Discriminative Embeddings of Latent Variable Models for Structured Data

(ICML 2016)

Hanjun Dai , Bo Dai , Le Song,
College of Computing, Georgia Institute of Technology
https://qdata.github.io/deep2Read
Presenter: Arshdeep Sekhon

Fall 2018

Motivation

- kernel methods: two step process
- feature representations of these kernels independent of discriminative task

Kernels

- Bag of Structures Kernels:
- the feature representations of these kernels are fixed before learning, with each dimension corresponding to a substructure

Kernels

- Bag of Structures Kernels:
- the feature representations of these kernels are fixed before learning, with each dimension corresponding to a substructure
- GM kernels:
- kernels based on probabilistic graphical models
- Example, Fisher kernel: fits a common generative model to the entire dataset, and then uses the empirical Fisher information matrix and the Fisher score of each data point to define the kernel
- probability product kernel: different generative model for each data point, and then uses inner products between distributions to define the kernel

Structure2vec

- learn the feature representation from label information
- scale up (not save the entire kernel matrix)

Structure2vec

- learn the feature representation from label information
- scale up (not save the entire kernel matrix)
- model each structured data point as a latent variable model
- embed the graphical model into feature spaces
- inner product in the embedding space to define kernels.
- learn the feature space by directly minimizing the empirical loss defined by the label information

Hilbert Space Embedding of Distributions

- map distributions to potentially infinite dimensional feature spaces
- map distributions to expected feature map
- possibly injective (gaussian kernel)

$$\mu_X := \mathbb{E}_X[\phi(X)] = \int_{\mathcal{X}} \phi(x)p(x)dx \; : \; \mathcal{P} \mapsto \mathcal{F}$$

$$\mu_X := \mathbb{E}_X[k(X,\cdot)] = \mathbb{E}_X[\phi(X)] = \int_{\Omega} \phi(x)\,\mathrm{d}P(x)$$

Discriminative Embed

Hilbert Space Embedding of Distributions

- treat expected feature map μ_X as a sufficient statistic
- $f(p(x)) = \tilde{f}(\mu_X)$

Hilbert Space Embedding of Distributions

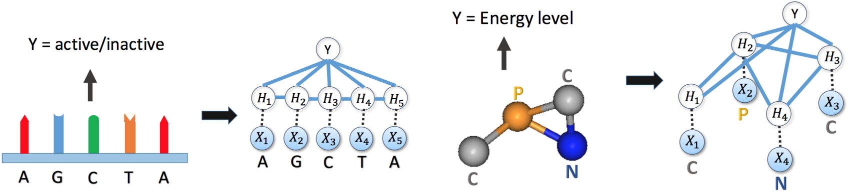
- treat expected feature map μ_X as a sufficient statistic
- $f(p(x)) = \tilde{f}(\mu_X)$
- Operator $T : P \to R^d$
- $T \odot p(x) = \tilde{T} \odot \mu_X$

Model for a structured data point

- every data point is a graph
- each node has value x_i
- each data point is an instance from a graphical model
- Each Node has X_i with a hidden variable H_i
- graphical model structure of each data point as conditional independece structure

$$p(\{H_i\},\{X_i\}) \propto \prod_{i\in\mathcal{V}} \Phi(H_i, X_i) \prod_{(i,j)\in\mathcal{E}} \Psi(H_i, H_j)$$

Pairwise Markov Random Fields



Embedding Latent Variable Models

- $p(H_i|\{x_i\})$ embed this into a feature map $\phi(H_i) \in R^d$
- very hard to compute

$$p(H_i|\{x_i\}) = \int_{\mathcal{H}^{V-1}} p(H_i, \{h_j\} \,|\, \{x_j\}) \prod_{j \in \mathcal{V} \backslash i} dh_j.$$

Belief Propagation: Introduction

- for estimating marginals
- Usually, probability defined in terms of product groups

$$p(x_1, x_2, x_3) = \frac{1}{Z} f(x_1, x_2) g(x_1, x_3) h(x_1, x_2, x_3) \qquad (1)$$

- f,g,h are potentials or functions to determine probabilities
- in some cases, conditional probabilities

Belief Propagation: Introduction

- Marginals: $p(x_1), p(x_2), p(x_3)$
- maximizer: $argmax\ p(x_1, x_2, x_3, x_n)$
- say each has S states
- $O(S^N)$ complexity: exhaustive addition or exhaustive search

Belief Propagation: Introduction

- neighbors pass messages to nodes
- estimate marginal probability for the state spaces of the nodes
- Estimated marginal probabilities: beliefs

Pairwise Markov Random Field

$$P(x_1, x_2, \ldots, x_n) = \frac{1}{Z} \prod_{i=1}^{N} g_i(x_i) \prod_{<ij>} f_i j(x_i, x_j) \quad (2)$$

- g,f are unary and pairwise factors/potentials
- BP also for factor graphs

Message Passing

- Node i sends to Node j: $m_{ij}(x_j)$
- high value of message: node i “believes” marginal value $P(x_j)$ is high
- random or uniform initialization
- For $m_{ij}(x_j)$: messages into i (except from j) also considered

Message Update

$$m_{ij}^{new}(x_j) = \Sigma_{x_i} f_{ij}(x_i, x_j) g_i(x_i) \prod_{k \in Nbd(i)--j} m_{ki}^{old}(x_i) \tag{3}$$

$$m_{ij}^{new}(x_j) = \Sigma_{x_i} f_{ij}(x_i, x_j) h(x_i) \tag{4}$$

Message Pasing

- for one pair: message in both directions
- but $f_{ij}(x_i, x_j) = f_{ji}(x_j, x_i)$
- not the same as symmetric potential
- incoming messages for a node sum to 1: $\Sigma_{x_j} m_{ij}(x_j) = 1$

Schedule

- update everything in parallel vs one msg at a time
- depends on graph structure

Read out

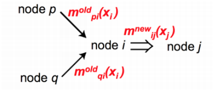
- $$b_i(x_i) \propto g_i(x_i) \prod_{k \in Nbd(i)} m_{ki}(x_i) \tag{5}$$
- exact marginal probability if normalized beleif and no loops
- can be easily formulated as max product to find best state configuration
- factor graph variation also exists

Example

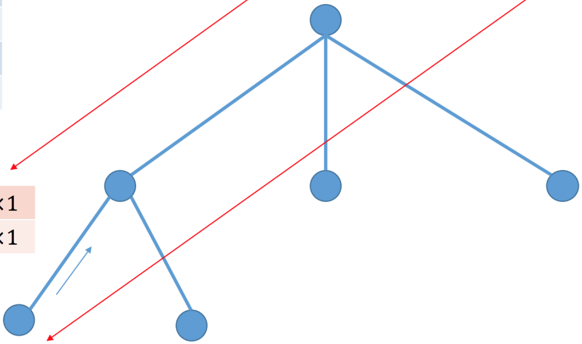
Example (Sum-Product)

A	2
B	1

A -> A	1
A -> B	2
B -> A	3
B-> B	4



$$m_{ij}^{new}(x_j) = \sum_{x_i} f_{ij}(x_i, x_j) g_i(x_i) \underbrace{\prod_{k \in Nbd(i) \backslash j} m_{ki}^{old}(x_i)}_{h(x_i)}$$

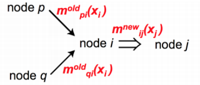


A	1×2 + 3×1
B	2×2 + 4×1

A	2
B	1

Example

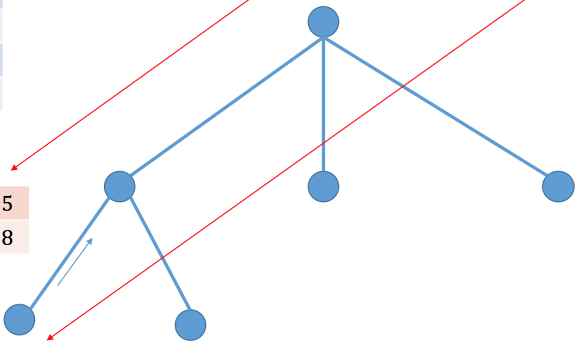
Example (Sum-Product)



$$m_{ij}^{new}(x_j) = \sum_{x_i} f_{ij}(x_i, x_j) g_i(x_i) \underbrace{\prod_{k \in Nbd(i) \setminus j} m_{ki}^{old}(x_i)}_{h(x_i)}$$

A	2
B	1

A -> A	1
A -> B	2
B -> A	3
B-> B	4



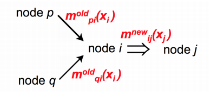
A	5
B	8

A	2
B	1

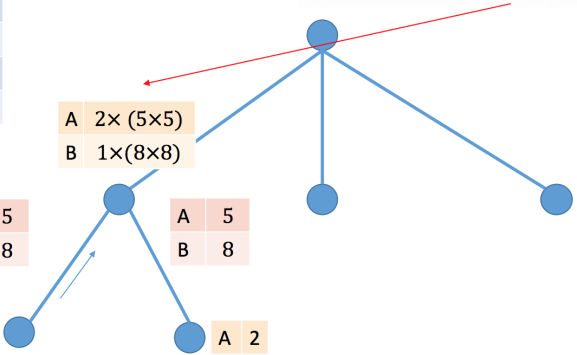
Example (Sum-Product)

A	2
B	1

A -> A	1
A -> B	2
B -> A	3
B-> B	4



$$m_{ij}^{new}(x_j) = \sum_{x_i} f_{ij}(x_i, x_j) g_i(x_i) \underbrace{\prod_{k \in Nbd(i) \setminus j} m_{ki}^{old}(x_i)}_{h(x_i)}$$



A	2× (5×5)
B	1×(8×8)

A	5
B	8

A	5
B	8

A	2
B	1

A	2
B	1

Loopy Belief Propagation: Approximation

- Run BP on loopy graph
- Message passing performs well on tree structured graphs.
- for loopy graphs , messages may circulate indefinitely around the loops : may not converge.
- Even when they converge, the stable equilibrium may not represent the posterior probabilities of the nodes.

Loopy Belief Propagation: Theory

define the true distribution (P) over a graphical model as

$$P(X) = \frac{1}{Z} \prod_{f_a \in F} f_a(X_a) \tag{6}$$

F denotes the set of all factors

P is the product of the individual factors in the the factor graph

ELBO:

$$-H_Q(X) - \Sigma_{f_a \in F} E_Q log(f_a(X_a)) \tag{7}$$

Loopy Belief Propagation: Theory

For a tree graph:

$$b(x) = \prod_a b_a(x_a) \prod_i b_i(x_i)^{1-d_i} \tag{8}$$

Entropy for tree structured:

$$H_{tree} = -\sum_a \sum_{x_a} b_a(x_a) \log b_a(x_a) + \sum_i (d_i - 1) \sum_{x_i} b_i(x_i) \log b_i(x_i)$$

$$\begin{aligned} F_{tree} &= \sum_a \sum_{x_a} b_a(x_a) \log\left(\frac{b_a(x_a)}{f_a(x_a)}\right) + \sum_i (1 - d_i) \sum_{x_i} b_i(x_i) \log b_i(x_i) \\ &= F_{12} + F_{23} + \ldots + F_{67} + F_{78} - F_1 - F_5 - F_2 - F_6 - F_3 - F_7 \end{aligned}$$

Loopy Belief Propagation: Theory

Take tree elbo as approximation for general factor graphs: called bethe free energy

$$
\begin{aligned}
F_{Bethe} &= \sum_a \sum_{x_a} b_a(x_a) \log\left(\frac{b_a(x_a)}{f_a(x_a)}\right) + \sum_i (1 - d_i) \sum_{x_i} b_i(x_i) \log b_i(x_i) \\
&= F_{12} + F_{23} + ... + F_{67} + F_{78} - F_1 - F_5 - 2F_2 - 2F_6 - ... - F_8
\end{aligned}
$$

Loopy Belief Propagation: Theory Proof

$$L = F_{Bethe} + \sum_i \gamma_i \left\{ 1 - \sum_{x_i} b_i(x_i) \right\} + \sum_a \sum_{i \in N(a)} \sum_{x_i} \lambda_{ai}(x_i) \left\{ b_i(x_i) - \sum_{X_a \backslash x_i} b_a(X_a) \right\} \tag{27}$$

Setting the derivate with respect to the paramaters to zero:

$$\frac{\partial L}{\partial b_i(x_i)} = 0 \implies \qquad b_i(x_i) \propto \exp\left(\frac{1}{d_i - 1} \sum_{a \in N(i)} \lambda_{ai}(x_i)\right) \tag{28}$$

$$\frac{\partial L}{\partial b_a(X_a)} = 0 \implies \qquad b_a(X_a) \propto \exp\left(-\log f_a(X_a) + \sum_{i \in N(a)} \lambda_{ai}(x_i)\right) \tag{29}$$

If we set $\lambda_{ai}(x_i) = \log m_{i \to a} = \log \prod_{b \in N(i) \backslash a} m_{b \to i}(x_i)$, we obtain:

$$b_i(x_i) \propto f_i(x_i) \prod_{a \in N(i)} m_{a \to i}(x_i) \tag{30}$$

$$b_a(X_a) \propto f_a(X_a) \prod_{i \in N(a)} \prod_{c \in N(i) \backslash a} m_{c \to i}(x_i) \tag{31}$$

Now, if we use the fact that $m_{a \to i}(x_i) = \sum_{X_a \backslash x_i} b_a(X_a)$, where we are excluding the message $m_{i \to a}$:

$$m_{a \to i}(x_i) = \sum_{X_a \backslash x_i} f_a(X_a) \prod_{j \in N(a) \backslash i} \prod_{b \in N(j) \backslash a} m_{b \to j}(x_j) \tag{32}$$

we do not need to optimize explicitly for q(X) over the entire space of possibilities

We can just focus on the set of doubleton and singleton beliefs to relax the optimization objective

$$b^{\star} = \arg\min_{b \in M_o} \{F_{Bethe}(p, b)\}$$

Mean Field Inference: Background

Posterior hard to compute:

$$p(z|x,\alpha) = \frac{p(z,x|\alpha)}{\int_z p(z,x|\alpha)} \tag{9}$$

KL divergence:

$$KL(q||p) = E[log\frac{q(z)}{p(z|x)}] \tag{10}$$

ELBO (Variational Free Energy):

$$E_q[logp(x,z)] - E[log(q(z))] \tag{11}$$

(12)

Mean Field Inference: Variational Distribution

- assume the variational distribution over the latent variables factorizes as
$$q(z_1, z_2, \ldots, z_m) = \prod_{j=1} q(z_j) \tag{13}$$
- Not the true posterior: the latent variables are independent

Mean Field Variational Inference

- approximate $p(\{H_i\}|\{x_i\})$ with a product of independent density components $\prod_{i \in V} q(_i(h_i))$

$$\min_{q_1,\ldots,q_d} \int_{\mathcal{H}^d} \prod_{i \in \mathcal{V}} q_i(h_i) \log \frac{\prod_{i \in \mathcal{V}} q_i(h_i)}{p(\{h_i\} \mid \{x_i\})} \prod_{i \in \mathcal{V}} dh_i.$$

$$\begin{aligned}
\log q_i(h_i) =& c_i + \log(\Phi(h_i, x_i)) + \sum_{j \in \mathcal{N}(i)} \int_{\mathcal{H}} q_j(h_j) \log(\Psi(h_i, h_j)\Phi(h_j, x_j)) dh_j \\
=& c_i' + \log \Phi(h_i, x_i) + \sum_{j \in \mathcal{N}(i)} \int_{\mathcal{H}} q_j(h_j) \log \Psi(h_i, h_j) dh_j
\end{aligned}$$

Embedding Mean Field Variational Inference

- $q_i(h_i)$ is a functional of set of neighboring marginals $\{q_j\}_{j \in N_i}$
- $q_i(h_i) = f(h_i, x_i, \{q_j\}_{j \in N(i)})$
- for each marginal q_i, we have an injective embedding

$$\tilde{\mu}_i = \int \phi(h_i) q_i(h_i) dh_i \tag{14}$$

- $q_i(h_i) = \tilde{f}(h_i, x_i, \{\mu_j\}_{j \in N(i)})$
- $\tilde{\mu}_i = \tilde{T} \odot (x_i, \{\tilde{\mu}\}_{j \in N(i)})$
- parametrize $\tilde{T}$ before hand
- use any nonlinear function mappings. For instance, we can parameterize it as a neural network
- $\mu_i = \sigma(W_1 x_i + W_2 \Sigma_{j \in N(i)} \tilde{\mu}_j)$

Embedded Mean Field

Algorithm 1 **Embedded Mean Field**

Input: parameter $\mathbf{W}$ in $\widetilde{\mathcal{T}}$
Initialize $\widetilde{\mu}_i^{(0)} = \mathbf{0}$, for all $i \in \mathcal{V}$
for $t = 1$ **to** T **do**
  for $i \in \mathcal{V}$ **do**
    $l_i = \sum_{j \in \mathcal{N}(i)} \widetilde{\mu}_i^{(t-1)}$
    $\widetilde{\mu}_i^{(t)} = \sigma(W_1 x_i + W_2 l_i)$
  end for
end for{fixed point equation update}
return $\{\widetilde{\mu}_i^T\}_{i \in \mathcal{V}}$

Embedding Loopy Belief Propagation

$$\min_{\{q_{ij}\}_{(i,j)\in\mathcal{E}}} -\sum_i (|\mathcal{N}(i)|-1)\int_{\mathcal{H}} q_i(h_i)\log\frac{q_i(h_i)}{\Phi(h_i,x_i)}dh_i + \sum_{i,j}\int_{\mathcal{H}^2} q_{ij}(h_i,h_j)\log\frac{q_{ij}(h_i,h_j)}{\Psi(h_i,h_j)\Phi(h_i,x_i)\Phi(h_j,x_j)}dh_i dh_j$$

$$m_{ij}(h_j) \propto \int_{\mathcal{H}} \prod_{k\in\mathcal{N}(i)\backslash j} m_{ki}(h_i)\Phi_i(h_i,x_i)\Psi_{ij}(h_i,h_j)dh_i,$$

$$q_i(h_i) \propto \Phi(h_i,x_i)\prod_{j\in\mathcal{N}(i)} m_{ji}(h_i).$$

Embedding Loopy Belief Propagation

$$m_{ij}(h_j) = f\left(h_j, x_i, \{m_{ki}\}_{k \in \mathcal{N}(i) \setminus j}\right),$$
$$q_i(h_i) = g\left(h_i, x_i, \{m_{ki}\}_{k \in \mathcal{N}(i)}\right).$$

$$\begin{aligned}
\widetilde{\nu}_{ij} &= \widetilde{\mathcal{T}}_1 \circ \left(x_i, \{\widetilde{\nu}_{ki}\}_{k \in \mathcal{N}(i) \setminus j}\right), \\
\widetilde{\mu}_i &= \widetilde{\mathcal{T}}_2 \circ \left(x_i, \{\widetilde{\nu}_{ki}\}_{k \in \mathcal{N}(i)}\right).
\end{aligned}$$

$$\widetilde{\nu}_{ij} = \sigma\Big(W_1 x_i + W_2 \sum_{k \in \mathcal{N}(i) \setminus j} \widetilde{\nu}_{ki}\Big)$$
$$\widetilde{\mu}_i = \sigma\Big(W_3 x_i + W_4 \sum_{k \in \mathcal{N}(i)} \widetilde{\nu}_{ki}\Big)$$

Algorithm 3 Discriminative Embedding

Input: Dataset $\mathcal{D} = \{\chi_n, y_n\}_{n=1}^N$, loss function $l(f(\chi), y)$.
Initialize $\mathbf{U}^0 = \{\mathbf{W}^0, \mathbf{u}^0\}$ randomly.
for $t = 1$ **to** T **do**
  Sample $\{\chi_t, y_t\}$ uniform randomly from $\mathcal{D}$.
  Construct latent variable model $p(\{H_i^t\}|\chi_n)$ as (5).
  Embed $p(\{H_i^t\}|\chi_n)$ as $\{\widetilde{\mu}_i^n\}_{i \in \mathcal{V}_n}$ by Algorithm 1 or 2 with $\mathbf{W}^{t-1}$.
  Update $\mathbf{U}^t = \mathbf{U}^{t-1} + \lambda_t \nabla_{\mathbf{U}^{t-1}} l(f(\widetilde{\mu}^n; \mathbf{U}^{t-1}), y_n)$.
end for
return $\mathbf{U}^T = \{\mathbf{W}^T, \mathbf{u}^T\}$

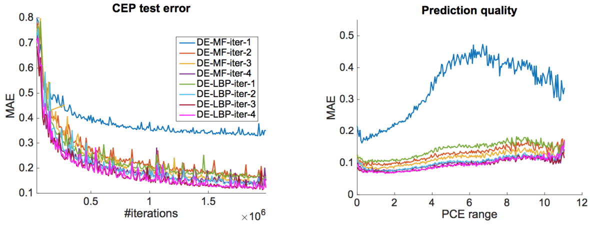
Experiments

- Baselines : Kernels + SVM
- SCOP dataset (7329 sequences)
- FC_RES data: CRISPR Cas9 dataset, whether guide RNA will direct Cas9 to target DNA (5310 guides)

	FC_RES	SCOP
kmer-single	0.7606±0.0187	0.7097±0.0504
kmer-concat	0.7576±0.0235	0.8467±0.0489
mismatch	0.7690±0.0197	0.8637±0.1192
fisher	0.7332±0.0314	0.8662±0.0879
DE-MF	**0.7713±0.0208**	0.9068±0.0685
DE-LBP	0.7701±0.0225	**0.9167±0.0639**

Harvard Clean Energy PRoject

- overall efficiency of the energy conversion process in a solar cell ; power conversion efficiency (PCE)
- expensive simulations for the 2.3 million candidate molecules



(a) Test error vs iterations

(b) Prediction quality

Figure 4: Details of training and prediction results for DE-MF and DE-LBP with different number of fixed point iterations.

	test MAE	test RMSE	# params
Mean Predictor	1.9864	2.4062	1
WL lv-3	0.1431	0.2040	1.6m
WL lv-6	0.0962	0.1367	1378m
DE-MF	0.0914	0.1250	0.1m
DE-LBP	**0.0850**	**0.1174**	0.1m