Bayesian Deep Learning for recommender system

Towards Bayesian Deep Learning : A Survey

I Jeong Han

May 27, 2019

Department of Statistics, University of Seoul

Table of contents

Bayesian Deep Learning

Bayesian Deep Learning(BDL)

BDL(Bayesian Deep Learning) is combined by organically integrating the merits of Neural Networks and Probabilistic Graphical Models in a single principled probabilistic framework.

Deep learning has achieved significant success in many perception tasks including *seeing*, *reading* and *hearing*.

However, the ability of *thinking* such as causal inference, logic deduction, and dealing with uncertainty is apparently beyond the capability of conventional deep learning methods.

Probabilistic graphical model is a model that is proficient in reasoning and uncertainty.

Bayesian Deep Learning(BDL)

Components

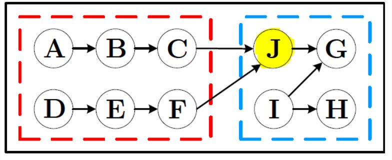
Usually, a BDL model consists of two components, perception component and task-specific component.

- The **perception component** is a Bayesian formulation of a certain type of neural networks.
- The **task-specific component** describes the relationship among different hidden or observed variables using probablistic graphical models.

Bayesian Deep Learning(BDL)

General Framework for BDL

The figure shows the PGM of a simple BDL model as an example.



- Ω_p : the set of perception variables (A,B, and C).
  Usually the set includes the weights and neurons in the probabilistic formulation of a deep learning model.
- Ω_h : the set of hinge variables (J).
  Variables directly interact with Ω_p from Ω_t.
- Ω_t : the set of task variables (G,I,H).
  They have no direct relation to the perception component.

Introduction

Advantage

- The BDL is that it provides a principled way of unifying deep learning and PGM.
- The implicit regularization built into the BDL.
- The BDL provides a principled Bayesian approach of handling parameter uncertainty.

Introduction

Uncertainty
When BDL is applied to complex tasks, there are three kinds of parameter uncertainty that need to be taken into account:

1) Uncertainty on the neural network parameters.
2) Uncertainty on the task-specific parameters.
3) Uncertainty of exchanging information between the perception component and the task-specific component.

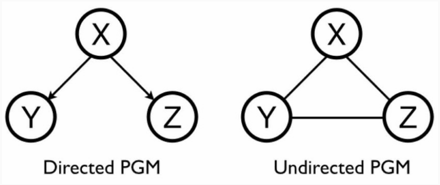
It is worth noting that the third uncertainty could only be handled under a unified framework like BDL.

Probabilistic Graphical Models

Probabilistic Graphical Models(PGM)

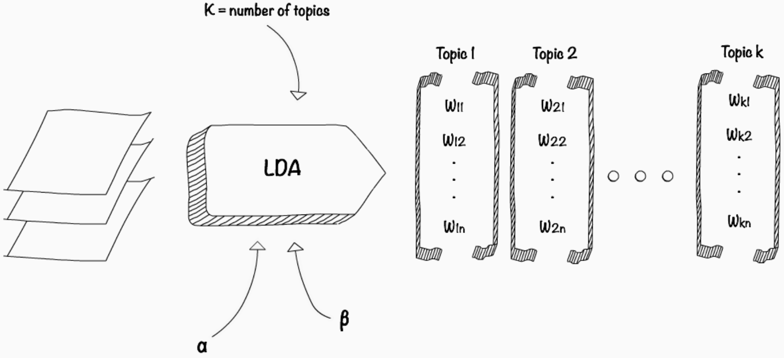
The PGM use diagrammatic representations to describe random variables and relationships among them. It has nodes(vertices) to represent random variables and links(edges) to express probabilistic relationships among them.

There are essentially two types of PGM, the survey mainly focuses on directed PGM (also known as Bayesian networks).



Probabilistic Graphical Models(PGM)

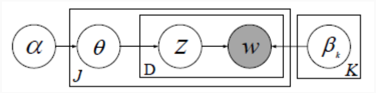
A classic example of PGM would be LDA(latent Dirichlet allocation).



Probabilistic Graphical Models(PGM)

The figure shows the Probabilistic graphical model for LDA(latent Dirichlet allocation).

- J: number of documents, D: number of words in document
- *w* : observed variables, θ,z : latent variables, α, β : parameters



And the generative process is as follows:

- For each document j ($j = 1, 2, 3 \cdots, J$),
  1) Draw topic proportions $\theta_j \sim$ Dirichlet(α)
  2) For each word w_{jn} of item (paper) $\mathbf{w}_{j'}$
     a) Draw topic assignment $z_{jn} \sim$ Mult(θ_j)
     b) Draw word $w_{jn} \sim$ Mult($\beta_{z_{jn}}$)

Stacked Denoising Autoencoders

Stacked denosing autoencoders(SDAE)

Stacked denosing autoencoders, which is a kind of multilayer denosing AE, is a feedforward neural network for learning representations (encoding) of the input data by learning to **predict the clean input itself in the output**.

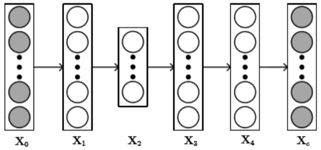


Fig. 1. A 2-layer SDAE with $L = 4$.

- We can randomly set 30% of the entries in $\mathbf{X}_c$ to 0 and get $\mathbf{X}_0$.
- The hidden layer in the middle ($\mathbf{X}_2$) can be constrained to be a bottleneck to learn compact representations.

Stacked denosing autoencoders(SDAE)

An SDAE solves the following optimization problem:

$$\min_{\{\mathbf{W}_l\},\{\mathbf{b}_l\}} \|\mathbf{X}_c - \mathbf{X}_L\|_F^2 + \lambda \sum_l \|\mathbf{W}_l\|_F^2$$

$$\text{subject to } \mathbf{X}_l = \sigma(\mathbf{X}_{l-1}\mathbf{W}_l + \mathbf{b}_l), l = 1, ..., L-1$$
$$\mathbf{X}_L = \mathbf{X}_{L-1}\mathbf{W}_L + \mathbf{b}_L$$

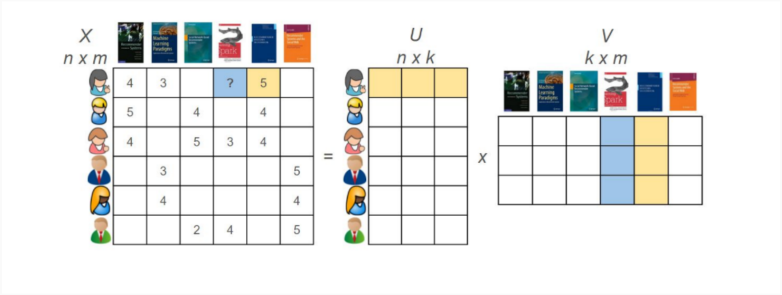
- SDAE can be regarded as a multilayer perception for regression tasks.
- The input $\mathbf{X}_0$ is the corrupted version of the data and the target is the clean version of the data $\mathbf{X}_c$.(denoising)

Recommender Systems

BDL for Recommender Systems

There have been few attempts to develop deep learning model for Collaborative Filtering(CF).

CF breaks the rating matrix into two smaller matrices: a user-centric one and an item-centric one. Then, these two are compressed into a latent space to get detached latent representation of both the users and items.



BDL for Recommender Systems

But CF-based method do not incorporate content information like CTR which is crucial for accurate recommendation.

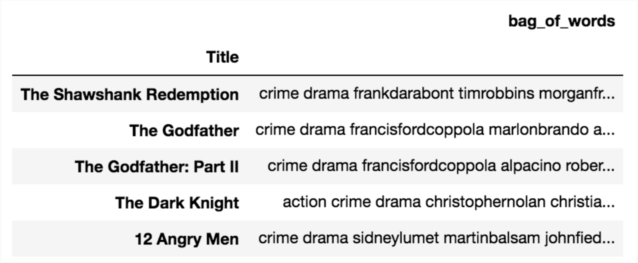
To solve the previous challenge, a hierarchical Bayesian model called collaborative deep learning (CDL) for RS is introduced.

CDL based on Bayesian approach couples deep representation learning for the content information and collaborative filtering for the rating matrix, allowing two-way interaction between two.

Collaborative Deep Learning

Notation and Problem Formulation

- $\mathbf{X}_{c\ J\times B}$: The entire collection of J items (movies), where $\mathbf{X}_{c,j*}$ is row j vector for item j based on a vocabluary size B.



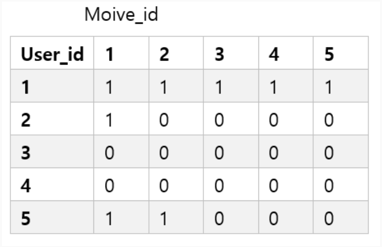
Title	bag_of_words
The Shawshank Redemption	crime drama frankdarabont timrobbins morganfr...
The Godfather	crime drama francisfordcoppola marlonbrando a...
The Godfather: Part II	crime drama francisfordcoppola alpacino rober...
The Dark Knight	action crime drama christophernolan christia...
12 Angry Men	crime drama sidneylumet martinbalsam johnfied...

(action, drama, frankdarabont, timrobbins, morganfr, ...)

$$\mathbf{X}_{c,1} = (1,1,1,1,1,0,0,0...)\ ,\mathbf{X}_{c,2} = (1,1,0,0,0,1,1,0...)$$
$$\mathbf{X}_{c,3} = (1,1,0,0,0,1,1,1...)\ ,\mathbf{X}_{c,4} = (1,1,0,0,0,0,0,0...)$$
$$\mathbf{X}_{c,5} = (1,1,0,0,0,0,0,...)$$

Collaborative Deep Learning

- $\mathbf{R}_{I \times J}$: An rating matrix.
  $\mathbf{R}_{ij} = 1$ if user i has movie j in his or her personal library.



Moive_id

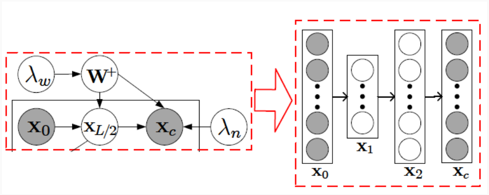
User_id	1	2	3	4	5
1	1	1	1	1	1
2	1	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	1	1	0	0	0

Given part of the ratings in $\mathbf{R}$ and the content information $\mathbf{X}_c$, the problem is to predict the other ratings in R. Also, it is general enough to handle other recommendation tasks (e.g., tag recommendation).

Collaborative Deep Learning

- $\mathbf{X}_c$: The matrix plays the role of clean input to the SDAE.
- $\mathbf{X}_0$: The noise-corrupted matrix.
- $\mathbf{X}_l$: The output of layer l of the SDAE, which is $J \times K_l$ matrix.
  $\mathbf{X}_{l,j*}$: row j of $\mathbf{X}_l$
- $\mathbf{W}_l$, $\mathbf{b}_l$: The weight matrix and bias vector of layer l.
  $\mathbf{W}_{l,n*}$: column n of $\mathbf{W}_l$
- $\mathbf{W}^+$: The collection of all layers of weight matrices and biases for convenience.

Generalized Bayesian SDAE



1) For each layer l of the SDAE network,
    a) For each column n of the weight matrix $\mathbf{W}_l$, draw

$$\mathbf{W}_{l,*n} \sim \mathcal{N}(\mathbf{0}, \lambda_w^{-1}\mathbf{I}_{K_l})$$

    b) Draw the bias vector $\mathbf{b}_l \sim \mathcal{N}(\mathbf{0}, \lambda_w^{-1}\mathbf{I}_{K_l})$.
    c) For each row j of $\mathbf{X}_l$, draw

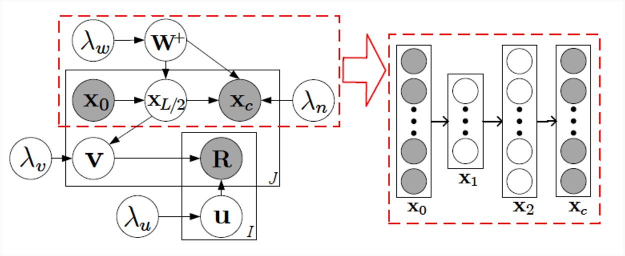
$$\mathbf{X}_{l,j*} \sim \mathcal{N}(\sigma(\mathbf{X}_{l-1,j*}\mathbf{W}_l + \mathbf{b}_l), \lambda_s^{-1}\mathbf{I}_{K_l})$$

2) For each item j, draw a clean input

$$\mathbf{X}_{c,j*} \sim \mathcal{N}(\mathbf{X}_{L,j*}, \lambda_n^{-1}\mathbf{I}_B)$$

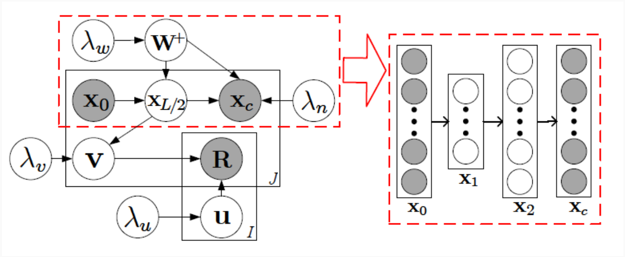
Collaborative Deep Learning

The generative process of CDL is defined as follows:



1) For each layer l of the SDAE network,
   a) For each column n of the weight matrix $\mathbf{W}_l$, draw $\mathbf{W}_{l,*n} \sim \mathcal{N}(\mathbf{0}, \lambda_w^{-1}\mathbf{I}_{K_l})$
   b) Draw the bias vector $\mathbf{b}_l \sim \mathcal{N}(\mathbf{0}, \lambda_w^{-1}\mathbf{I}_{K_l})$.
   c) For each row j of $\mathbf{X}_l$, draw

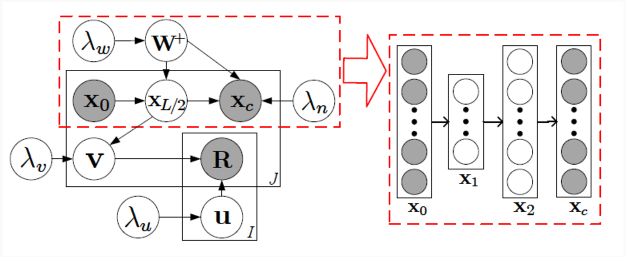
$$\mathbf{X}_{l,j*} \sim \mathcal{N}(\sigma(\mathbf{X}_{l-1,j*}\mathbf{W}_l + \mathbf{b}_l), \lambda_s^{-1}\mathbf{I}_{K_l})$$



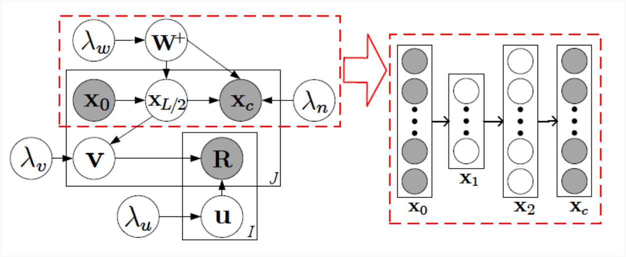
2) For each item j,
   a) Draw a clean input $\mathbf{X}_{c,j*} \sim \mathcal{N}(\mathbf{X}_{L,j*}, \lambda_n^{-1}\mathbf{I}_J)$
   b) Draw the latent item offset vector $\boldsymbol{\epsilon}_j \sim \mathcal{N}(\mathbf{0}, \lambda_\nu^{-1}\mathbf{I}_K)$ and then set the latent item vector: $\mathbf{v}_j = \boldsymbol{\epsilon}_j + \mathbf{X}^T_{\frac{L}{2},j*}$
3) Draw a latent user vector for each user i:

$$u_i \sim \mathcal{N}(\mathbf{0}, \lambda_u^{-1}\mathbf{I}_K)$$

4) Draw a rating $\mathbf{R}_{ij}$ for each user-item pair (i, j): $\mathbf{R}_{ij} \sim \mathcal{N}(\mathbf{u}_i^T\mathbf{v}_j, \mathbf{C}_{ij}^{-1})$



Note that the middle layer $\mathbf{X}_{L/2}$ serves as a bridge between the ratings and content information.
This middle layer, along with the latent offset ϵ_j, is the key that enables CDL to simultaneously learn an effective feature representation and capture the similarity and (implicit) relationship between items (and users). (Refer the latent item vector $\mathbf{v}_j = \epsilon_j + \mathbf{X}^T_{\frac{L}{2},j*}$)



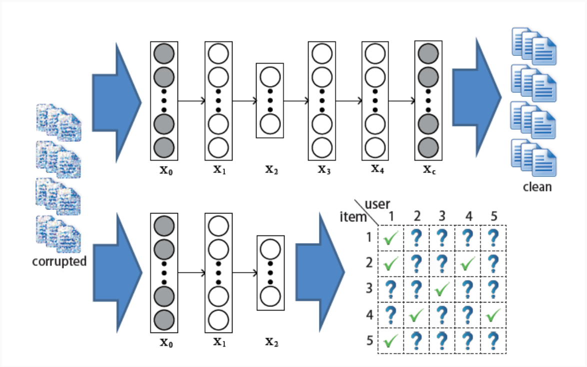
- The perception variables $\Omega_p = \{\{\mathbf{W}_l\}, \{\mathbf{b}_l\}, \{\mathbf{X}_l\}, \mathbf{X}_c\}$
- The hinge variables $\Omega_h = \{\mathbf{V}\}$
- The task variables $\Omega_t = \{\mathbf{U}, \mathbf{R}\}$

Collaborative Deep Learning

Maximizing the posterior probability is equivalent to maximizing the joint log-likelihood of $\mathbf{U}, \mathbf{V}, \{\mathbf{X}_l\}, \mathbf{X}_c, \{\mathbf{W}_l\}, \{\mathbf{b}_l\}$, and $\mathbf{R}$ given hyperparameters λ_-

$$\begin{aligned}\mathcal{L} = &- \frac{\lambda_u}{2} \sum_i \|\mathbf{u}_i\|_2^2 - \frac{\lambda_w}{2} \sum_l (\|\mathbf{W}_l\|_F^2 + \|\mathbf{b}_l\|_2^2) \\ &- \frac{\lambda_v}{2} \sum_j \|\mathbf{v}_j - X_{\frac{L}{2},j*}^T\|_2^2 - \frac{\lambda_n}{2} \sum_j \|\mathbf{X}_{L,j*} - \mathbf{X}_{c,j*}\|_2^2 \\ &- \frac{\lambda_S}{2} \sum_l \sum_j \|\sigma(\mathbf{X}_{l-1,j*}\mathbf{W}_l + \mathbf{b}_l - \mathbf{X}_{l,j*})\|_2^2 - \sum_{i,j} \frac{\mathbf{C}_{ij}}{2} (\mathbf{R}_{ij} - \mathbf{u}_i^T \mathbf{v}_j)^2.\end{aligned}$$

Collaborative Deep Learning



- For $\mathbf{u}_i$ and $\mathbf{v}_j$, block coordinate descent is used.
- Given $\mathbf{U}$ and $\mathbf{V}$, we can learn the $\mathbf{W}_l$ and $\mathbf{b}_l$ using the back-propagation learning algorithm.

Thank you