k} \tag{37}$$

Use many coupling layer to enhance expessive capability. Parts of image does not change $\Rightarrow$exchange copy/affine split:

- exchange within channel
- exchange channel $\Rightarrow$channel rotation use matrix/$1 \times 1$ convolution in MoFlow

## 4.2 MoFlow

**Summary** Flow-based on molecular graphs, channel rotation as $1 \times 1$ convolution, relational GCN layer, graph conditional flow(GCF), use sigmoid rather than exp, split dimensions.

**"Coupling Layer"**:

$$Z_{1:d} = X_{1:d} \tag{38}$$

$$Z_{d+1:n} = X_{d+1:n} \odot sigmoid(S_\Theta(X_{1:d})) + T_\Theta(X_{1:d}) \tag{39}$$

here $S \sim$ scaling, $T \sim$ translation, both by DNNs. 每个耦合层交换上一层 copy 的 dims, 通过一个 channel 上的旋转 $W \in \mathbb{R}^{c\times c}$, 等价于一个 1*1 卷积, 变换后的 Y 分为 $(Y_{1:c/2}, Y_{c/2+1,n})$ 送入下一层. 采用 split-dims 的 trick, 增加交换 channel 的模型自由度增加: $X \in \mathbb{R}^{c\times n\times n} \Rightarrow \mathbb{R}^{ch^2\times n/h\times n/h}$

#### 4.2.1 GCF/Graph Conditional Flow

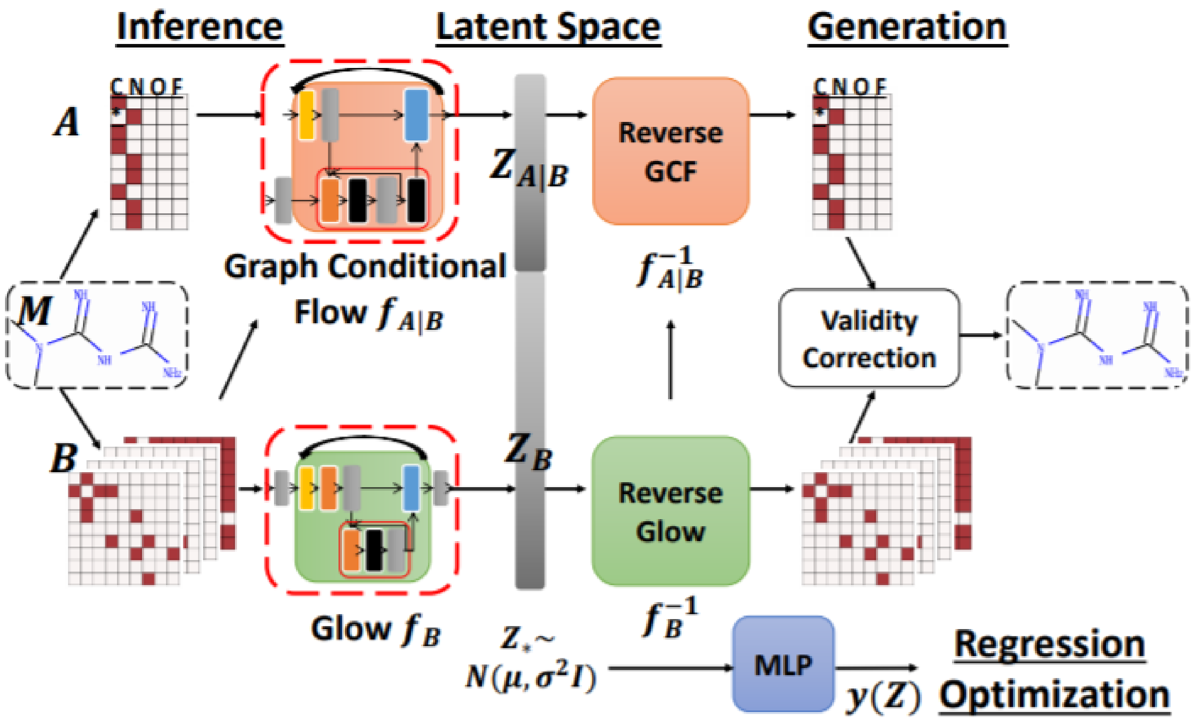



Figure 1: The outline of our MoFlow. A molecular graph $M$ (e.g. Metformin) is represented by a feature matrix $A$ for atoms and adjacency tensors $B$ for bonds. Inference: the graph conditional flow (GCF) $f_{\mathcal{A}|\mathcal{B}}$ for atoms (Sec. 4.2) transforms $A$ given $B$ into conditional latent vector $Z_{A|B}$, and the Glow $f_{\mathcal{B}}$ for bonds (Sec. 4.3) transform $B$ into latent vector $Z_B$. The latent space follows a spherical Gaussian distribution. Generation: the generation process is the reverse transformations of previous operations, followed by a validity correction (Sec. 4.4) procedure which ensures the chemical validity. We summarize MoFlow in Sec. 4.5. Regression and optimization: the mapping $y(Z)$ between latent space and molecular properties are used for molecular graph optimization and property prediction (Sec. 5.3, Sec. 5.4).

**Def.(B conditioned flow)** ….
We have

$$J_{A|B} = \frac{\partial f_{A|B}}{\partial (A, B)} = \left[\begin{array}{c|c} \frac{\partial f_{A|B}}{\partial A} & \frac{\partial f_{A|B}}{\partial B} \\ \hline & \\ \boldsymbol{O} & \boldsymbol{I} \end{array}\right] \tag{40}$$

$$\det J_{A|B} = \det \frac{\partial f_{A|B}}{\partial A} \tag{41}$$

GCF layer:

$$Z_{A|B} = (Z_{A_1|B}, Z_{A_2|B}), A = (A_1|A_2) \tag{42}$$

$$\begin{cases} \textbf{copy } Z_{A_1|B} = A_1 \\ \textbf{affine } Z_{A_2|B} = A_2 \odot sigmoid(S_\Theta(A_1|B)) + T_\Theta(A_1|B) \end{cases} \tag{43}$$

Special designed $S, T$ using R-GCN:

$$graphconv(A_1) = \sum_{i=[C]} \tilde{B}_i (M \odot A) W_i + (M \odot A) W_0, \text{where } M \text{ is the mask of split,} \tag{44}$$

$$\tilde{B} \text{ is normalized } \boldsymbol{B}_i : \tilde{B}_i = \boldsymbol{D}^{-1} \boldsymbol{B}_i, \tag{45}$$

$$\text{where } \boldsymbol{D} \text{ is the full deg-mat. } \boldsymbol{D} = \sum_{c,i} \boldsymbol{B}_{c,i,j} = \sum_c \boldsymbol{D}_i, \text{ computed only once!} \tag{46}$$

#### 4.2.2 Validity Correction & Misc

使用价约束

$$\sum_{c,j} c \times B_{c,i,j} \le Valency_i + Ch_i, \text{where } Ch_i \text{ is formal charge} \tag{47}$$

$$\tag{48}$$

具体的有效性校验方法:

1. 检查价约束, 满足去 2, 否则去 3
2. 返回最大连通子图
3. 第 i 个原子不满足, 对于第 i 个原子, 删去最高阶键, 去 1

这种方法试图再分子上做最小修改来满足价约束.

**Note** 为了防止学到的 prob. dist. 退化, 在数据集上增加 dequantization, 每个 dim 加噪声 $\sim U[0, 0.6]$

### 4.3 GraphNVP

# 5 WGAN

**Idea** Wasserstain 距离代替 KL/JS 距离.
**Method**

- 判别器不用 sigmoid, loss 不取 log
- 判别器参数截断 ⇒ 为了让判别器 Lipschitz 连续.
- trick: 不用基于 momentum 的优化器 (Adam etc.), 用 RMSProp/SGD.

# 6 GMMN+AE

## 6.1 Structure & Idea

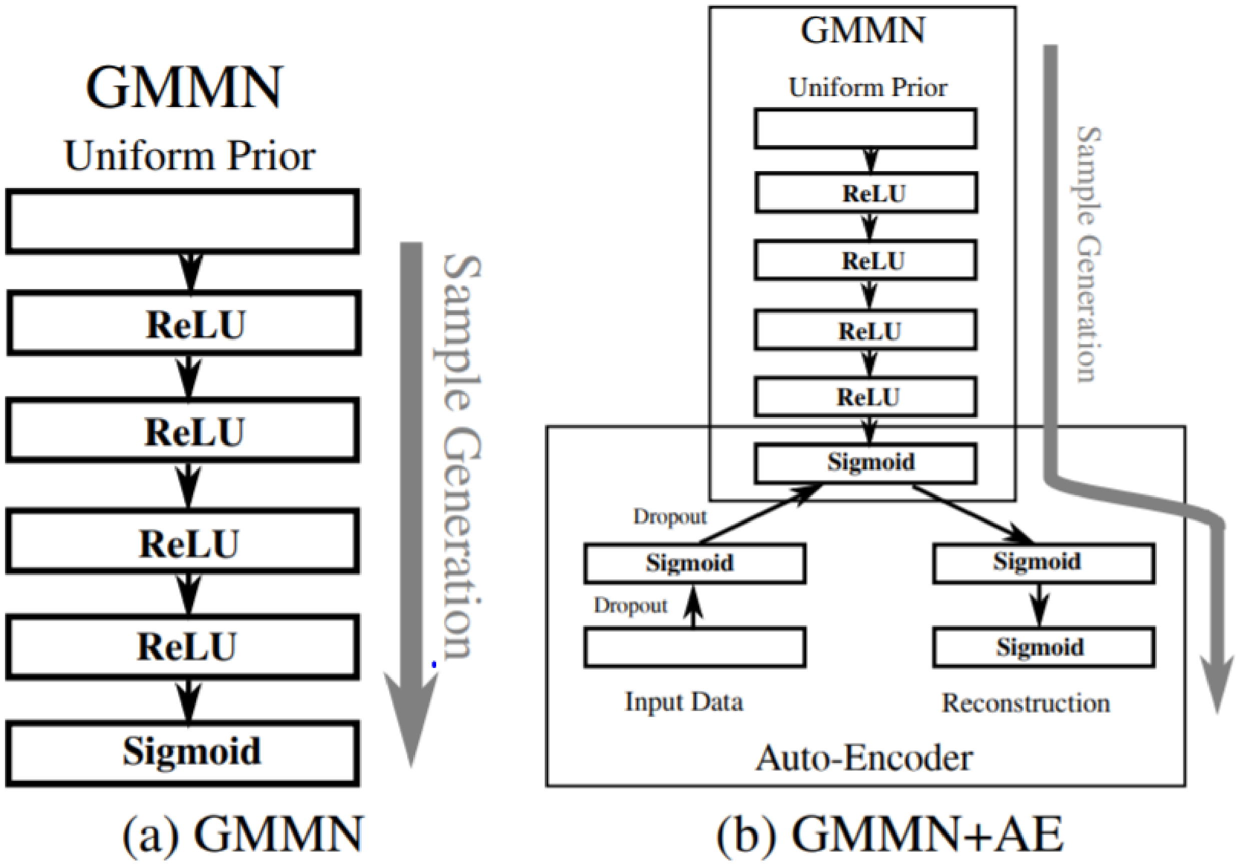



*Figure 1.* Example architectures of our generative moment matching networks. (a) GMMN used in the input data space. (b) GMMN used in the code space of an auto-encoder.

Use MMD(Maximum Mean Discrepency) loss:

$$L_{MMD^2} = ||\frac{1}{N}\sum_i \phi(x_i) - \frac{1}{M}\sum_j \phi(y_j)||^2 \tag{49}$$

$$= \frac{1}{N^2}\sum_{i,i'} K(x_i, x_{i'}) + \frac{1}{M^2}\sum_{i,i'} K(y_i, y_{i'}) - \frac{1}{NM}\sum_{i,j} K(x_i, y_j) \tag{50}$$

使用 k 阶多项式作为核, 则等价于匹配 k 阶矩!⇒ 使用高斯核, 以匹配所有阶矩 (看作幂级数), 这也是 GMMN 的名字由来 (Moment-Matching):

$$K(x,y) = \exp\left(-\frac{1}{2\sigma}||x-y||^2\right) \tag{51}$$

设生成的数据为 $(x_i^s)$,gt. 为 $(x_i^d)$, 则偏导

$$\frac{\partial L_{MMD^2}}{\partial x_{ip}^s} = \frac{1}{\sigma}\left(\frac{2}{M^2}\sum_{j=[M]} K(x_i^s, x_j^s)(x_{jp}^s - x_{ip}^s) - \frac{2}{NM}\sum_{j=[N]} K(x_i^s, x_j^d)(x_{jp}^d - x_{ip}^s)\right) \tag{52}$$

### 6.2 Training

1. 逐层训练 AE
2. Finetune AE
3. 训练 GMMN

## 7 FoldingNet - An AntoEncoder

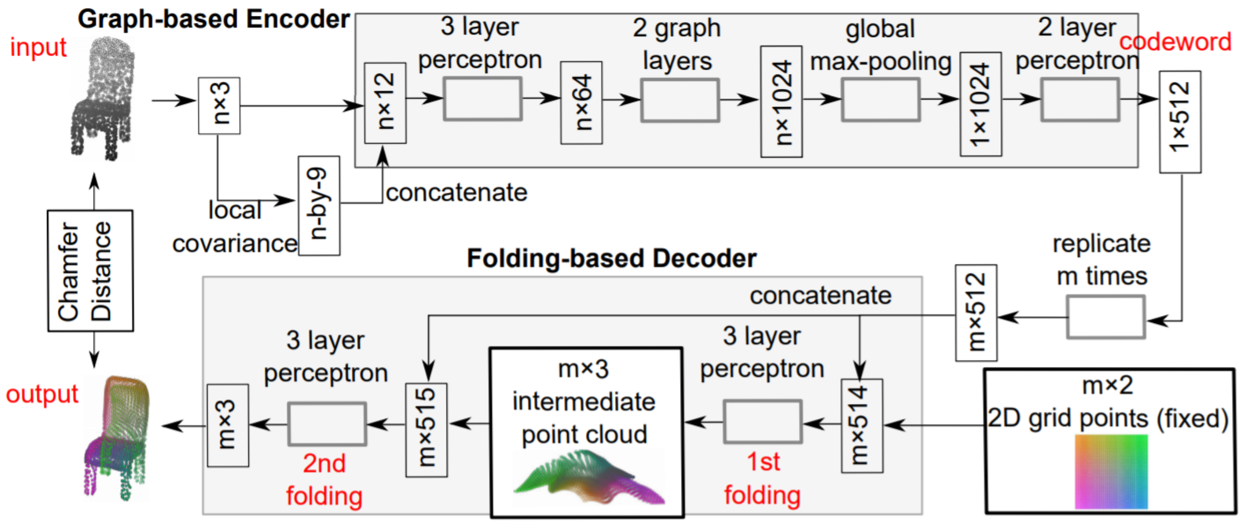



Figure 1. **FoldingNet Architecture**. The graph-layers are the graph-based max-pooling layers mentioned in (2) in Section 2.1. The 1st and the 2nd folding are both implemented by concatenating the codeword to the feature vectors followed by a 3-layer perceptron. Each perceptron independently applies to the feature vector of a single point as in [41], i.e., applies to the rows of the $m$-by-$k$ matrix.

使用 (扩展的)Chamfer 距离

$$d_{CH}(S,\widehat{S}) = \max\{\frac{1}{|S|}\sum_{x\in S}\min_{x\in\widehat{S}}||x-\widehat{x}||, \frac{1}{|\widehat{S}|}\sum_{x\in\widehat{S}}\min_{x\in S}||x-\widehat{x}||\} \tag{53}$$

这个距离让两个点云的点互相配准.

使用基于图的 Encoder: 使用的特征为局部 (KNN 上的) 协方差[1]+ 位置 ($n \times 12$), 简要结构: MLP+GNN-Aggregation+MLP⇒Codeword 其中 Graph Layers

$$\boldsymbol{Y} = \boldsymbol{A}_{\max}(\boldsymbol{X})\boldsymbol{K} \tag{54}$$

$$\boldsymbol{A}_{\max}(\boldsymbol{X})_{ij} = ReLU(\max_{k\in\mathcal{N}(i)} x_{kj}) \tag{55}$$

基于折叠的 Decoder: 重复 m 次 codeword, 和 $2d$ 格点 concat 送到 MLP(1st-folding) 得到中间折叠点云, 和 codeword concat 之后再送到第二个 folding-mlp 中得到结果.

**Prop.** Encoder proposed is permutation-invariant.

**Prop.** Decoder proposed can shape arbitrary point cloud.

# 8 PointFlow: Flow-based Generative Model on Point Clouds

**Idea** As Title

## 8.1 Continuous Normalizing Flow(CNF)

正则化流, 通过一系列可逆变换 $f_i$:

$$x = f_1 \circ \cdots \circ f_n(y) \tag{56}$$

$$\log P(x) = \log P(y) - \sum_i |\log\det \mathcal{J}_{f_i}| \tag{57}$$

离散的正则化流被推广到连续的正则化流—CNFs

$$\frac{\partial y(t)}{t} = f(y(t), t) \tag{58}$$

$$\text{Thus } = y(t_0) + \int_{t_0}^{t_1} f(y(t), t)dt, y(t_0) \sim P(y) \tag{59}$$

$$\log P(x) = \log P(y(t_0)) - \int_{t_0}^{t_1} \mathcal{T}r(\frac{\partial f}{\partial y(t)})dt \tag{60}$$

一个黑盒 ODE 求解器可以用于估计流的输出和输入的梯度!

## 8.2 Variational Auto-Encoder

Optimize ELBO

$$\log P_\theta(X) \geq \log P_\theta(X) - D_{KL}(Q_\phi(z|X)||P_\theta(z|X)) \tag{61}$$

$$= \mathbb{E}_{Q_\phi(z|X)}[\log P_\theta(X|z)] - D_{KL}(Q_\phi(z|X)||P_\psi\theta(z)) \tag{62}$$

[1] 回忆协方差公式

$$\text{Cov}(\boldsymbol{X}) = \mathbb{E}[(\boldsymbol{X} - \mathbb{E}\boldsymbol{X})(\boldsymbol{X} - \mathbb{E}\boldsymbol{X})^T]$$

## 8.3 Model

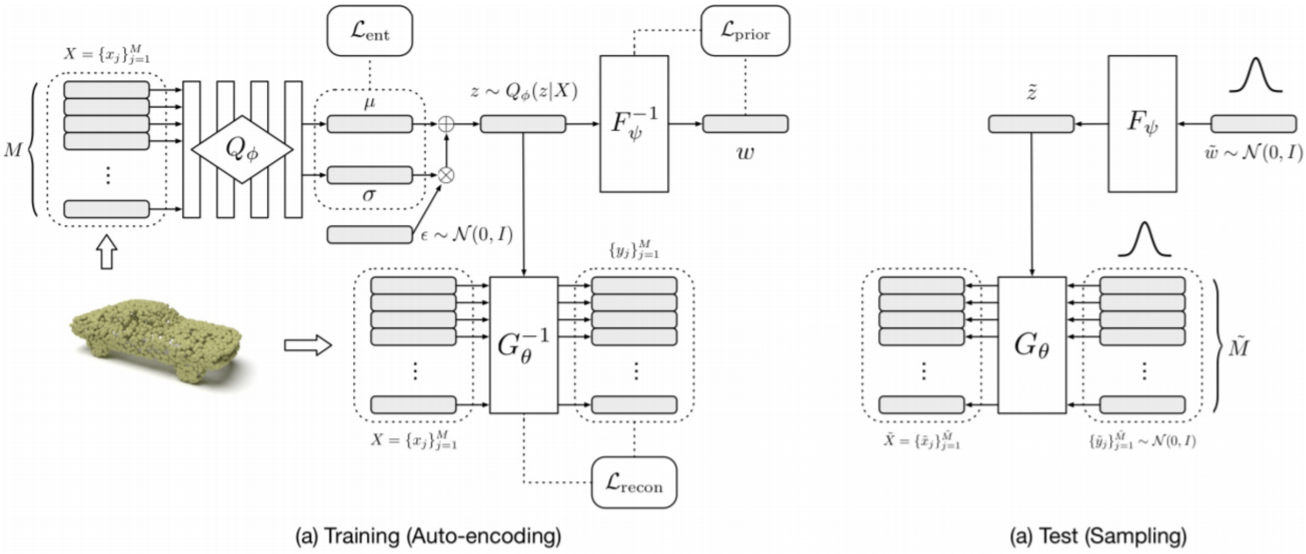



Figure 2: Model architecture. (a) At training time, the encoder $Q_\phi$ infers a posterior over shape representations given an input point cloud $X$, and samples a shape representation $z$ from it. We then compute the probability of $z$ in the prior distribution ($\mathcal{L}_{\text{prior}}$) through a inverse CNF $F_\psi^{-1}$, and compute the reconstruction likelihood of $X$ ($\mathcal{L}_{\text{recon}}$) through another inverse CNF $G_\theta^{-1}$ conditioned on $z$. The model is trained end-to-end to maximize the evidence lower bound (ELBO), which is the sum of $\mathcal{L}_{\text{prior}}$, $\mathcal{L}_{\text{recon}}$, and $\mathcal{L}_{\text{ent}}$ (the entropy of the posterior $Q_\phi(z|X)$). (b) At test time, we sample a shape representation $\tilde{z}$ by sampling $\tilde{w}$ from a Gaussian prior and transforming it with $F_\psi$. To sample points from the shape represented by $\tilde{z}$, we first sample points from the 3-D Gaussian prior and then move them according to the CNF parameterized by $\tilde{z}$.

**Summary** VAE-like. Decoder: Flow-based, i.e. CNF; Prior: CNF-based; Encoder: some simple permutation-invariant encoder.

**Notations**

$$z \sim \text{Latent Repr. for Shape} \tag{63}$$
$$y \sim \text{Simple Distribution/Source Dist. to be Transformed} \tag{64}$$
$$x \sim \text{Point Cloud} \tag{65}$$
$$\tag{66}$$

Point cloud lld

$$\log P_\theta(X|z) = \sum_{x \in X} \log P_\theta(x|z) \tag{67}$$

model $P_(x|z)$ by 条件 CNF

$$x = G_\theta(y(t_0); z) \tag{68}$$
$$= y(t_0) + \int_{t_0}^{t_1} g_\theta(y(t), t; z)dt, y(t_0) \sim P(y) = \mathcal{N}(0, I) \tag{69}$$

reconstruction lld:

$$\log P(x) = \log P(y(t_0)) - \int_{t_0}^{t_1} \mathcal{J}_{g_\theta(t)} dt \tag{70}$$

虽然用高斯分布的先验在 shape repr. 上可行, 但是有证据证明这受限的分布先验在 VAE 中会限制性能. 使用另一个 CNF 来参数化可学习的先验来减少影响

$$D_{KL}(Q_\phi(z|X)||P_\psi\theta(z)) = \mathbb{E}_{Q_\phi(z|X)}[\log P_\psi\theta(z)] - H(P_\psi\theta(z)) \tag{71}$$

obtain $P_\psi$ by $P(w) \sim \mathcal{N}(0, I)$ and CNF

$$z = F_\psi(w(t_0)) \tag{72}$$

$$\triangleq w(t_0) + \int_{t_0}^{t_1} f_\psi(w(t), t)dt, w(t_0) \sim P(w) = \mathcal{N}(0, I) \tag{73}$$

log-probability

$$\log P(x) = \log P(F_\psi^{-1}(z)) - \int_{t_0}^{t_1} \mathcal{J}_{f_\psi(t)} dt \tag{74}$$

最终的 loss term(ELBO)

$$\begin{aligned}
\mathcal{L}(X; \phi, \psi, \theta) &= \mathbb{E}_{Q_\phi(z|x)}\left[\log P_\psi(z) + \log P_\theta(X \mid z)\right] + H\left[Q_\phi(z \mid X)\right] \\
&= \mathbb{E}_{Q_\phi(z|X)}\left[\log P\left(F_\psi^{-1}(z)\right) - \int_{t_0}^{t_1} \operatorname{Tr}\left(\frac{\partial f_\psi}{\partial w(t)}\right) dt\right. \\
&\left. + \sum_{x \in X}\left(\log P\left(G_\theta^{-1}(x; z)\right) - \int_{t_0}^{t_1} \operatorname{Tr}\left(\frac{\partial g_\theta}{\partial y(t)}\right) dt\right)\right] \\
&+ H\left[Q_\phi(z \mid X)\right]
\end{aligned} \tag{75}$$

can be interpretes in 3 parts:

1. Prior: $\mathcal{L}_{\text{prior}}(X; \psi, \phi) \triangleq \mathbb{E}_{Q_\phi(z|x)}[\log P_\psi(z)]$, use reparametrization to MC-sample:

$$\mathbb{E}_{Q_\phi(z|x)}\left[\log P_\psi(z)\right] \approx \frac{1}{L} \sum_{l=1}^{L} \log P_\psi\left(\mu + \epsilon_l \odot \sigma\right) \tag{76}$$

2. Recon. ld.: $\mathcal{L}_{\text{recon}}(X; \theta, \phi) \triangleq \mathbb{E}_{Q_\phi(z|x)}[\log P_\theta(X \mid z)]$, 依然使用 MC 采样估计.

3. Posterior Entopy: $\mathcal{L}_{\text{ent}}(X; \phi) \triangleq H[Q_\phi(z \mid X)]$, has form

$$H\left[Q_\phi(z \mid X)\right] = \frac{d}{2}(1 + \ln(2\pi)) + \sum_{i=1}^{d} \ln \sigma_i \tag{77}$$

# 9 FFJORD

## 9.1 CNF

use some base dist. $\mathbf{z}_0 \sim p_{z_0}(\mathbf{z}_0)$, 通过含时 ODE 得到要建模的分布

$$\boldsymbol{z}(t_0) = \boldsymbol{z}_0 \tag{78}$$

$$\frac{\partial \boldsymbol{z}}{\partial t} = f(\boldsymbol{z}(t), t; \theta) \tag{79}$$

log-pdf 的方程 (*instantaneous change of variables* form.)

$$\frac{\partial \log p(\mathbf{z}(t))}{\partial t} = -\operatorname{Tr}\left(\frac{\partial f}{\partial \mathbf{z}(t)}\right) \tag{80}$$

$$\log p\left(\mathbf{z}\left(t_1\right)\right) = \log p\left(\mathbf{z}\left(t_0\right)\right) - \int^{t_1} \operatorname{Tr}\left(\frac{\partial f}{\partial \mathbf{z}(t)}\right) dt \tag{81}$$

$$\underbrace{\begin{bmatrix} \mathrm{z}_0 \\ \log p(\mathbf{x}) - \log p_{z_0}(\mathbf{z}_0) \end{bmatrix}}_{\text{solutions}} = \underbrace{\int_{t_1}^{t_0} \begin{bmatrix} f(\mathbf{z}(t), t; \theta) \\ -\operatorname{Tr}\left(\frac{\partial f}{\partial \mathbf{z}(t)}\right) \end{bmatrix} dt,}_{\text{dynamics}} \underbrace{\begin{bmatrix} \mathbf{z}\left(t_1\right) \\ \log p(\mathbf{x}) - \log p\left(\mathbf{z}\left(t_1\right)\right) \end{bmatrix} = \begin{bmatrix} \mathbf{x} \\ 0 \end{bmatrix}}_{\text{initial values}} \tag{82}$$

## 9.2 Backpropagation through ODE Solutions with Adjoint Method

Problem: calc. deriv. based on loss func.

$$L\left(\mathbf{z}\left(t_1\right)\right) = L\left(\int_{t_0}^{t_1} f(\mathbf{z}(t), t; \theta) dt\right) \tag{83}$$

Pontryagin(1962) 证明

$$\frac{dL}{d\theta} = -\int_{t_1}^{t_0} \left(\frac{\partial L}{\partial \mathbf{z}(t)}\right)^T \frac{\partial f(\mathbf{z}(t), t; \theta)}{\partial \theta} dt \tag{84}$$

值 $-\partial L / \partial \boldsymbol{z}(t)$ 称为 ODE 的伴随状态 (adjoint state). 使用一个 black-box ODE solver 来计算 $\boldsymbol{z}(t_1)$, 再用初值 $\partial L / \partial \boldsymbol{z}(t_1)$ 送进这个 ODE solver 来计算 (84)

## 9.3 Unbiased Linear-Time Log-Density Estimation

Hutchinson Estimator:

$$\operatorname{Tr}(A) = E_{p(\boldsymbol{\epsilon})}\left[\boldsymbol{\epsilon}^T A \boldsymbol{\epsilon}\right] \tag{85}$$

holds if $\mathbb{E}[\boldsymbol{\epsilon}] = 0, \operatorname{Cov} \boldsymbol{\epsilon} = I$ to avoid randomness, fix noise at each round of solving ODE

$$\begin{aligned} \log p\left(\mathbf{z}\left(t_1\right)\right) &= \log p\left(\mathbf{z}\left(t_0\right)\right) - \int_{t_0}^{t_1} \operatorname{Tr}\left(\frac{\partial f}{\partial \mathbf{z}(t)}\right) dt \\ &= \log p\left(\mathbf{z}\left(t_0\right)\right) - \int_{t_0}^{t_1} \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\boldsymbol{\epsilon}^T \frac{\partial f}{\partial \mathbf{z}(t)} \boldsymbol{\epsilon}\right] dt \\ &= \log p\left(\mathbf{z}\left(t_0\right)\right) - \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\int_{t_0}^{t_1} \boldsymbol{\epsilon}^T \frac{\partial f}{\partial \mathbf{z}(t)} \boldsymbol{\epsilon} dt\right] \end{aligned} \tag{86}$$

噪声分布可以选为高斯分布/Rademacher 分布[2] 并且向量和 Jacobian 的乘积, i.e. $\boldsymbol{\epsilon}\frac{\partial f}{\partial \mathbf{z}(t)}$, 可以快速算出 (通过 auto-diff)

Trick: Bottleneck width $H$ to reduce variance of estimator.

**Algorithm 1** Unbiased stochastic log-density estimation using the FFJORD model

**Require:** dynamics $f_\theta$, start time $t_0$, stop time $t_1$, minibatch of samples $\mathbf{x}$.

```
ε ← sample_unit_variance(x.shape)                      ▷ Sample ε outside of the integral
function f_aug([z_t, log p_t], t):                      ▷ Augment f with log-density dynamics.
    f_t ← f_θ(z(t), t)                                  ▷ Evaluate dynamics
    g ← ε^T ∂f/∂z |_z(t)                                ▷ Compute vector-Jacobian product with automatic differentiation
    T̃r = matrix_multiply(g, ε)                          ▷ Unbiased estimate of Tr(∂f/∂z) with ε^T ∂f/∂z ε
    return [f_t, −T̃r]                                   ▷ Concatenate dynamics of state and log-density
end function
[z, Δ_logp] ← odeint(f_aug, [x, 0⃗], t_0, t_1)          ▷ Solve the ODE, ie. ∫_{t_0}^{t_1} f_aug([z(t), log p(z(t))], t) dt
log p̂(x) ← log p_{z_0}(z) - Δ_logp                      ▷ Add change in log-density
return log p̂(x)
```

# 10 Dequantization to Learn Discrete Distribution

为了近似一个离散空间上的 pd., 需要通过在数据点上加入噪声, 使用 "去量化" 技巧 (dequantization). 可变性更好的 noise ⇒ 更紧的下界 ⇒learned noise?.

**Theorem** 加入合适的噪声后的连续随机变量的 ld(likelihood) 是对应离散随机变量 ld 的下界.

## 10.1 Dequantization as Latent Variable Model

$$P_{model}(x) = \int P_\vartheta(x|v)p(v)dv, \tag{88}$$

$$\text{where } P_\vartheta(x|v) = \mathbb{1}[v \in B_\vartheta(x)] \tag{89}$$

称 $P_\vartheta(x|v)$ 是量化子 (quantizer). 不同的量化子导致了不同的去量化方法. Half-infinite dequant. for bin. var.: $B(x) = \{x \cdot u | u \in \mathbb{R}_+^D\}, x \in -1, 1$; Hypercube dequant. for grid var.(images etc.): $B(x) = \{x + u | u \in [0,1)^D\}$

上述积分难以计算, 引入去量化子 $q_\phi(v|x)$, 注意它具有不重叠的紧支撑集, 为此标记 $u = v + x$

[2]在 $\{-1,1\}$ 上均匀分布的离散分布

$$f(k) = \begin{cases} 1/2 & \text{if } k = -1 \\ 1/2 & \text{if } k = +1 \\ 0 & \text{otherwise} \end{cases} \tag{87}$$

$$P_{model}(x) = \int \frac{q_\phi(u|x) P_\vartheta(x|v) p(v)}{q_\phi(u|x)} dv \tag{90}$$

$$= \mathbb{E}_{u\sim q_\phi(u|x)} \left[ \frac{P_\vartheta(x|v)p(v)}{q_\phi(u|x)} \right] \tag{91}$$

现有的方法常常使用格点积分作为离散和连续模型的区分

$$P(x) = \int_{[0,1)^D} p(x+u) du \tag{92}$$

下面提出三种 dequant. ： *i) variational inference, ii) weighted importance sampling, iii) variational Renyi approx.*

## 10.2 Variational Dequantization

根据 Jensen 不等式, 得到 lld 的变分代理函数

$$\log P_{model}(x) \geq \mathbb{E}_{u\sim q_\phi(u|x)} \left[ \log \frac{P_\vartheta(x|v) p_\theta(v)}{q_\phi(u|x)} \right] \tag{93}$$

注意去量化子要有紧支撑集, 所以对其输出进行 sigmoid; 并且进而有

$$\log P_{model}(x) \geq \mathbb{E}_{u\sim q_\phi(u|x)} [\log p_\theta(v)] + \mathbb{H}[q_\phi] \tag{94}$$

熵一项防止了概率分布退化到离散点上的 delta-peak, 从而推出了变分去量化 (vi dequant.)

## 10.3 Importance-Weighted Dequantization

除此之外, 还可以把去量化分布看作 proposal dist., 用采样多次替代 Jensen 不等式:

$$\log P_{model}(x) \geq \log \left[ \frac{1}{K} \sum_{k\in[K]} \frac{P_\vartheta(x|v_k) p_\theta(v_k)}{q_\phi(u_k|x)} \right] \tag{95}$$

若提案分布限制于紧支撑集上, 则有

$$\log P_{model}(x) \geq \log \left[ \frac{1}{K} \sum_{k\in[K]} \frac{p_\theta(v_k)}{q_\phi(u_k|x)} \right] \tag{96}$$

$$= \log [w_k(x)] \tag{97}$$

$$\text{where } w_k(x) \triangleq \frac{p_\theta(v_k)}{q_\phi(u_k|x)} \tag{98}$$

若 $K \to \infty$, 则取等号, 否则给出了 lld 的一个下界 (*iw-bound*), 故给出了 vi 界的更好估计 ⇒iw-dequant.

### 10.4 Renyi Dequantization

vi/iw-去量化都可以看作变分 Renyi 去量化的特例. lld 可以用 Renyi Divergence 提供下界

$$\log P_{model}(x) \geq \frac{1}{1-\alpha} \log \left[ \frac{1}{K} \sum_{k \in [K]} \left( \frac{P_\vartheta(x|v_k) p_\theta(v_k)}{q_\phi(u_k|x)} \right)^{1-\alpha} \right] \tag{99}$$

$$\text{where } \alpha \in [0, 1) \tag{100}$$

vi-bound $\alpha \to 1$, iw-bound $\alpha = 0$. [Li & Turner, 2016] 考虑小于 0 的 $\alpha$, 这可能在低采样数时提供更紧的下界 (当 $K \to \infty$, 实际上提供了一个上界). 令 $\alpha = -\infty$, 得到 VR-max(variational Renyi max-approximation), 一个 iw-bound 的快速估计

$$\log P_{model}(x) \approx \log \max_k w_k(x) \tag{101}$$

**Detail** 用 Cholesky 分解计算协方差矩阵 $\Lambda = \Gamma\Gamma^T$, $\Gamma^T$ 可学习.

### 10.5 Dequantization Distribution

**Uniform Dequant.** $q_\phi(u|x)$ is uniform in $\mathcal{B}(x)$

**Gaussian Dequant.** 更具表达力的是条件 logit-正态分布 (cond. logit-normal dist.)

$$q_\phi(u|x) = sigmoid(\mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))) \tag{102}$$

**Flow-based Dequant.**

$$q_\phi(u|x) = q_\phi(\varepsilon = f_\phi(sigmoid^{-1}(u); x)|x) \det \boldsymbol{J} \tag{103}$$

由基分布 $q_\phi(\varepsilon|x)$ 和流双射 $f \in \mathbb{R}^D \to \mathbb{R}^D$ 组成. 这里的基分布采用对角高斯分布, 以及两种双射: coupling layer/flow/bipartite 和 autoregressive.

*Bipartite Dequant.(Dinh et al., 2017)* 使用流模型的耦合层:

$$\begin{cases} (\text{copy}) u_1 = \varepsilon + 1 \\ (\text{affine}) u_2 = \varepsilon_2 \odot s_\phi(\varepsilon_1; x) + t_\phi(\varepsilon_1) \end{cases} \tag{104}$$

(105)

为了保证所有分量都被变换, 应用另一个更改了 copy 层位置的耦合层.

*Autoregressive Dequant./ARD(Kingma et al., 2016)* 使用一个自回归模型

$$[m, s] = ARM_\phi(\varepsilon, h) \tag{106}$$

$$u = s \odot \varepsilon + m \tag{107}$$

其中 $h$ 是上下文变量, 基于条件变量 $x$, 通过网络 $s$ 计算出来.

### 10.6 (Choice of) Continuous Distribution

可以按前一节那样任意地选择量化子 $p_\theta(v)$. 但是在训练中需要采样 $v \sim p_\theta(v)$, 故使用自回归模型是很慢的, 故只考虑对角协方差/正常协方差的高斯分布和二分 flow-based 模型.

# 11 DGI: Deep Graph Infomax

## 11.1 Backgrounds, Approach, Math

**Target** Learn a encoder $\mathcal{E} = \mathcal{E}(\boldsymbol{X}, \boldsymbol{A}) : \mathbb{R}^{N\times F} \times \mathbb{R}^{N\times N} \to \mathbb{R}^{N\times F}$.

**Approach** 最大化局部互信息. 使用 *Readout* 函数来获得全局图特征 $\vec{s} = \mathcal{R}(\mathcal{E}(\mathbf{X}, \mathbf{A}))$. 为了能够计算 MI, 引入判别器 $\mathcal{D} : \mathbb{R}^F \times \mathbb{R}^F \to \mathbb{R}$, $\mathcal{D}\left(\vec{h}_i, \vec{s}\right)$ 代表了两个图的 (repr.) 相似度. 判别器的负样本通过把一个图和一个不同的图联系在一起组成. 对于多图场景 (ModelNet/Molecule Graphs) 这可以通过采样其他图得到; 对于单图情景 (Cora etc.), 必须定义一个 (随机) 损坏函数 $\mathcal{C} : \mathbb{R}^{N\times F} \times \mathbb{R}^{N\times N} \to \mathbb{R}^{M\times F} \times \mathbb{R}^{M\times M}$

为此, 使用 contrastive loss

$$\mathcal{L} = \frac{1}{N+M}\left(\sum_{i=1}^{N} \mathbb{E}_{(\mathbf{X},\mathbf{A})}\left[\log \mathcal{D}\left(\vec{h}_i, \vec{s}\right)\right] + \sum_{j=1}^{M} \mathbb{E}_{(\tilde{\mathbf{X}},\tilde{\mathbf{A}})}\left[\log\left(1 - \mathcal{D}\left(\vec{\tilde{h}}_j, \vec{s}\right)\right)\right]\right) \tag{108}$$

这个 Jensen-Shannon divergence 本质上是互信息的 estimator!

**Lemma 1.** $\{\boldsymbol{X}^{(k)}\}_{k\in[|\boldsymbol{X}|]}$ 从 $p(\boldsymbol{X})$ 中取出的一系列节点表示, 且 $p\left(\mathbf{X}^{(k)}\right) = p\left(\mathbf{X}^{(k')}\right) \forall k, k'$, 并且 $\mathcal{R}(\odot)$ 是确定性 Readout 函数, $\vec{s}^{(k)} = \mathcal{R}\left(\mathbf{X}^{(k)}\right)$, 具有边缘分布 $p(\vec{s})$. 则基于联合分布的最优分类器 $p(\mathbf{X}, \vec{s})$ 和边缘分布的乘积 $p(\mathbf{X})p(\vec{s})$ 的误差有上界 $\mathrm{Err}^* = \frac{1}{2}\sum_{k=1}^{|\mathbf{X}|} p\left(\vec{s}^{(k)}\right)^2$. 当 $\mathcal{R}$ 是单射时达到上界.

**Corollary 1.** 此后都假设 $\mathcal{R}$ 是单射, 假设 $\vec{s}$ 的状态不少于 $|\mathbf{X}|$, 则最优全局表示满足 $|\vec{s}^*| = |\mathbf{X}|$.

**Theorem 1.** $\vec{s}^\star = \mathrm{argmax}_{\vec{s}}\, \mathrm{I}(\mathbf{X}; \vec{s})$

**Theorem 2.** 令 $\mathbf{X}_i^{(k)} = \{\vec{x}_j\}_{j\in n\left(\mathbf{X}^{(k)}, i\right)}$, 是第 k 层图卷积的特征, $\vec{h}_i = \mathcal{E}\left(\mathbf{X}_i^{(k)}\right)$, 假设 $|\mathbf{X}_i| = |\mathbf{X}| = |\vec{s}| \geq \left|\vec{h}_i\right|$, 则最小化 $p\left(\vec{h}_i, \vec{s}\right)$ and $p\left(\vec{h}_i\right)p(\vec{s})$ 的 $\vec{h}_i$ 也让 MI 最大化.

## 11.2 Algorithm

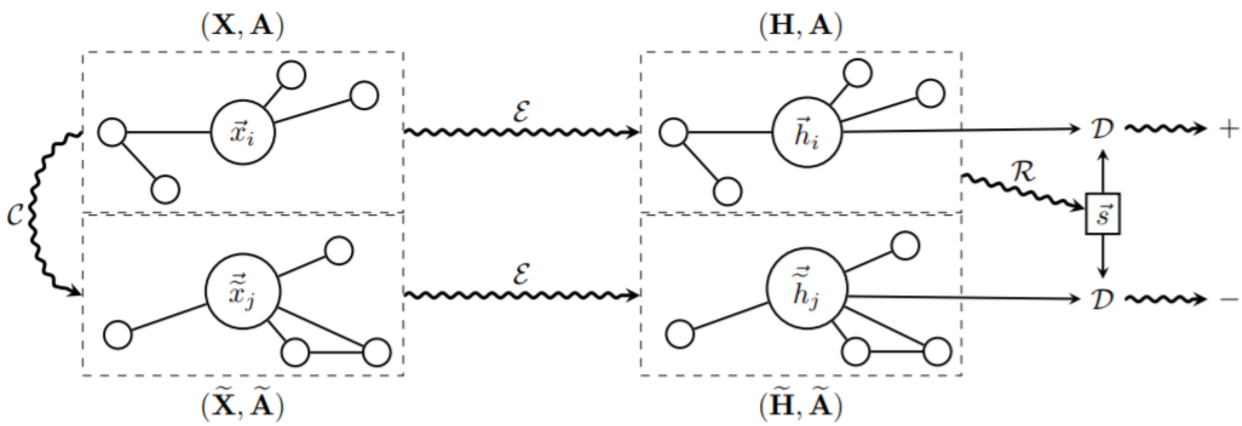



Figure 1: A high-level overview of Deep Graph Infomax. Refer to Section 3.4 for more details.

1. 从损坏函数中采样 $(\tilde{\mathbf{X}}, \widetilde{\mathbf{A}}) \sim \mathcal{C}(\mathbf{X}, \mathbf{A})$
2. 获得正负样本的 patch node-repr.,$\mathbf{H} = \mathcal{E}(\mathbf{X}, \mathbf{A}) = \left\{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_N\right\}$, $\tilde{\mathbf{H}} = \mathcal{E}(\tilde{\mathbf{X}}, \tilde{\mathbf{A}}) = \left\{\widetilde{h}_1, \widetilde{h}_2, \ldots, \widetilde{h}_M\right\}$
3. 获得全局特征表示 $\vec{s} = \mathcal{R}(\mathbf{H})$.
4. 根据方程 (108) 更新 $\mathcal{R}, \mathcal{D}, \mathcal{E}$ 参数.

**Details** 使用 PReLU, 迁移学习任务上 (transductive, Cora, Citeseer, PubMed) 使用 GCN

$$\mathcal{E}(\mathbf{X}, \mathbf{A}) = \sigma\left(\hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}\Theta\right) \tag{109}$$

在推断任务上 (inductive, Reddit) 使用 mean-aggr 和 GraphSAGE-GCN

$$\mathrm{MP}(\mathbf{X}, \mathbf{A}) = \hat{\mathbf{D}}^{-1}\hat{\mathbf{A}}\mathbf{X}\Theta \tag{110}$$

在多图任务上 (PPI) 使用三层带有 dense skip conn. 的 mean-pooling 层

$$\begin{aligned}
\mathbf{H}_1 &= \sigma\left(\mathbf{MP}_1(\mathbf{X}, \mathbf{A})\right) \\
\mathbf{H}_2 &= \sigma\left(\mathbf{MP}_2\left(\mathbf{H}_1 + \mathbf{X}\mathbf{W}_{\text{skip}}, \mathbf{A}\right)\right) \\
\mathcal{E}(\mathbf{X}, \mathbf{A}) &= \sigma\left(\mathbf{MP}_3\left(\mathbf{H}_2 + \mathbf{H}_1 + \mathbf{X}\mathbf{W}_{\text{skip}}, \mathbf{A}\right)\right)
\end{aligned} \tag{111}$$

在 Readout 函数上使用简单的 graph-mean-aggr

$$\mathcal{R}(\mathbf{H}) = \sigma\left(\frac{1}{N}\sum_{i=1}^{N}\vec{h}_i\right) \tag{112}$$

在判别器上使用简单的双线性打分函数

$$\mathcal{D}\left(\vec{h}_i, \vec{s}\right) = \sigma\left(\vec{h}_i^T\mathbf{W}\vec{s}\right) \tag{113}$$

## 12 GraphSAGE: Inductive Representation Learning on Graph

### 12.1 Embedding Generation/FP

**Idea** 在 k-hops 上逐层做 aggr.! Weisfeiler-Lehman 图同构检验的连续推广.

**Algorithm 1:** GraphSAGE embedding generation (i.e., forward propagation) algorithm

**Input** : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth $K$; weight matrices $\mathbf{W}^k, \forall k \in \{1, \dots, K\}$; non-linearity $\sigma$; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, \dots, K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

**Output**: Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

```
h_v^0 ← x_v, ∀v ∈ V ;
for k = 1...K do
    for v ∈ V do
        h_N(v)^k ← AGGREGATE_k({h_u^{k-1}, ∀u ∈ N(v)});
        h_v^k ← σ(W^k · CONCAT(h_v^{k-1}, h_N(v)^k))
    end
    h_v^k ← h_v^k / ||h_v^k||_2, ∀v ∈ V
end
z_v ← h_v^K, ∀v ∈ V
```

使用固定大小的邻域函数 $\mathcal{N}(v)$ 以使用固定大小的权重 $\boldsymbol{W}$, 本工作使用邻域上的均匀采样. (?那么非均匀或者随时间变化的采样呢? ) 为了进行图上的无监督学习, 引入 graph-loss(鼓励相邻节点具有相似的学到的表示)

$$J_{\mathcal{G}}(\mathbf{z}_u) = -\log\left(\sigma\left(\mathbf{z}_u^{\top}\mathbf{z}_v\right)\right) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log\left(\sigma\left(-\mathbf{z}_u^{\top}\mathbf{z}_{v_n}\right)\right) \tag{114}$$

这里 $\sigma$ 是 sigmoid 函数, $v$ 是从 $u$ 开始的固定长的随机游走序列上的节点, $P_n$ 是负样本分布.

### 12.2 Aggragator Selection

**Mean Aggregator** 和 GCN 不同, mean-aggr. 的 repr. 和上一层的表示 concat, 可以看作 skip-conn., 大幅改善了性能.

$$\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W} \cdot \text{MEAN}\left(\left\{\mathbf{h}_v^{k-1}\right\} \cup \left\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\right\}\right)\right) \tag{115}$$

**LSTM Aggregator** 由于 LSTM 并不是内蕴轮换不变的, 所以使用结点的随机打乱作为输入.

**Pooling Aggregator**

$$\text{AGGREGATE}_k^{\text{pool}} = \max\left(\left\{\sigma\left(\mathbf{W}_{\text{pool}}\mathbf{h}_{u_i}^k + \mathbf{b}\right), \forall u_i \in \mathcal{N}(v)\right\}\right) \tag{116}$$

注意是 MLP+max-pooling.

# 13 SGC: Simplified Graph Convolution

回顾 GCN 中的图卷积, node-wise

$$\mathbf{h}_i^{(k)} \leftarrow \frac{1}{d_i+1}\mathbf{h}_i^{(k-1)} + \sum_{j=1}^{n} \frac{a_{ij}}{\sqrt{(d_i+1)(d_j+1)}}\mathbf{h}_j^{(k-1)} \tag{117}$$

matrix-repr.

$$\begin{aligned} \mathbf{S} &= \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}} \\ \overline{\mathbf{H}}^{(k)} &\leftarrow \mathbf{S}\mathbf{H}^{(k-1)} \end{aligned} \tag{118}$$

每一层的 feat.-trans. 和最后的分类器

$$\begin{aligned} \mathbf{H}^{(k)} &\leftarrow \mathrm{ReLU}\left(\overline{\mathbf{H}}^{(k)}\Theta^{(k)}\right) \\ \hat{\mathbf{Y}}_{\mathrm{GCN}} &= \mathrm{softmax}\left(\mathbf{S}\mathbf{H}^{(K-1)}\Theta^{(K)}\right) \end{aligned} \tag{119}$$

SGC 直接在 k-hops 上聚合 (可以看作在 k-hop 连接图上聚集)

$$\hat{\mathbf{Y}}_{\mathrm{SGC}} = \mathrm{softmax}\left(\mathbf{S}^{K}\mathbf{X}\Theta\right) \tag{120}$$

这是一个凸优化问题, 可以通过二阶方法或者 SGD 来求解.

回顾 ChebNet,

$$\mathbf{U}\hat{\mathbf{G}}\mathbf{U}^{\top}\mathbf{x} \approx \sum_{i=0}^{k}\theta_i\mathbf{\Delta}^i\mathbf{x} = \mathbf{U}\left(\sum_{i=0}^{k}\theta_i{}^{i}\right)\mathbf{U}^{\top}\mathbf{x} \tag{121}$$

$$\tag{122}$$

# 14 FastGCN

## 14.1 Method

回忆 GCN

$$\tilde{H}^{(l+1)} = \hat{A}H^{(l)}W^{(l)}, \quad H^{(l+1)} = \sigma\left(\tilde{H}^{(l+1)}\right), \quad l=0,\ldots,M-1, \quad L = \frac{1}{n}\sum_{i=1}^{n} g\left(H^{(M)}(i,:)\right) \tag{123}$$

写成泛函/积分变换的形式

$$\begin{aligned} \tilde{h}^{(l+1)}(v) = \int \hat{A}(v,u)h^{(l)}(u)W^{(l)}dP(u), \quad h^{(l+1)}(v) = \sigma\left(\tilde{h}^{(l+1)}(v)\right), \quad l=0,\ldots,M-1 \\ L = \mathrm{E}_{v\sim P}\left[g\left(h^{(M)}(v)\right)\right] = \int g\left(h^{(M)}(v)\right)dP(v) \end{aligned} \tag{124}$$

把每个节点看作是 (连续 iid) 随机变量! 写成这种形式可以便利地使用 Monte-Carlo estimator 来估计, 每层使用 $t_l$ 个采样来计算

$$\tilde{h}_{t_{l+1}}^{(l+1)}(v) := \frac{1}{t_l}\sum_{j=1}^{t_l}\hat{A}\left(v,u_j^{(l)}\right)h_{t_l}^{(l)}\left(u_j^{(l)}\right)W^{(l)}, \quad h_{t_{l+1}}^{(l+1)}(v) := \sigma\left(\tilde{h}_{t_{l+1}}^{(l+1)}(v)\right), \quad l=0,\ldots,M-1 \tag{125}$$

损失的估计 (这个估计是相容的 (以 1 概率收敛至真实值))

$$L_{t_0,t_1,\ldots,t_M} := \frac{1}{t_M}\sum_{i=1}^{t_M} g\left(h_{t_M}^{(M)}\left(u_i^{(M)}\right)\right) \tag{126}$$

对于 mini-batch

$$L_{\text{batch}} = \frac{1}{t_M}\sum_{i=1}^{t_M} g\left(H^{(M)}\left(u_i^{(M)},:\right)\right) \tag{127}$$

以及每一层的 FP

$$H^{(l+1)}(v,:) = \sigma\left(\frac{n}{t_l}\sum_{j=1}^{t_l}\hat{A}\left(v,u_j^{(l)}\right)H^{(l)}\left(u_j^{(l)},:\right)W^{(l)}\right), \quad l=0,\ldots,M-1 \tag{128}$$

其中 $n$ 是图节点数量, 作为正则化系数 (从矩阵形式到积分形式).

## 14.2 Variance Reduction

**Summary** Utilize Importance Sampling, Degree Weighted.

Use notations

| | Function | Samples | Num. samples |
|---|---|---|---|
| Layer $l+1$; random variable $v$ | $\tilde{h}_{tl+1}^{(l+1)}(v)\to y(v)$ | $u_i^{(l+1)}\to v_i$ | $t_{l+1}\to s$ |
| Layer $l$; random variable $u$ | $h_{t_l}^{(l)}(u)W^{(l)}\to x(u)$ | $u_j^{(l)}\to u_j$ | $t_l\to t$ |

(129)

consider layer repr.

$$G := \frac{1}{s}\sum_{i=1}^{s} y\left(v_i\right) = \frac{1}{s}\sum_{i=1}^{s}\left(\frac{1}{t}\sum_{j=1}^{t}\hat{A}\left(v_i,u_j\right)x\left(u_j\right)\right) \tag{130}$$

compute it's variance

$$\operatorname{Var}\{G\} = R + \frac{1}{st}\iint \hat{A}(v,u)^2 x(u)^2 dP(u)dP(v) \tag{131}$$

$$\text{where } R = \frac{1}{s}\left(1-\frac{1}{t}\right)\int e(v)^2 dP(v) - \frac{1}{s}\left(\int e(v)dP(v)\right)^2, e(v) = \int \hat{A}(v,u)x(u)dP(u) \tag{132}$$

第一项很难再优化, 由于取决于下一层的采样. 优化第二项, 引入新的采样分布, 则为了保持 $G$ 期望不变

$$y_Q(v) := \frac{1}{t}\sum_{j=1}^{t}\hat{A}\left(v,u_j\right)x\left(u_j\right)\left(\left.\frac{dP(u)}{dQ(u)}\right|_{u_j}\right), \quad u_1,\ldots,u_t\sim Q \tag{133}$$

此时

$$G_Q := \frac{1}{s}\sum_{i=1}^{s} y_Q\left(v_i\right) = \frac{1}{s}\sum_{i=1}^{s}\left(\frac{1}{t}\sum_{j=1}^{t}\hat{A}\left(v_i,u_j\right)x\left(u_j\right)\left(\left.\frac{dP(u)}{dQ(u)}\right|_{u_j}\right)\right) \tag{134}$$

**Theorem** 当

$$dQ(u) = \frac{b(u)|x(u)|dP(u)}{\int b(u)|x(u)|dP(u)} \quad \text{where} \quad b(u) = \left[\int \hat{A}(v,u)^2 dP(v)\right]^{\frac{1}{2}} \tag{135}$$

时, 方差最小, 为

$$\mathrm{Var}\left\{G_{Q}\right\}=R+\frac{1}{st}\left[\int b(u)|x(u)|dP(u)\right]^{2} \tag{136}$$

然而实际上 $|x(u)|$ 会变化且难以计算, 直接用 (... 那你论证半天为个啥...)

$$dQ(u)=\frac{b(u)^{2}dP(u)}{\int b(u)^{2}dP(u)} \tag{137}$$

MC 形式

$$q(u)=\|\hat{A}(:,u)\|^{2}/\sum_{u'\in V}\left\|\hat{A}\left(:,u'\right)\right\|^{2},\quad u\in V \tag{138}$$

即和节点度数正比, 此时的每一层 FP 公式

$$H^{(l+1)}(v,:)=\sigma\left(\frac{1}{t_{l}}\sum_{j=1}^{t_{l}}\frac{\hat{A}\left(v,u_{j}^{(l)}\right)H^{(l)}\left(u_{j}^{(l)},:\right)W^{(l)}}{q\left(u_{j}^{(l)}\right)}\right),\quad u_{j}^{(l)}\sim q,\quad l=0,\ldots,M-1 \tag{139}$$

# 15 GWNN: Wavelet Transform on Graph

## 15.1 Supplementary Math: Real and Complex Wavelets

Function $\psi\in L^{2}(\mathbb{R})$ called **orthogonal wavelet**, if it could be used to define a orthogonal complete basis of Hilbert space $L^{2}(\mathbb{R})$. Given $\psi$, the basis are

$$\psi_{jk}(x)=2^{\frac{j}{2}}\psi\left(2^{j}x-k\right) \tag{140}$$

under normal inner-product on $L^{2}(\mathbb{R})$, it's orthogonal

$$\begin{aligned}\left\langle\psi_{jk},\psi_{lm}\right\rangle&=\int_{-\infty}^{\infty}\psi_{jk}(x)\overline{\psi_{lm}(x)}dx\\&=\delta_{jl}\delta_{km}\end{aligned} \tag{141}$$

**Integral Wavelet Transform**

$$\left[W_{\psi}f\right](a,b)=\frac{1}{\sqrt{|a|}}\int_{-\infty}^{\infty}\overline{\psi\left(\frac{x-b}{a}\right)}f(x)dx \tag{142}$$

wavelet coefficient given by

$$c_{jk}=\left[W_{\psi}f\right]\left(2^{-j},k2^{-j}\right) \tag{143}$$

**Meyer Wavelet** in frequency-domain defined

$$\Psi(\omega):=\begin{cases}\frac{1}{\sqrt{2\pi}}\sin\left(\frac{\pi}{2}\nu\left(\frac{3|\omega|}{2\pi}-1\right)\right)e^{j\omega/2} & \text{if } 2\pi/3<|\omega|<4\pi/3\\\frac{1}{\sqrt{2\pi}}\cos\left(\frac{\pi}{2}\nu\left(\frac{3|\omega|}{4\pi}-1\right)\right)e^{j\omega/2} & \text{if } 4\pi/3<|\omega|<8\pi/3\\0 & \text{otherwise}\end{cases} \tag{144}$$

where(standard impl.)

$$\nu(x) := \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } 0 < x < 1 \\ 1 & \text{if } x > 1 \end{cases} \tag{145}$$

it can also be

$$\nu(x) := \begin{cases} x^4\left(35 - 84x + 70x^2 - 20x^3\right) & \text{if } 0 < x < 1 \\ 0 & \text{otherwise.} \end{cases} \tag{146}$$

in time-domain a close form is obtained

$$\phi(t) = \begin{cases} \frac{2}{3} + \frac{4}{3\pi} & t = 0 \\ \frac{\sin\left(\frac{2\pi}{3}t\right) + \frac{4}{3}t\cos\left(\frac{4\pi}{3}t\right)}{\pi t - \frac{16\pi}{9}t^3} & \text{otherwise} \end{cases} \tag{147}$$

and $\psi(t) = \psi_1(t) + \psi_2(t)$,

$$\begin{aligned} \psi_1(t) &= \frac{\frac{4}{3\pi}\left(t-\frac{1}{2}\right)\cos\left[\frac{2\pi}{3}\left(t-\frac{1}{2}\right)\right] - \frac{1}{\pi}\sin\left[\frac{4\pi}{3}\left(t-\frac{1}{2}\right)\right]}{\left(t-\frac{1}{2}\right) - \frac{16}{9}\left(t-\frac{1}{2}\right)^3} \\ \psi_2(t) &= \frac{\frac{8}{3\pi}\left(t-\frac{1}{2}\right)\cos\left[\frac{8\pi}{3}\left(t-\frac{1}{2}\right)\right] + \frac{1}{\pi}\sin\left[\frac{4\pi}{3}\left(t-\frac{1}{2}\right)\right]}{\left(t-\frac{1}{2}\right) - \frac{64}{9}\left(t-\frac{1}{2}\right)^3} \end{aligned} \tag{148}$$

**Mexican Hat Wavelet** 1d-form Ricker Wavelet, 2nd deriv. of Gaussian dist.

$$\psi(t) = \frac{2}{\sqrt{3\sigma}\pi^{1/4}}\left(1 - \left(\frac{t}{\sigma}\right)^2\right)e^{-\frac{t^2}{2\sigma^2}} \tag{149}$$

2d-form Marr Wavelet

$$\psi(x, y) = \frac{1}{\pi\sigma^4}\left(1 - \frac{1}{2}\left(\frac{x^2+y^2}{\sigma^2}\right)\right)e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{150}$$

**Morlet Wavelet**

$$\Psi_\sigma(t) = c_\sigma \pi^{-\frac{1}{4}} e^{-\frac{1}{2}t^2}\left(e^{i\sigma t} - \kappa_\sigma\right) \tag{151}$$

scale factor

$$c_\sigma = \left(1 + e^{-\sigma^2} - 2e^{-\frac{3}{4}\sigma^2}\right)^{-\frac{1}{2}} \tag{152}$$

## 15.2 Graph Wavelets

定义一系列图上的小波 $\psi_s = \{\psi_{si}\}$,$\psi_{si}$ 代表以结点 i 为中心, 尺度为 s 的小波, 数学上可以写成

$$\psi_s = \boldsymbol{U}\boldsymbol{G}_s\boldsymbol{U}^\top \tag{153}$$

其中 $\boldsymbol{U}$ 是拉普拉斯矩阵的特征向量, $\boldsymbol{G}_s = \operatorname{diag}\left(g\left(s\lambda_1\right), \ldots, g\left(s\lambda_n\right)\right), g(s\lambda_i) = e^{s\lambda_i}$(... 就这? 这不是说 $\boldsymbol{G}_s = \exp(s\boldsymbol{\Lambda})$?) 图上的小波变换

$$\hat{\boldsymbol{x}} = \psi_s^{-1}x \tag{154}$$

小波基的卷积

$$\boldsymbol{x} *_{\mathcal{G}} \boldsymbol{y} = \psi_s\left(\left(\psi_s^{-1}\boldsymbol{y}\right) \odot \left(\psi_s^{-1}\boldsymbol{x}\right)\right) \tag{155}$$

### 15.3 GWNN

GWNN Layer

$$\boldsymbol{X}_{[:,j]}^{m+1} = h\left(\psi_s \sum_{i=1}^{p} \boldsymbol{F}_{i,j}^{m} \psi_s^{-1} \boldsymbol{X}_{[:,i]}^{m}\right) \quad j = 1, \cdots, q \tag{156}$$

in node-wise favor

$$\boldsymbol{x}_j^{m+1} = h\left(\psi_s \sum_{i=1}^{p} \boldsymbol{F}_{i,j}^{m} \psi_s^{-1} \boldsymbol{x}_i^{m}\right) \quad j = 1, \cdots, q \tag{157}$$

where $\boldsymbol{F}$ is diagonal. On inductive missons(Cora etc.), 使用两层 (ReLU,softmax)GWNN. Parameters $O(npq)$, bad! Detach feat. trans. and graph conv.(as if GCN)

$$\boldsymbol{X}^{m+1} = h\left(\psi_s \boldsymbol{F}^m \psi_s^{-1} \boldsymbol{X}^m \boldsymbol{W}\right) \tag{158}$$

**Advantages**

1. 高效性: 小波基可以通过快速方法得到 (Chebyshev 估计,m 阶对应复杂度 $O(m|E|)$, 无需昂贵的 EVD).
2. 高稀疏性.
3. 局部化卷积.
4. 可变的邻域.

#### 15.3.1 Details

1. $\boldsymbol{F}$ 是一个对角阵 (特征向量的滤波器)
2. 只用了一个尺度 (严格地说是两个 $s, -s$), 核函数是 heat kernel:$e^{-t}$
3. 可以用`pygsp`包的内建函数来计算 Chebyshev 系数
   - `pygsp.filters.approximations.compute_cheby_coeff(filter, order)`
   - `pygsp.filters.approximations.cheby_op(G, c, signal)`
4. 源代码里用了一个 trick, 即在 $N \times N$ 单位阵上应用`cheby_op(G, c, I)`来得到 $\psi_s$ 的稀疏表示. 最后还用 L1 范数归一化.
5. Shapes: $\boldsymbol{X}^m, \boldsymbol{X}^{m'} \in \mathbb{R}^{N\times F}, \psi_s \in \mathbb{R}^{F\times N}, \psi_s^{-1} \in \mathbb{R}^{N\times F}, \boldsymbol{F} = \mathrm{Diag}(\boldsymbol{f}) \in \mathbb{R}^{N\times N}$,

## 16 Graph Wavelets

### 16.1 经典小波变换/CWT

小波

$$\psi_{s,a}(x) = \frac{1}{s}\psi\left(\frac{x-a}{s}\right) \tag{159}$$

(经典) 小波变换/CWT

$$W_f(s,a)=\int_{-\infty}^{\infty}\frac{1}{s}\psi^*\left(\frac{x-a}{s}\right)f(x)dx \tag{160}$$

可逆, 若满足 admissibility cond.

$$\int_0^{\infty}\frac{|\hat{\psi}(\omega)|^2}{\omega}d\omega=C_\psi<\infty \tag{161}$$

逆变换/IWT

$$f(x)=\frac{1}{C_\psi}\int_0^{\infty}\int_{-\infty}^{\infty}W_f(s,a)\psi_{s,a}(x)\frac{dads}{s} \tag{162}$$

定义算子

$$T^s f(a)=W_f(s,a) \tag{163}$$

有

$$\bar{\psi}_s(x)=\frac{1}{s}\psi^*\left(\frac{-x}{s}\right) \tag{164}$$

则有

$$\begin{aligned}\left(T^s f\right)(a)&=\int_{-\infty}^{\infty}\frac{1}{s}\psi^*\left(\frac{x-a}{s}\right)f(x)dx=\int_{-\infty}^{\infty}\bar{\psi}_s(a-x)f(x)dx\\&=\left(\bar{\psi}_s\star f\right)(a)\end{aligned} \tag{165}$$

频域上有

$$\widehat{T^s f}(\omega)=\hat{\bar{\psi}}_s(\omega)\hat{f}(\omega) \tag{166}$$

以及

$$\hat{\bar{\psi}}_s(\omega)=\hat{\psi}^*(s\omega) \tag{167}$$

那么

$$\left(T^s f\right)(x)=\frac{1}{2\pi}\int_{-\infty}^{\infty}e^{i\omega x}\hat{\psi}^*(s\omega)\hat{f}(\omega)d\omega \tag{168}$$

## 16.2 谱小波变换/SGWT

SGWT 核 $g\Rightarrow T_g=g(\mathcal{L})$, 有频谱

$$\widehat{T_g f}(\ell)=g\left(\lambda_\ell\right)\hat{f}(\ell) \tag{169}$$

使用 IFT

$$\left(T_g f\right)(m)=\sum_{\ell=0}^{N-1}g\left(\lambda_\ell\right)\hat{f}(\ell)\chi_\ell(m) \tag{170}$$

局域化的图小波 $\psi_{t,n}=T_g^t\delta_n$, 展开得

$$\psi_{t,n}(m)=\sum_{\ell=0}^{N-1}g\left(t\lambda_\ell\right)\chi_\ell^*(n)\chi_\ell(m) \tag{171}$$

小波系数

$$W_f(t,n)=\left(T_g^t f\right)(n)=\sum_{\ell=0}^{N-1}g\left(t\lambda_\ell\right)\hat{f}(\ell)\chi_\ell(n) \tag{172}$$

#### 16.2.1 Scaling Functions

小波都和第一特征向量 $\chi_0$ 正交, 并和特征值接近 0 的 eig-vec 几乎正交. 于是引入尺度函数, 类似地通过一个函数 $h:\mathbb{R}^+ \to \mathbb{R}$ 定义, 满足 $h(0)=0, h(\infty)=0$, $\phi_n = T_h\delta_n = h(\mathcal{L})\delta_n$, 系数 $S_f(n)=\langle \phi_n, f\rangle$ .

将会在之后看到, 当 $G(\lambda)=h(\lambda)^2+\sum_{j=1}^{J} g\left(t_j\lambda\right)^2$ 有界且离开 0 时, 可以达到稳定近似.

### 16.3 SGWT 的性质

#### 16.3.1 Inverse SGWT

**Lemma** 若 SGWT 核满足 admissibility cond.

$$\int_0^\infty \frac{g^2(x)}{x}dx = C_g < \infty \tag{173}$$

且 $g(0)=0$, 则

$$\frac{1}{C_g}\sum_{n=1}^{N}\int_0^\infty W_f(t,n)\psi_{t,n}(m)\frac{dt}{t} = f^{\#}(m) \tag{174}$$

且

$$f = f^{\#} + \hat{f}(0)\chi_0 \tag{175}$$

#### 16.3.2 局域性

**Lemma** 定义 $d_G(m,n)$ 为结点最短路径长度 (不考虑边权). 若 $d_G(m,n) > s$, $(\mathcal{L}^s)_{m,n} = 0$

**Lemma** Let $\psi_{t,n} = T_g^t\delta_n$ and $\tilde{\psi}_{t,n} = T_{\tilde{g}}^t\delta_n$ be the wavelets at scale t generated by the kernels $g$ and $\tilde{g}$. If $|g(t\lambda)-\tilde{g}(t\lambda)| \leqslant M(t)$ for all $\lambda \in [0, \lambda_{N-1}]$ , then $\left|\psi_{t,n}(m)-\tilde{\psi}_{t,n}(m)\right| \leqslant M(t)$ for each vertex m. Additionally, $\left\|\psi_{t,n}-\tilde{\psi}_{t,n}\right\|_2 \leqslant \sqrt{N}M(t)$

**Lemma** Let g be $K+1$ times continuously differentiable, satisfying $g(0)=0, g^{(r)}(0)=0$ for all $r<K$, and $g^{(K)}(0)=C\neq 0$ Assume that there is some $t'>0$ such that $\left|g^{(K+1)}(\lambda)\right| \leqslant B$ for all $\lambda\in[0,t'\lambda_{N-1}]$. Then, for $\tilde{g}(t\lambda)=(C/K!)(t\lambda)^K$ we have

$$M(t) = \sup_{\lambda\in[0,\lambda_{N-1}]} |g(t\lambda)-\tilde{g}(t\lambda)| \leqslant t^{K+1}\frac{\lambda_{N-1}^{K+1}}{(K+1)!}B$$

for all $t<t'$

**Theorem** Let G be a weighted graph with Laplacian $\mathcal{L}$. Let $g$ be a kernel satisfying the hypothesis of Lemma 5.4, with constants $t'$ and B. Let m and n be vertices of G such that $d_G(m,n)>K$. Then there exist constants $D$ and $t''$, such that

$$\frac{\psi_{t,n}(m)}{\|\psi_{t,n}\|} \leqslant Dt$$

for all $t<\min\left(t', t''\right)$

#### 16.3.3 Spectral Wavelet Frames

使用中必然使用 J 个 t 的离散采样, 导致 NJ 个小波和 N 个伸缩函数 (尺度函数). 我们称一个在离散化的尺度上的小波为一个帧. 一些 Hilbert 空间上的向量组成的帧 $\Gamma_k \in \mathcal{H}$, 不等式

$$A\|f\|^2 \leqslant \sum_k |\langle f, \Gamma_k\rangle|^2 \leqslant B\|f\|^2$$

控制了数值稳定性.

**Theorem** Given a set of scales $\{t_j\}_{j=1}^J$, the set $F = \{\phi_n\}_{n=1}^N \cup \{\psi_{t_j,n}\}_{j=1}^J N_{n=1}$ forms a frame with bounds $A, B$ given by

$$A = \min_{\lambda\in[0,\lambda_{N-1}]} G(\lambda)$$
$$B = \max_{\lambda\in[0,\lambda_{N-1}]} G(\lambda)$$

where $G(\lambda) = h^2(\lambda) + \sum_j g(t_j\lambda)^2$

### 16.4 Fast SGWT Approximation by Polynomials

**Lemma** (多项式逼近的有效性)Let $\lambda_{\max} \geqslant \lambda_{N-1}$ be any upper bound on the spectrum of $\mathcal{L}$. For fixed $t > 0$, let $p(x)$ be a polynomial approximant of $g(tx)$ with $L_\infty$ error $B = \sup_{x\in[0,\lambda_{\max}]} |g(tx) - p(x)|$. Then the approximate wavelet coefficients $\tilde{W}_f(t,n) = (p(\mathcal{L})f)_n$ satisfy

$$\left|W_f(t,n) - \tilde{W}_f(t,n)\right| \leqslant B\|f\|$$

获得这么一个估计在使用时往往需要知道一个特征值上界的估计 $\lambda_{max}$, 但这是个很容易的问题, 只需要做一些矩阵-向量乘法即可, 比如 Arnoldi 迭代或者 Jacobi-Davidson 算法.

使用 Chebyshev 多项式逼近: 由数值分析得知 Chebyshev 时同阶多项式逼近性能最好的.

$$T_0(\lambda) = 1, T_1(\lambda) = \lambda, \tag{176}$$
$$T_j(\lambda) = 2\lambda T_{j-1}(\lambda) - T_{j-2}(\lambda) \tag{177}$$
$$T_n(x) = \cos(n\,\mathrm{acos}(x)) \tag{178}$$

使用变换 $x = a(y+1), a = \lambda_{max}/2$ 来把 x 变换到 $[-1,1]$ 上. 假设使用一系列离散化的尺度 $t_n$, 记偏移的 CP $\bar{T}(x) = T_k\left(\frac{x-a}{a}\right)$, 可写

$$g(t_n x) = \frac{1}{2}c_{n,0} + \sum_{k=1}^{\infty} c_{n,k}\bar{T}_k(x) \tag{179}$$

系数

$$c_{n,k} = \frac{2}{\pi}\int_0^\pi \cos(k\theta) g(t_n(a(\cos(\theta)+1)))\,d\theta \tag{180}$$

(简单的数值积分即可, 计算快速). 对于任何尺度系 $t_j$, 截断级数到 $M_j$ 项来逼近核函数 $g$, 我们有

估计

$$\tilde{W}_f\left(t_j, n\right)=\left(\frac{1}{2} c_{j,0} f+\sum_{k=1}^{M_j} c_{j,k} \bar{T}_k(\mathcal{L}) f\right)_n \tag{181}$$

$$\Rightarrow \tilde{W}_f\left(t_j, :\right)=\frac{1}{2} c_{j,0} f+\sum_{k=1}^{M_j} c_{j,k} \bar{T}_k(\mathcal{L}) f \tag{182}$$

$$\tilde{S}_f(n)=\left(\frac{1}{2} c_{0,0} f+\sum_{k=1}^{M_0} c_{0,k} \bar{T}_k(\mathcal{L}) f\right)_n \tag{183}$$

$$\Rightarrow \tilde{S}_f=\frac{1}{2} c_{0,0} f+\sum_{k=1}^{M_0} c_{0,k} \bar{T}_k(\mathcal{L}) f \tag{184}$$

$$\text{with efficient comp. of } \bar{T}_k(\mathcal{L}) f=\frac{2}{a}(\mathcal{L}-I)\left(\bar{T}_{k-1}(\mathcal{L}) f\right)-\bar{T}_{k-2}(\mathcal{L}) f \tag{185}$$

#### 16.4.1 Fast Approximation of Adjoint

可以认为 $W: \mathbb{R}^N \rightarrow \mathbb{R}^{N(J+1)}$ 是一个线性变换, 且 $Wf=\left(\left(T_h f\right)^T,\left(T_g^{t_1} f\right)^T, \ldots,\left(T_g^{t_J} f\right)^T\right)^T$, 考虑其多项式估计 $\tilde{W}=\left(\left(p_0(\mathcal{L}) f\right)^T,\left(p_1(\mathcal{L}) f\right)^T, \ldots,\left(p_J(\mathcal{L}) f\right)^T\right)^T$, 这里展示其伴随算子的快速近似算法. 有

$$\begin{aligned}
\langle\eta, W f\rangle_{N(J+1)} &=\left\langle\eta_0, T_h f\right\rangle+\sum_{j=1}^J\left\langle\eta_j, T_g^{t_j} f\right\rangle_N \\
&=\left\langle T_h^* \eta_0, f\right\rangle+\left\langle\sum_{j=1}^J\left(T_g^{t_j}\right)^* \eta_j, f\right\rangle_N=\left\langle T_h \eta_0+\sum_{j=1}^J T_g^{t_j} \eta_j, f\right\rangle_N
\end{aligned} \tag{186}$$

这表明

$$W^* \eta=T_h \eta_0+\sum_{j=1}^J T_g^{t_j} \eta_n \tag{187}$$

为了计算逆变换 (伪逆 $L=\left(W^* W\right)^{-1} W^*$ 是差异 norm 最小的逆变换), 计算变换和其伴随算子的乘积

$$\tilde{W}^* \tilde{W} f=\sum_{j=0}^J p_j(\mathcal{L})\left(p_j(\mathcal{L}) f\right)=\left(\sum_{j=0}^J\left(p_j(\mathcal{L})\right)^2\right) f \tag{188}$$

记 $P(x)=\sum_{j=0}^J\left(p_j(x)\right)^2=\frac{1}{2} d_0+\sum_{k=1}^{M^*} d_k \bar{T}_k(x), M^* \max \left(M_j\right)$, 有公式

$$T_k(x) T_l(x)=\frac{1}{2}\left(T_{k+l}(x)+T_{|k-l|}(x)\right) \tag{189}$$

设 $c_{j,k}^{\prime}=c_{j,k}$ for $k \geqslant 1$ and $c_{j,0}^{\prime}=\frac{1}{2} c_{j,0}$ 有

$$d_{j,k}^{\prime}=\begin{cases}
\frac{1}{2}\left(c_{j,0}^{\prime} 2+\sum_{i=0}^{M_n} c_{j,i}^{\prime} 2\right) & \text{if } k=0 \\
\frac{1}{2}\left(\sum_{i=0}^k c_{j,i}^{\prime} c_{j,k-i}^{\prime}+\sum_{i=0}^{M_j-k} c_{j,i}^{\prime} c_{j,k+i}^{\prime}+\sum_{i=k}^{M_j} c_{j,i}^{\prime} c_{j,i-k}^{\prime}\right) & \text{if } 0<k \leqslant M_j \\
\frac{1}{2}\left(\sum_{i=k-M_j}^{M_j} c_{j,i}^{\prime} c_{j,k-i}^{\prime}\right) & \text{if } M_j<k \leqslant 2M_j
\end{cases} \tag{190}$$

设

$$d_{n,0}=2d'_{j,0} \text{ and } d_{j,k}=d'_{j,k} \text{ for } k\geqslant 1, \text{ and setting } d_k=\sum_{j=0}^{J} d_{j,k} \tag{191}$$

则有

$$\tilde{W}^*\tilde{W}f=P(\mathcal{L})f=\frac{1}{2}d_0 f+\sum_{k=1}^{M^*} d_k \bar{T}_k(\mathcal{L})f \tag{192}$$

#### 16.4.2 Inverse Calculation

使用伪逆 $L=(W^*W)^{-1}W^*$, 给定小波系数 c, 可以通过方程

$$(W^*W)f=W^*c \tag{193}$$

计算原信号. 直接解是困难的, 使用快速共轭梯度法, 或者经典的帧算法 (frame algorithm).

### 16.5 Implementations and Details

A good example:

$$g(x;\alpha,\beta,x_1,x_2)=\begin{cases} x_1^{\alpha}x^{-\alpha} & \text{for } x<x_1 \\ s(x) & \text{for } x_1\leqslant x\leqslant x_2 \\ x_2^{\beta}x^{-\beta} & \text{for } x>x_2 \end{cases} \tag{194}$$

其中 $s(x)$ 是三次样条. 伸缩函数 $h(x)=h(0)\exp\left(-\left(\frac{x}{0.6\lambda \min}\right)^4\right)$, 尺度按从小到大对数线性间隔选取, $t_1=x_2/\lambda_{\min}, t_I=x_2/\lambda_{\max}, \lambda_{\min}=\lambda_{\max}/K$

# 17 GMNN: Graph Markov Neural Network

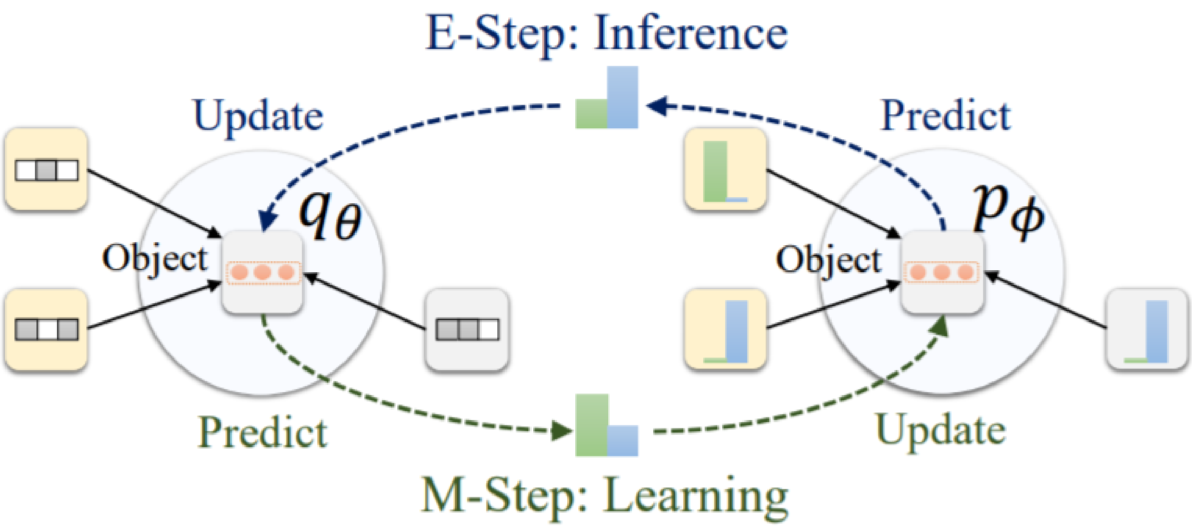



*Figure 1.* Framework overview. Yellow and grey squares are labeled and unlabeled objects. Grey/white grids are attributes. Histograms are label distributions of objects. Orange triple circles are object representations. GMNN is trained by alternating between an E-step and an M-step. See Sec. 4.4 for the detailed explanation.

**Idea** 使用统计关系学习 (SRL) 建模

$$p\left(\mathbf{y}_V \mid \mathbf{x}_V\right)=\frac{1}{Z\left(\mathbf{x}_V\right)} \prod_{\left(n_i, n_j\right) \in E} \psi_{i, j}\left(\mathbf{y}_{n_i}, \mathbf{y}_{n_j}, \mathbf{x}_V\right) \tag{195}$$

而 GNN 模型则忽略 labels 之间的关系

$$p\left(\mathbf{y}_V \mid \mathbf{x}_V\right)=\prod_{n \in V} p\left(\mathbf{y}_n \mid \mathbf{x}_V\right) \tag{196}$$

具体上讲, GMNN 使用一个条件随机场 (CRF)+ 平均场近似 (mean-field approx.) 来建模, 并用 EM 算法来优化.

## 17.1 Psedolikelihood Variaional EM

优化 ELBO

$$\begin{aligned} & \log p_\phi\left(\mathbf{y}_L \mid \mathbf{x}_V\right) \geq \\ & \mathbb{E}_{q_\theta\left(\mathbf{y}_U \mid \mathbf{x}_V\right)}\left[\log p_\phi\left(\mathbf{y}_L, \mathbf{y}_U \mid \mathbf{x}_V\right)-\log q_\theta\left(\mathbf{y}_U \mid \mathbf{x}_V\right)\right] \end{aligned} \tag{197}$$

这里 $q_\theta$ 是任意分布, 当

$$q_\theta(\mathbf{y}_U \mid \mathbf{x}_V) = p_\phi(\mathbf{y}_U \mid \mathbf{y}_L, \mathbf{x}_V) \tag{198}$$

取等号. 使用经典的 EM 算法来学习! 然而 $p_\phi$ 中的配分函数难以计算, 使用以下 psedo-ld

$$\begin{aligned} \ell_{PL}(\phi) &\triangleq \mathbb{E}_{q_\theta(\mathbf{y}_U \mid \mathbf{x}_V)}\left[\sum_{n \in V} \log p_\phi\left(\mathbf{y}_n \mid \mathbf{y}_{V \setminus n}, \mathbf{x}_V\right)\right] \\ &= \mathbb{E}_{q_\theta(\mathbf{y}_U \mid \mathbf{x}_V)}\left[\sum_{n \in V} \log p_\phi\left(\mathbf{y}_n \mid \mathbf{y}_{\mathrm{NB(n)}}, \mathbf{x}_V\right)\right] \end{aligned} \tag{199}$$

以上伪似然函数广泛应用于 Markov 学习中.

## 17.2 Inference

这一步设计计算后验分布 $p_\phi(\mathbf{y}_U \mid \mathbf{y}_L, \mathbf{x}_V)$, 但这是困难的, 使用另一个变分分布来计算, 并使用平均场近似

$$q_\theta(\mathbf{y}_U \mid \mathbf{x}_V) = \prod_{n \in U} q_\theta(\mathbf{y}_n \mid \mathbf{x}_V) \tag{200}$$

使用一个 GNN 来参数化上述公式的每一项

$$q_\theta(\mathbf{y}_n \mid \mathbf{x}_V) = \mathrm{MLP}[\mathrm{Cat}(\mathbf{y}_n \mid \mathrm{softmax}(W_\theta \mathbf{h}_{\theta,n}))] \tag{201}$$

根据平均场近似, 最优值为

$$\begin{aligned} &\log q^*(\mathbf{y}_n \mid \mathbf{x}_V) = \\ &\mathbb{E}_{q_\theta\left(\mathbf{y}_{\mathrm{NB}(n) \cap U} \mid \mathbf{x}_V\right)}\left[\log p_\phi\left(\mathbf{y}_n \mid \mathbf{y}_{\mathrm{NB}(n)}, \mathbf{x}_V\right)\right] + \text{const.} \end{aligned} \tag{202}$$

使用 Monte-Carlo 估计

$$\begin{aligned} &\mathbb{E}_{q_\theta\left(\mathbf{y}_{\mathrm{NB}(n) \cap U} \mid \mathbf{x}_V\right)}\left[\log p_\phi\left(\mathbf{y}_n \mid \mathbf{y}_{\mathrm{NB}(n)}, \mathbf{x}_V\right)\right] \\ &\simeq \log p_\phi\left(\mathbf{y}_n \mid \hat{\mathbf{y}}_{\mathrm{NB}(n)}, \mathbf{x}_V\right) \end{aligned} \tag{203}$$

其中 $\hat{\mathbf{y}}_{\mathrm{NB}(n)} = \{\hat{\mathbf{y}}_{n'}\}_{n' \in \mathrm{NB}(n)}$, 且对于任何 unlabeled neighbors, 使用采样的标签 $\hat{\mathbf{y}}_{n'} \sim q_\theta(\mathbf{y}_{n'} \mid \mathbf{x}_V)$, 实践中发现只取一个 (unlabeled) 样本几乎和取很多样本效果相当 (!), 效率考虑只取一个, 综上,

$$q^*(\mathbf{y}_n \mid \mathbf{x}_V) \approx p_\phi\left(\mathbf{y}_n \mid \hat{\mathbf{y}}_{\mathrm{NB}(n)}, \mathbf{x}_V\right) \tag{204}$$

那么我们可以把后者作为 (最大化) 目标, 然后最小化 KL 散度

$$\mathrm{KL}\left(p_\phi\left(\mathbf{y}_n \mid \hat{\mathbf{y}}_{\mathrm{NB}(n)}, \mathbf{x}_V\right) \| q_\theta(\mathbf{y}_n \mid \mathbf{x}_V)\right) \tag{205}$$

进一步还是用并行更新策略, 独立的为每个 unlabeled node 优化

$$O_{\theta,U} = \sum_{n \in U} \mathbb{E}_{p_\phi\left(\mathbf{y}_n \mid \hat{\mathbf{y}}_{\mathrm{NB}(n)}, \mathbf{x}_V\right)}\left[\log q_\theta(\mathbf{y}_n \mid \mathbf{x}_V)\right] \tag{206}$$

以及在 labeled node 上优化

$$O_{\theta,L} = \sum_{n \in L} \log q_\theta(\mathbf{y}_n \mid \mathbf{x}_V) \tag{207}$$

最终的 loss

$$O_\theta = O_{\theta,U} + O_{\theta,L} \tag{208}$$

**Algorithm 1** Optimization Algorithm

**Input:** A graph $G$, some labeled objects $(L, \mathbf{y}_L)$.
**Output:** Object labels $\mathbf{y}_U$ for unlabeled objects $U$.
Pre-train $q_\theta$ with $\mathbf{y}_L$ according to Eq. (11).
**while** not converge **do**
 ⊡ *M-Step: Learning Procedure*
 Annotate unlabeled objects with $q_\theta$.
 Denote the sampled labels as $\hat{\mathbf{y}}_U$.
 Set $\hat{\mathbf{y}}_V = (\mathbf{y}_L, \hat{\mathbf{y}}_U)$ and update $p_\phi$ with Eq. (14).
 ⊡ *E-Step: Inference Procedure*
 Annotate unlabeled objects with $p_\phi$ and $\hat{\mathbf{y}}_V$.
 Denote the predicted label distribution as $p_\phi(\mathbf{y}_U)$.
 Update $q_\theta$ with Eq. (10), (11) based on $p_\phi(\mathbf{y}_U), \mathbf{y}_L$.
**end while**
Classify each unlabeled object $n$ based on $q_\theta(\mathbf{y}_n|\mathbf{x}_V)$.

## 17.3 Learning

直接使用 GNN 来建模, 而非使用势函数

$$p_\phi\left(\mathbf{y}_n \mid \mathbf{y}_{\mathrm{NB}(n)}, \mathbf{x}_V\right) = \operatorname{Cat}\left(\mathbf{y}_n \mid \operatorname{softmax}\left(W_\phi \mathbf{h}_{\phi,n}\right)\right) \tag{209}$$

还可以使用 SRL 中的 techniques, 同时把 $\boldsymbol{y}_{NB(n)}, \boldsymbol{x}_{NB(n)}$ 送到 GNN 中作为 in-feature. 最终的优化目标

$$O_\phi = \sum_{n \in V} \log p_\phi\left(\hat{\mathbf{y}}_n \mid \hat{\mathbf{y}}_{\mathrm{NB(n)}}, \mathbf{x}_V\right) \tag{210}$$

## 17.4 Optimization

在有标签数据上预训练 $q_\theta$, 再 EM. 最后用 $q_\theta$ 来预测 (往往比用 $p_\phi$ 准确率高)

| | GCN [9] | Vanilla SGD | GraphSAGE [5] | FastGCN [1] | VR-GCN [2] |
|---|---|---|---|---|---|
| Time complexity | $O\left(L\|A\|_0 F + LNF^2\right)$ | $O\left(d^L NF^2\right)$ | $O\left(r^L NF^2\right)$ | $O\left(rLNF^2\right)$ | $O\left(L\|A\|_0 F + LNF^2 + r^L NF^2\right)$ |
| Memory complexity | $O\left(LNF + LF^2\right)$ | $O\left(bd^L F + LF^2\right)$ | $O\left(br^L F + LF^2\right)$ | $O\left(brLF + LF^2\right)$ | $O\left(LNF + LF^2\right)$ |

# 18 ClusterGCN: Fast Deep & Large GCNs

## 18.1 Vanilla ClusterGCN: Cluster For Batch

GCN 需要整个 epoch 来更新一次梯度, 使用 mini-batch SGD 可能可以增加性能, 为此使用 batch-estimator

$$\frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\nabla \operatorname{loss}\left(y_i, z_i^{(L)}\right) \tag{211}$$

估计 loss. 但这会增加一整个 epoch 的计算时间, SGD 导致了 node-repr. 聚合了 $O(d^L)$ 个邻居的信息, 导致 BP 复杂度很高. 为此定义 embedding utilization(嵌入效用), 为一个节点的表示在 BP 中被重复利用的次数. 在 GCN 中很高, 每层都为 $d$, 但是在 GraphSAGE/FastGCN 中是一个很低的常数, 由于 k-hops 很难重叠.

为此, 考虑到一个 batch 的 emb. util. 是其中的边数 $\|A_{\mathcal{B},\mathcal{B}}\|_0$, 故提出想法: 每次取出边数最大的 (导出) 子图. 对于一个图 $G$, 有分割

$$\bar{G} = [G_1, \cdots, G_c] = [\{\mathcal{V}_1, \mathcal{E}_1\}, \cdots, \{\mathcal{V}_c, \mathcal{E}_c\}] \tag{212}$$

据此, 有

$$A = \bar{A} + \Delta = \begin{bmatrix} A_{11} & \cdots & A_{1c} \\ \vdots & \ddots & \vdots \\ A_{c1} & \cdots & A_{cc} \end{bmatrix} \tag{213}$$

以及

$$\bar{A} = \begin{bmatrix} A_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A_{cc} \end{bmatrix}, \Delta = \begin{bmatrix} 0 & \cdots & A_{1c} \\ \vdots & \ddots & \vdots \\ A_{c1} & \cdots & 0 \end{bmatrix} \tag{214}$$

令 $\bar{A}'$ 是正则化后的邻接矩阵, FP 公式变得对于 cluster 分离

$$\begin{aligned} Z^{(L)} &= \bar{A}'\sigma\left(\bar{A}'\sigma\left(\cdots\sigma\left(\bar{A}'XW^{(0)}\right)W^{(1)}\right)\cdots\right)W^{(L-1)} \\ &= \begin{bmatrix} \bar{A}'_{11}\sigma\left(\bar{A}'_{11}\sigma\left(\cdots\sigma\left(\bar{A}'_{11}X_1W^{(0)}\right)W^{(1)}\right)\cdots\right)W^{(L-1)} \\ \vdots \\ \bar{A}'_{cc}\sigma\left(\bar{A}'_{cc}\sigma\left(\cdots\sigma\left(\bar{A}'_{cc}X_cW^{(0)}\right)W^{(1)}\right)\cdots\right)W^{(L-1)} \end{bmatrix} \end{aligned} \tag{215}$$

loss 同理

$$\mathcal{L}_{\bar{A}'} = \sum_t \frac{|V_t|}{N}\mathcal{L}_{\bar{A}'_{tt}}, \mathcal{L}_{\bar{A}'_{tt}} = \frac{1}{|\mathcal{V}_t|}\sum_{i\in\mathcal{V}_t}\operatorname{loss}\left(y_i, z_i^{(L)}\right) \tag{216}$$

使用图节点聚类方法来产生分割 (Metis or Graclus), 本作中使用的是 METIS 算法[3]

[3]

## 18.2 Stochastic Multiple Partitions

以上分割算法的问题：固定地排除了一些边；并且倾向于把相似的结点放在一起，可能引入 bias. 解决方案：先分割出相对大量的聚类，再随机选取一些聚类并在一起作为 batch. 加快收敛.

**Algorithm 1:** Cluster GCN

**Input**: Graph $A$, feature $X$, label $Y$;
**Output**: Node representation $\bar{X}$

Partition graph nodes into $c$ clusters $\mathcal{V}_1, \mathcal{V}_2, \cdots, \mathcal{V}_c$ by METIS;
**for** $iter = 1, \cdots, max\_iter$ **do**
  Randomly choose $q$ clusters, $t_1, \cdots, t_q$ from $\mathcal{V}$ without replacement;
  Form the subgraph $\bar{G}$ with nodes $\bar{\mathcal{V}} = [\mathcal{V}_{t_1}, \mathcal{V}_{t_2}, \cdots, \mathcal{V}_{t_q}]$ and links $A_{\bar{\mathcal{V}}, \bar{\mathcal{V}}}$ ;
  Compute $g \leftarrow \nabla \mathcal{L}_{A_{\bar{\mathcal{V}}, \bar{\mathcal{V}}}}$ (loss on the subgraph $A_{\bar{\mathcal{V}}, \bar{\mathcal{V}}}$) ;
  Conduct Adam update using gradient estimator $g$
Output: $\{W_l\}_{l=1}^{L}$

## 18.3 Analysis of Deeper Networks

一个方法是增加 residual-links

$$X^{(l+1)} = \sigma\left(A' X^{(l)} W^{(l)}\right) + X^{(l)} \tag{217}$$

另一个想法是

$$X^{(l+1)} = \sigma\left(\left(A' + I\right) X^{(l)} W^{(l)}\right) \tag{218}$$

(from **Wikipedia**)**METIS** is a software package for graph partitioning that implements various multilevel algorithms. METIS' multilevel approach has three phases and comes with several algorithms for each phase:

1. Coarsen the graph by generating a sequence of graphs G0, G1, ..., GN, where G0 is the original graph and for each $0 \leq i \leq j \leq N$, the number of vertices in Gi is greater than the number of vertices in Gj.
2. Compute a partition of GN
3. Project the partition back through the sequence in the order of GN, ..., G0, refining it with respect to each graph.

The final partition computed during the third phase (the refined partition projected onto G0) is a partition of the original graph.

用于强调上一层的 embedding, 为了提供数值稳定性, 使用度正则化 (区分于 GCN 的对称正则化)

$$\tilde{A}=(D+I)^{-1}(A+I) \tag{219}$$

以及 FP 公式

$$X^{(l+1)}=\sigma\left((\tilde{A}+\lambda \operatorname{diag}(\tilde{A})) X^{(l)} W^{(l)}\right) \tag{220}$$

实验证明这提高了深层网络的性能.

# 19 GAT: Graph Attention Network

GAT 层:

1. feat. trans. $\boldsymbol{h}^{\prime}=\boldsymbol{W} \boldsymbol{h}$
2. atten. coeff. $e_{ij}=a(\boldsymbol{h}_i^{\prime}, \boldsymbol{h}_j^{\prime})$
3. atten. on neighbors $\boldsymbol{\alpha}_i=\operatorname{softmax}(\boldsymbol{e}_i), \alpha_{ij}=\frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}$, 本文中使用单层 MLP+concat feat. 作为注意力层, 则有

$$\alpha_{ij}=\frac{\exp\left(\text{ LeakyReLU }\left(\overrightarrow{\mathbf{a}}^T\left[\mathbf{W} \vec{h}_i \| \mathbf{W} \vec{h}_j\right]\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\text{ Leaky ReLU}\left(\overrightarrow{\mathbf{a}}^T\left[\mathbf{W} \vec{h}_i \| \mathbf{W} \vec{h}_k\right]\right)\right)} \tag{221}$$

4. 进一步, 使用 multi-head atten.

$$\vec{h}_i^{\prime}=\|_{k=1}^K \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right) \tag{222}$$

最终层则使用 mean-aggr 而非 concat

$$\vec{h}_i^{\prime}=\sigma\left(\frac{1}{K} \sum_{k=1}^K \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right) \tag{223}$$

5. trick: transductive task 上使用随机采样的固定大小的邻域结点

# 20 Note on Probalistic Graphical Models

## 20.1 Bayesian Networks

**Theorem** (d-分离的完备性) 几乎所有 (在测度意义上) 的能被 BN 表征的概率分布 P(CSDs) 都满足: 若两节点 d-分离, 则它们条件独立.
**Theorem** (I-等价判定) 若两个 BN 有相同的骨架 (无向图基底) 和相同的 v-结构 ($X \rightarrow Z \leftarrow Y$)
朴素贝叶斯, 贝叶斯网络 (一个 DAG)

## 20.2 Undirected Networks

**Definition** 一个 (或一些)r.v. $D$ 的因子是一个函数 $\phi : \mathrm{dom}(D) \to \mathbb{R}$. 并且定义因子的乘积, $\phi_1 : \mathrm{dom}((X_i) \cup (Y_j)) \to \mathbb{R}, \phi_2 : \mathrm{dom}((Y_j) \cup (Z_k)) \to \mathbb{R}$, $\psi(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}) = \phi_2(\boldsymbol{X}, \boldsymbol{Y}) \times \phi_2(\boldsymbol{Y}, \boldsymbol{Z})$.
**Definition** 一个被

$$\Phi = \{\phi_1(\boldsymbol{D}_1), \ldots, \phi_K(\boldsymbol{D}_K)\}$$

参数化的 Gibbs 分布 $P_\Phi$ 满足

$$P_\Phi(X_1, \ldots, X_n) = \frac{1}{Z} \tilde{P}_\Phi(X_1, \ldots, X_n) \tag{224}$$

且

$$\tilde{P}_\Phi(X_1, \ldots, X_n) = \phi_1(\boldsymbol{D}_1) \times \phi_2(\boldsymbol{D}_2) \times \cdots \times \phi_m(\boldsymbol{D}_m) \tag{225}$$

其中配分函数

$$Z = \sum_{X_1, \ldots, X_n} \tilde{P}_\Phi(X_1, \ldots, X_n) \tag{226}$$

**Definition** MN 的约化 (reduction) $\mathcal{H}[\boldsymbol{u}]$ 和 $P_\Phi[\boldsymbol{u}]$ 定义为在变量集合 $\boldsymbol{U}$ 上取值后的分布/图. 且他们是一一对应的.
**Definition** $\boldsymbol{X}, \boldsymbol{Y}$ 关于 $\boldsymbol{Z}$ 分离, 若前二者之间没有不通过 $\boldsymbol{Z}$ 的路径.
**Theorem** $P$ 是 Gibbs 分布, factorize MN$\mathcal{H}$, 则后者是前者的 I-map(即独立关系包含前者).
**Theorem** (Hammersley-Clifford)$\mathcal{H}$ 是 MN, 是前者结点上的正分布 $P$ 的 I-map, 则 $P$ 是 Gibbs 分布, 且 factorize MN$\mathcal{H}$.
**Theorem** 若 $\boldsymbol{X}, \boldsymbol{Y}$ 关于 $\boldsymbol{Z}$ 不分离, 那么 $\boldsymbol{X}, \boldsymbol{Y}$ 关于 $\boldsymbol{Z}$ 不独立.
类似的, 我们可以说在几乎所有分布上独立可以推出在图上分离.
**Definition**

$$\mathcal{I}_p(\mathcal{H}) = \{(X \perp Y \mid \mathcal{X} - \{X, Y\}) : X - Y \notin \mathcal{H}\} \tag{227}$$

是 pairwise-sepration of $\mathcal{H}$,

$$\mathcal{I}_\ell(\mathcal{H}) = \{(X \perp \mathcal{X} - \{X\} - \mathrm{MB}_\mathcal{H}(X) \mid \mathrm{MB}_\mathcal{H}(X)) : X \in \mathcal{X}\} \tag{228}$$

是 markov-blanket of $\mathcal{H}$.
**Theorem** 以下结论等价 1. $P \models \mathcal{I}_\ell(\mathcal{H})$. 2. $P \models \mathcal{I}_p(\mathcal{H})$. 3. $P \models \mathcal{I}(\mathcal{H})$.
**Definition** $\phi(\boldsymbol{D}) = \exp(-\epsilon(\boldsymbol{D}))$, $\epsilon(\boldsymbol{D})$ 是能量函数.

**Definition 20.1** *一个分布 $P$ 是一个 log-linear 模型, 在 $\mathcal{H}$ 上, 若它和以下参数关联:*
*1. a set of features $\mathcal{F} = \{f_1(\boldsymbol{D}_1), \ldots, f_k(\boldsymbol{D}_k)\}$, where each $\boldsymbol{D}_i$ is a complete subgraph in $\mathcal{H}$,*
*2. a set of weights $w_1, \ldots, w_k$ such that*

$$P(X_1, \ldots, X_n) = \frac{1}{Z} \exp\left[-\sum_{i=1}^k w_i f_i(\boldsymbol{D}_i)\right]$$

**Example**
- Ising Model: 二元 r.v. $X_i \in \{-1, +1\}$, $\epsilon_{i,j}(x_i, x_j) = w_{i,j} x_i x_j$, 有能量函数

$$P(\xi) = \frac{1}{Z} \exp\left(-\sum_{i<j} w_{i,j} x_i x_j - \sum_i u_i x_i\right) \tag{229}$$

- Boltzmann Dist.: 二元 r.v. $X_i \in \{0,1\}$, 边上的能量如同 Ising 模型, 但每个随机变量都分配了 pdf $\operatorname{sigmoid}(z), z=-\left(\sum_j w_{i,j} x_j\right)-w_i$
- Metric CRF: 使用 CRF 来标注图节点, 有能量函数
$$E\left(x_1, \ldots, x_n\right)=\sum_i \epsilon_i\left(x_i\right)+\sum_{(i,j) \in \mathcal{E}} \epsilon_{i,j}\left(x_i, x_j\right) \tag{230}$$
其中能量函数的取法导致了不同的模型, Ising Model:
$$\epsilon_{i,j}\left(x_i, x_j\right)=\begin{cases}0 & x_i=x_j \\ \lambda_{i,j} & x_i \neq x_j\end{cases}=\delta_{x_i, x_j} \lambda_{i,j} \tag{231}$$
Potts Model 定义了结点的度规函数 $\mu$(需要满足非负性, 自反性, 三角不等式) 用于能量函数, 并适用于多种标签的情况. 若一个度规满足前二者 (非负性, 自反性), 则称其为 semi-metric/半度规的. CV 中常用的能量函数截断范数
$$\epsilon\left(x_i, x_j\right)=\min \left(c\left\|x_i-x_j\right\|_p, \operatorname{dist}_{\max }\right) \tag{232}$$

**Definition 20.2** 令 $\ell(\xi)=\log P(\xi)$ *Canonical energy on clique,* 关于一个特定的赋值 $\xi^*=\left(x_1^*, \ldots, x_n^*\right)$
$$\epsilon_{\boldsymbol{D}}^*(\boldsymbol{d})=\sum_{Z \subset D}(-1)^{|D-Z|} \ell\left(\boldsymbol{d}_{\boldsymbol{Z}}, \xi_{-\boldsymbol{Z}}^*\right) \tag{233}$$

**Proposition 20.3** *Let $\mathcal{B}$ be a Bayesian network over $\mathcal{X}$ and $\boldsymbol{E}=\boldsymbol{e}$ an obseruation. Let $\boldsymbol{W}=\mathcal{X}-\boldsymbol{E}$. Then $P_{\mathcal{B}}(\boldsymbol{W} \mid \boldsymbol{e})$ is a Gibbs distribution defined by the factors $\Phi=\left\{\phi_{X_i}\right\}_{\mathcal{X}_i \in \mathcal{X}}$, where*
$$\phi_{X_i}=P_{\mathcal{B}}\left(X_i \mid \mathrm{Pa}_{X_i}\right)[\boldsymbol{E}=\boldsymbol{e}]$$
*The partition function for this Gibbs distribution is $P(e)$*

**Definition 20.4** *Moralized map for BN$\mathcal{G}$ 定义为一个同样节点的无向图 $M[\mathcal{G}]$, 其中一条边 $(X,Y)$ 存在若在 $\mathcal{G}$ 中有一条有向边连接, 或者他们是 moral 的 (具有相同的子结点).*

**Proposition 20.5** *For BN $\mathcal{G}$, $M[\mathcal{G}]$ 是极小 I-map.*

**Proposition 20.6** *Let $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ be three disjoint sets of nodes in a Bayesian network $\mathcal{G}$. Let $\boldsymbol{U}=\boldsymbol{X} \cup \boldsymbol{Y} \cup \boldsymbol{Z}$, and let $\mathcal{G}^{\prime}=\mathcal{G}^{+}[\boldsymbol{U}]$ be the induced Bayesian network over $\boldsymbol{U} \cup$ Ancestors $_U$. Let $\mathcal{H}$ be the moralized* graph $\mathcal{M}\left[\mathcal{G}^{\prime}\right]$. *Then* $\mathrm{d}-\operatorname{sep}_{\mathcal{G}}(\boldsymbol{X} ; \boldsymbol{Y} \mid \boldsymbol{Z})$ *if and only if* $\operatorname{sep}_{\mathcal{H}}(\boldsymbol{X} ; \boldsymbol{Y} \mid \boldsymbol{Z})$

**Theorem 20.7** *若 MN $\mathcal{H}$ 是弦图, 则存在 BN $\mathcal{G}$ such that $\mathcal{I}(\mathcal{H})=\mathcal{I}(\mathcal{G})$*

**Definition 20.8** *CRF 是一个无向图 $\mathcal{H}$, 节点为 $\boldsymbol{X} \cup \boldsymbol{Y}$, 带有因子 $\phi_1\left(\boldsymbol{D}_1\right), \ldots, \phi_m\left(\boldsymbol{D}_m\right)$ such that each $\boldsymbol{D}_i \nsubseteq \boldsymbol{X}$, models dist. such as*
$$\begin{aligned}
P(\boldsymbol{Y} \mid \boldsymbol{X}) & =\frac{1}{Z(\boldsymbol{X})} \tilde{P}(\boldsymbol{Y}, \boldsymbol{X}) \\
\tilde{P}(\boldsymbol{Y}, \boldsymbol{X}) & =\prod_{i=1}^m \phi_i\left(\boldsymbol{D}_i\right) \\
Z(\boldsymbol{X}) & =\sum_Y \tilde{P}(\boldsymbol{Y}, \boldsymbol{X})
\end{aligned} \tag{234}$$

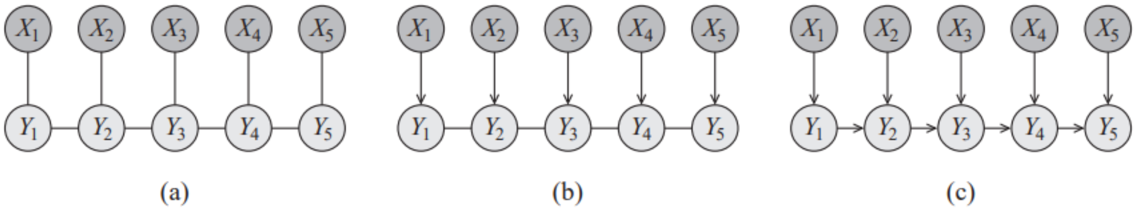



**Figure 4.14 Different linear-chain graphical models:** (a) a linear-chain-structured conditional random field, where the feature variables are denoted using grayed-out ovals; (b) a partially directed variant; (c) a fully directed, non-equivalent model. The $X_i$'s are assumed to be always observed when the network is used, and hence they are shown as darker gray.

**Example** 考虑只有一个 $Y$ 的 CRF(朴素 Markov 模型), 能量函数

$$\phi_i\left(X_i, Y\right)=\exp \left\{w_i \boldsymbol{I}\left\{X_i=1, Y=1\right\}\right\} \tag{235}$$

我们可以得到

$$P\left(Y=1 \mid x_1, \ldots, x_k\right)=\operatorname{sigmoid}\left(w_0+\sum_{i=1}^k w_i x_i\right) \tag{236}$$

a sigmoid-regression model!

## 20.3 Local Probablistic Models | i.e. Specific Models Corresponds to Last 2 Sections

表式 $\Rightarrow$ 复杂度极高!
确定性 CPD 由父节点的函数决定

$$P\left(x \mid \mathrm{pa}_X\right)=\begin{cases}1 & x=f\left(\mathrm{pa}_X\right) \\ 0 & \text{otherwise}\end{cases} \tag{237}$$

树形 CPD: 类似于决策树, 但每个节点都 annotate 一个子结点上的分布
基于规则的 CPD
noisy-or CPD

$$\begin{aligned} P\left(y^0 \mid X_1, \ldots, X_k\right) &= \left(1-\lambda_0\right) \prod_{i: X_i=x_i^1}\left(1-\lambda_i\right) \\ P\left(y^1 \mid X_1, \ldots, X_k\right) &= 1-\left[\left(1-\lambda_0\right) \prod_{i: X_i=x_i^1}\left(1-\lambda_i\right)\right] \end{aligned} \tag{238}$$

sigmoid CPD

$$P\left(y^1 \mid X_1, \ldots, X_k\right)=\operatorname{sigmoid}\left(w_0+\sum_{i=1}^k w_i X_i\right) \tag{239}$$

Gaussian CPD

$$p\left(Y \mid x_1, \ldots, x_k\right)=\mathcal{N}\left(\beta_0+\beta_1 x_1+\cdots+\beta_k x_k ; \sigma^2\right) \tag{240}$$

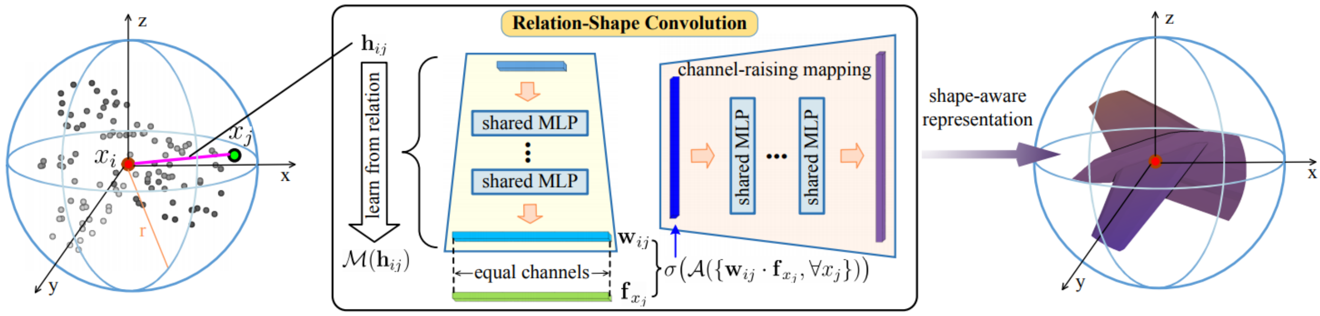



Figure 2. Overview of relation-shape convolution (RS-Conv). The key is to learn from relation. Specifically, the convolutional weight for $x_j$ is converted to $\mathbf{w}_{ij}$, which learns a mapping $\mathcal{M}$ (Eq. (2)) on predefined geometric relation vector $\mathbf{h}_{ij}$. In this way, the inductive convolutional representation $\sigma\big(\mathcal{A}(\{\mathbf{w}_{ij} \cdot \mathbf{f}_{x_j}, \forall x_j\})\big)$ (Eq. (3)) can expressively reason the spatial layout of points, resulting in discriminative shape awareness. As in image CNN [34], further channel-raising mapping is conducted for a more powerful shape-aware representation.

写成向量，则为

$$p(Y \mid \boldsymbol{x}) = \mathcal{N}\left(\beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}; \sigma^2\right) \tag{241}$$

条件线性高斯模型 (CLG)

$$p(X \mid \boldsymbol{u}, \boldsymbol{y}) = \mathcal{N}\left(a_{\boldsymbol{u},0} + \sum_{i=1}^{k} a_{\boldsymbol{u},i} y_i; \sigma_{\boldsymbol{u}}^2\right) \tag{242}$$

**Definition 20.9** *(conditional Bayesian networks)* 条件贝叶斯网络 $\mathcal{G}$ 是一个 *DAG*，节点是分离的三个集合的并 $\boldsymbol{X} \cup \boldsymbol{Y} \cup \boldsymbol{Z}$，$\boldsymbol{X}$ 没有父节点，称作输入，$\boldsymbol{Y}$ 称作输出，且条件概率分布由链式法则定义

$$P_{\mathcal{B}}(\boldsymbol{Y}, \boldsymbol{Z} \mid \boldsymbol{X}) = \prod_{X \in Y \cup Z} P\left(X \mid \mathrm{Pa}_X^{\mathcal{G}}\right) \tag{243}$$

. 边缘分布由求和给出

$$P_{\mathcal{B}}(\boldsymbol{Y} \mid \boldsymbol{X}) = \sum_{\boldsymbol{Z}} P_{\mathcal{B}}(\boldsymbol{Y}, \boldsymbol{Z} \mid \boldsymbol{X}) \tag{244}$$

### 20.4 Temporal Models

# 21 RSCNN(CVPR 19')

## 21.1 Architecture

**Idea** 使用空间卷积/spatial conv.，在球形邻域上.
一个广义卷积

$$\mathbf{f}_{P_{\text{sub}}} = \sigma\left(\mathcal{A}\left(\left\{\mathcal{T}\left(\mathbf{f}_{x_j}\right), \forall x_j\right\}\right)\right), d_{ij} < r \forall x_j \in \mathcal{N}\left(x_i\right) \tag{245}$$

要想是这个卷积 permut.-invar.，函数 $\mathcal{A}, \mathcal{T}$ 必须分别是对称的和 shared.

使用 shape-awared/geometric info 函数 $\mathcal{M}$(shared MLP 建模) 代替传统卷积

$$\mathcal{T}\left(\mathbf{f}_{x_j}\right)=\mathbf{w}_{ij}\cdot\mathbf{f}_{x_j}=\mathcal{M}\left(\mathbf{h}_{ij}\right)\cdot\mathbf{f}_{x_j} \tag{246}$$

则卷积形式变为

$$\mathbf{f}_{P_{\text{sub}}}=\sigma\left(\mathcal{A}\left(\left\{\mathcal{M}\left(\mathbf{h}_{ij}\right)\cdot\mathbf{f}_{x_j},\forall x_j\right\}\right)\right) \tag{247}$$

为了和 CNN 相对应, 使用 channel-raising MLP 来增多 channels.
最终, 这个卷积具有以下性质: permut. invar., 对于刚性变换的健壮性, shared weights, interacted point geometric.

### 21.2 Details & Implementation

使用 ReLU 激活函数, 使用 BN, $\mathcal{M}$ 使用三层 MLP, aggr. f. 为 max-pooling. Low-level 几何表示 $\boldsymbol{h}_{ij}=[\boldsymbol{x}_i-\boldsymbol{x}_j,\boldsymbol{x}_i,\boldsymbol{x}_j]$, channel-raising 使用单层 MLP.
点云采样方面, 使用原点云的 FPS 采样. 使用 3-scale 邻域 (不同于 PN++ 的 MSG)

## 22 SimpleView(ICLR 21' Candidate)

### 22.1 Simple Review of Existing Protocols

**数据增强**: 包括抖动, y-轴随机旋转, 随即平移和缩放. ModelNet40 由于已经对齐, y-轴随机旋转会降低性能.

| Model | PointNet(++) | DGCNN | RSCNN |
|---|---|---|---|
| | All | 随机旋转/平移 | 随机旋转/平移 |

**输入点数** PN(++) 使用 1024 个固定输入. PointCNN, RSCNN 使用每个 epoch 重采样的点.
**Loss** 大多数方法使用交叉熵, DGCNN 使用了平滑了的交叉熵 (label 经过平滑, 这个方法在所有结构上提高了性能)
**模型选择** PN(++) 使用最终收敛的模型, DGCNN/RSCNN 使用测试集上的最好模型.
**模型聚合** PN(++) 在 inference-time 把最终模型在不同旋转角度的输入上做判定 (10 次), 然后投票决定. RSCNN, DensePoint 在不同尺寸和角度的输入上判定 (300 次), 然后投票决定. DGCNN 完全没有投票.
比较性能, 本文提出的方法使用随机平移/缩放强化和 smooth-loss, 并且为了不利用任何测试集的信息, 使用 final model sel.

### 22.2 Model: SimpleView

**Idea** 使用多个视角的深度图像!
具体上, 使用六个 view(水平面四个, z 轴两个, 实验上这样性能最好), 并且在每张深度图上使用 ResNet18/4 骨架 (ResNet18, 滤波器数量为 1/4), concat 连接特征.

# 23 OT-Flow

## 23.1 Idea & Formulations

Formulation(based on FFJORD)

$$\partial_t \begin{bmatrix} \boldsymbol{z}(\boldsymbol{x},t) \\ \ell(\boldsymbol{x},t) \end{bmatrix} = \begin{bmatrix} \mathbf{v}(\boldsymbol{z}(\boldsymbol{x},t),t;\boldsymbol{\theta}) \\ \operatorname{tr}(\nabla \mathbf{v}(\boldsymbol{z}(\boldsymbol{x},t),t;\boldsymbol{\theta})) \end{bmatrix}, \quad \begin{bmatrix} \boldsymbol{z}(\boldsymbol{x},0) \\ \ell(\boldsymbol{x},0) \end{bmatrix} = \begin{bmatrix} \boldsymbol{x} \\ 0 \end{bmatrix} \tag{248}$$

在 FFJORD 的基础

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\rho_0(\boldsymbol{x})} \left\{ C(\boldsymbol{x},T) := \frac{1}{2}\|\boldsymbol{z}(\boldsymbol{x},T)\|^2 - \ell(\boldsymbol{x},T) + \frac{d}{2}\log(2\pi) \right\} \tag{249}$$

上增加最优输运代价

$$L(\boldsymbol{x},T) = \int_0^T \frac{1}{2}\|\mathbf{v}(\boldsymbol{z}(\boldsymbol{x},t),t)\|^2 \mathrm{d}t \tag{250}$$

满足上两个 cost 的和最小化时, 则必定存在势函数

$$\mathbf{v}(\boldsymbol{x},t;\boldsymbol{\theta}) = -\nabla\Phi(\boldsymbol{x},t;\boldsymbol{\theta}) \tag{251}$$

并且满足 HJB 方程 (Hamilton-Jacobi-Bellman Eq.)

$$-\partial_t\Phi(\boldsymbol{x},t) + \frac{1}{2}\|\nabla\Phi(\boldsymbol{z}(\boldsymbol{x},t),t)\|^2 = 0, \quad \Phi(\boldsymbol{x},T) = G(\boldsymbol{x}) \tag{252}$$

故引入惩罚项

$$R(\boldsymbol{x},T) = \int_0^T \left| \partial_t\Phi(\boldsymbol{z}(\boldsymbol{x},t),t) - \frac{1}{2}\|\nabla\Phi(\boldsymbol{z}(\boldsymbol{x},t),t)\|^2 \right| \mathrm{d}t \tag{253}$$

本工作直接不建模梯度函数 $\boldsymbol{v}$, 而是直接建模势函数 $\Phi$.

## 23.2 Parametrization of Model

势函数

$$\Phi(\boldsymbol{s};\boldsymbol{\theta}) = \boldsymbol{w}^\top N(\boldsymbol{s};\boldsymbol{\theta}_N) + \frac{1}{2}\boldsymbol{s}^\top\left(\boldsymbol{A}^\top\boldsymbol{A}\right)\boldsymbol{s} + \boldsymbol{b}^\top\boldsymbol{s} + c, \quad \text{where } \boldsymbol{\theta} = (\boldsymbol{w},\boldsymbol{\theta}_N,\boldsymbol{A},\boldsymbol{b},c) \tag{254}$$

其中 $N$ 是一个 NN(这里用的是一个简单的两层 ResNet), $\boldsymbol{A} \in \mathbb{R}^{r\times(d+1)}$, 且限制 rank
$r = \max(10,d)$ 这里后面三项建模了一个二次势函数, 也即一个线性动力系统, NN 则建模了非线性部分.
ResNet 结构

$$\begin{aligned} \boldsymbol{u}_0 &= \sigma(\boldsymbol{K}_0\boldsymbol{s} + \boldsymbol{b}_0) \\ N(\boldsymbol{s};\boldsymbol{\theta}_N) = \boldsymbol{u}_1 &= \boldsymbol{u}_0 + h\sigma(\boldsymbol{K}_1\boldsymbol{u}_0 + \boldsymbol{b}_1) \end{aligned} \tag{255}$$

梯度计算

$$\nabla_s\Phi(s;\boldsymbol{\theta}) = \nabla_s N(\boldsymbol{s};\boldsymbol{\theta}_N)\boldsymbol{w} + \left(\boldsymbol{A}^\top\boldsymbol{A}\right)\boldsymbol{s} + \boldsymbol{b} \tag{256}$$

Hessian Trace 计算

$$\operatorname{tr}\left(\nabla^2\Phi(\boldsymbol{s};\boldsymbol{\theta})\right) = \operatorname{tr}\left(\boldsymbol{E}^\top\nabla^2_{\boldsymbol{s}}(N(\boldsymbol{s};\boldsymbol{\theta}_N)\boldsymbol{w})\boldsymbol{E}\right) + \operatorname{tr}\left(\boldsymbol{E}^\top\left(\boldsymbol{A}^\top\boldsymbol{A}\right)\boldsymbol{E}\right) \tag{257}$$

后一项是 trivial 的, $\boldsymbol{E}$ 是 $\mathbb{R}^(d+1)$ 标准正交基的前 d 项,ResNet 项可以得到一个闭形式

$$\begin{aligned}\operatorname{tr}\left(\boldsymbol{E}^{\top}\nabla_{\boldsymbol{s}}^{2}\left(N\left(\boldsymbol{s};\boldsymbol{\theta}_{N}\right)\boldsymbol{w}\right)\boldsymbol{E}\right)=t_{0}+ht_{1},\quad & \text{where}\\ & t_{0}=\left(\sigma''\left(\boldsymbol{K}_{0}\boldsymbol{s}+\boldsymbol{b}_{0}\right)\odot\boldsymbol{z}_{1}\right)^{\top}\left(\left(\boldsymbol{K}_{0}\boldsymbol{E}\right)\odot\left(\boldsymbol{K}_{0}\boldsymbol{E}\right)\right)\mathbf{1}\\ & t_{1}=\left(\sigma''\left(\boldsymbol{K}_{1}\boldsymbol{u}_{0}+\boldsymbol{b}_{1}\right)\odot\boldsymbol{w}\right)^{\top}\left(\left(\boldsymbol{K}_{1}\nabla_{\boldsymbol{s}}\boldsymbol{u}_{0}^{\top}\right)\odot\left(\boldsymbol{K}_{1}\nabla_{\boldsymbol{s}}\boldsymbol{u}_{0}^{\top}\right)\right)\mathbf{1}\end{aligned} \tag{258}$$

第一层的 Hessian 计算复杂度 $O(md)$, 之后每多一层为 $O(m^2d)$, 总复杂度则为 $O(d)$

## 23.3 Exact Hessian of Multilayer NN

Exact Trace Computation Using (13) and the same $E$, we compute the trace in one forward pass through the layers. The trace of the first ResNet layer is

$$\begin{aligned}t_{0}&=\operatorname{tr}\left(\boldsymbol{E}^{\top}\nabla_{\boldsymbol{s}}\left(\boldsymbol{K}_{0}^{\top}\operatorname{diag}\left(\sigma''\left(\boldsymbol{K}_{0}\boldsymbol{s}+\boldsymbol{b}_{0}\right)\right)\boldsymbol{z}_{1}\right)\boldsymbol{E}\right)\\&=\operatorname{tr}\left(\boldsymbol{E}^{\top}\boldsymbol{K}_{0}^{\top}\operatorname{diag}\left(\sigma''\left(\boldsymbol{K}_{0}\boldsymbol{s}+\boldsymbol{b}_{0}\right)\odot\boldsymbol{z}_{1}\right)\boldsymbol{K}_{0}\boldsymbol{E}\right)\\&=\left(\sigma''\left(\boldsymbol{K}_{0}\boldsymbol{s}+\boldsymbol{b}_{0}\right)\odot\boldsymbol{z}_{1}\right)^{\top}\left(\left(\boldsymbol{K}_{0}\boldsymbol{E}\right)\odot\left(\boldsymbol{K}_{0}\boldsymbol{E}\right)\right)\mathbf{1}\end{aligned}$$

using the same notation as (14). For the last step, we used the diagonality of the middle matrix. Computing $t_0$ requires $\mathcal{O}(m\cdot d)$ FLOPS when first squaring the elements in the first $d$ columns of $\boldsymbol{K}_0$, then summing those columns, and finally one inner product.
To compute the trace of the entire ResNet, we continue with the remaining rows in (27) in reverse order to obtain

$$\operatorname{tr}\left(\boldsymbol{E}^{\top}\nabla_{\boldsymbol{s}}^{2}\left(N\left(\boldsymbol{s};\boldsymbol{\theta}_{N}\right)\boldsymbol{w}\right)\boldsymbol{E}\right)=t_{0}+h\sum_{i=1}^{M}t_{i}$$

where $t_i$ is computed as

$$\begin{aligned}t_{i}&=\operatorname{tr}\left(J_{i-1}^{\top}\nabla_{\boldsymbol{s}}\left(\boldsymbol{K}_{i}^{\top}\operatorname{diag}\left(\sigma''\left(\boldsymbol{K}_{i}\boldsymbol{u}_{i-1}(\boldsymbol{s})+\boldsymbol{b}_{i}\right)\right)\boldsymbol{z}_{i+1}\right)J_{i-1}\right)\\&=\operatorname{tr}\left(J_{i-1}^{\top}\boldsymbol{K}_{i}^{\top}\operatorname{diag}\left(\sigma''\left(\boldsymbol{K}_{i}\boldsymbol{u}_{i-1}+\boldsymbol{b}_{i}\right)\odot\boldsymbol{z}_{i+1}\right)\boldsymbol{K}_{i}J_{i-1}\right)\\&=\left(\sigma''\left(\boldsymbol{K}_{i}\boldsymbol{u}_{i-1}\mid\boldsymbol{b}_{i}\right)\odot\boldsymbol{z}_{i+1}\right)^{\top}\left(\left(\boldsymbol{K}_{i}J_{i-1}\right)\odot\left(\boldsymbol{K}_{i}J_{i-1}\right)\right)\mathbf{1}\end{aligned}$$

Here, $J_{i-1}=\nabla_{s}\boldsymbol{u}_{i-1}^{\top}\in\mathbb{R}^{m\times d}$ is a Jacobian matrix, which can be updated and over-written in the forward pass at a computational cost of $\mathcal{O}\left(m^{2}\cdot d\right)$ FLOPS. The $J$ update follows:

$$\begin{aligned}\nabla_{\boldsymbol{s}}\boldsymbol{u}_{i}^{\top}&=\nabla_{\boldsymbol{s}}\boldsymbol{u}_{i-1}+h\sigma'\left(\boldsymbol{K}_{i}\boldsymbol{u}_{i-1}+\boldsymbol{b}_{i}\right)\boldsymbol{K}_{i}^{\top}\nabla_{\boldsymbol{s}}\boldsymbol{u}_{i-1}\\J&\leftarrow J+h\sigma'\left(\boldsymbol{K}_{i}\boldsymbol{u}_{i-1}+\boldsymbol{b}_{i}\right)\boldsymbol{K}_{i}^{\top}J\end{aligned}$$