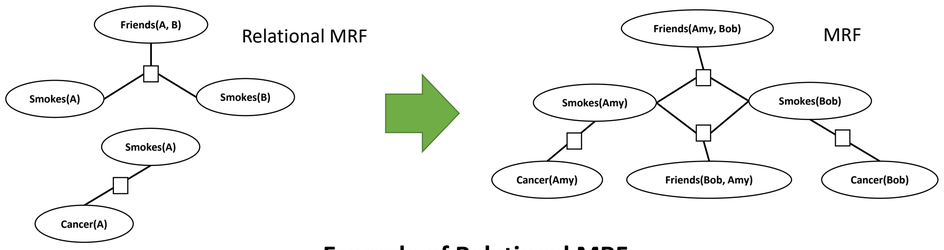
Relational Neural Markov Random Fields

Yuqiao Chen , Nicholas Ruozzi , Sriraam Natarajan

Motivation



Example of Relational MRF

Relational MRF is extremely useful for encoding human knowledge (e.g. MLN) by defining the dependency among features of objects.

Existing methods are limited:

- **Restricted potential functions, unable to model complicated dependency.**
- **Does not support learning models with continuous features.**

Learning hybrid MRF is hard:

- **Inference on MRF with continuous variables is expensive, which makes learning with MLE unreasonably slow.**

Background

Learning MRF by Maximizing Pseudo Likelihood (MPLE)

$$\log l(x^{(1:M)};\theta) \approx \frac{1}{M}\sum_m \sum_{i\in\mathcal{V}} \log p(x_i^{(m)} \mid MB_i^{(m)};\theta)$$

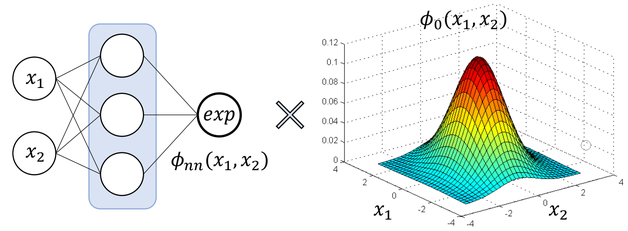
The parameter gradient is computed as

$$\nabla \log l(x^{(1:M)};\theta) = \frac{1}{M}\sum_m \sum_{i\in\mathcal{V}} \sum_{c\supset i} \Big[\nabla f_c(x_c^{(m)};\theta_c) - \mathop{\mathbb{E}}_{p(x_i|MB_i^{(m)};\theta)} \Big(\nabla f_c(x_i, x_{c\setminus i}^{(m)};\theta_c)\Big)\Big]$$

Computing the expectation term requires integration over the domain. It would be hard for continuous domain.

Relational Neural MRFs

Neural Net Potential Function

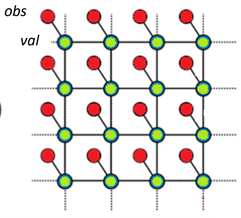


$$\phi(x_c) = \exp nn(x_c) \cdot \phi_0(x_c)$$

Helper function, which works as sampling prior

- With the express power of neural network, we can model complicated dependency.
- For unbounded continuous domain, the helper function can help sampling on region of interest.

Examples of RN-MRF



$$LG(1,0,1) : \phi_{nn1}(|val(P1) - val(P2)|) \text{ st. } nb(P1,P2)$$

$$LG(1,0,1) : \phi_{nn2}(|obs(P) - val(P)|)$$

$$U : \phi_{nn}(length(S), depth(S), type(S))$$

$$U : \phi_{mln1}(length(S)\text{>}0.5 \Rightarrow type(S)\text{!='W'})$$

$$U : \phi_{mln2}(type(S1)\text{='D'} \Rightarrow type(S2)\text{!='D'}) \text{ st. } nb(S1,S2)$$

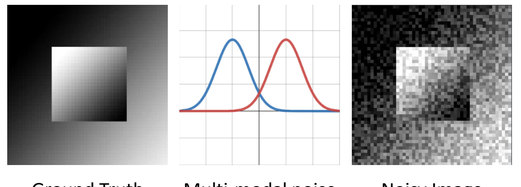
MPLE algorithm for learning RN-MRF

$$\nabla nn(x_i, x_{c\setminus i}) = \begin{cases} 1 & \text{for } x_i \text{ in data} \\ -\dfrac{\sum_{n\sim Q}\prod_{h\supset i}\phi_{nn_h}(x_i, x_{h\setminus i}^{(m)})\cdot\nabla nn_c(x_i^{(n)}, x_{c\setminus i}^{(m)})}{\sum_{n\sim Q}\prod_{h\supset i}\phi_{nn_h}(x_i, x_{h\setminus i}^{(m)})} & \text{for } x_i \text{ sampled from } Q \end{cases}$$

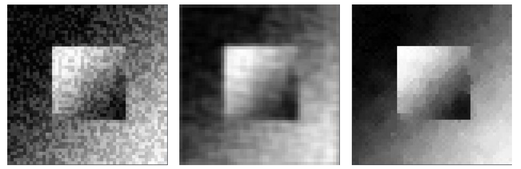
where the proposal distribution $Q(x_i) = \prod_{h\supset i}\phi_{0_h}(x_i, x_{h\setminus i}^{(m)})$

Results

Image Denoising with Multi-modal Noise



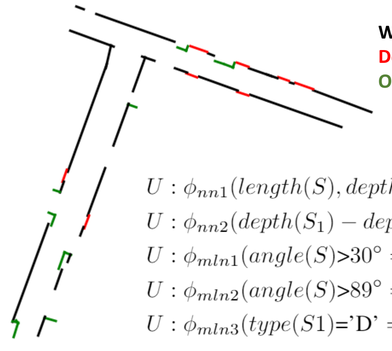
Ground Truth | Multi-modal noise | Noisy Image



Gaussian-MRF | CNN | RN-MRF

Model	ℓ_1 Error	ℓ_2 Error
Gaussian-MRF	186.14 ± 7.26	18.97 ± 1.47
CNN	120.33 ± 13.70	12.86 ± 3.33
RN-MRF(no Helper & FM)	58.55 ± 5.89	3.07 ± 0.89
RN-MRF(with only FM)	94.28 ± 17.12	7.70 ± 1.98
RN-MRF(with Helper & FM)	99.97 ± 20.09	7.81 ± 2.47

Robot Mapping with NN + Human Knowledge



$$U : \phi_{nn1}(length(S), depth(S), angle(S), type(S))$$

$$U : \phi_{nn2}(depth(S_1) - depth(S_2), type(S_1), type(S_2))$$

$$U : \phi_{mln1}(angle(S)\text{>}30° \Rightarrow type(S)\text{!='W'})$$

$$U : \phi_{mln2}(angle(S)\text{>}89° \Rightarrow type(S)\text{='O'})$$

$$U : \phi_{mln3}(type(S1)\text{='D'} \Rightarrow type(S2)\text{!='D'}) : nb(S1,S2)$$

Model	Accuracy	F1(Wall)	F1(Door)	F1(Other)
RN-MRF	0.876	0.944	0.821	0.809
RN-MRF(no MLN)	0.852	0.912	0.807	0.781
RN-CRF	0.901	0.945	0.88	0.836
RN-CRF(no MLN)	0.88	0.935	0.842	0.812
Neural-CRF	0.884	0.961	0.842	0.782
CG-MRF	0.762	0.838	0.703	0.667
Expert HMLN	0.761	0.815	0.807	0.487