Probabilistic Deep Models

Qiang Ji

Probabilistic Machine Learning

Deterministic machine learning such as Neural networks learn a mapping function $f(X, \Theta)$ that maps input X to output Y,

- They cannot effectively capture the uncertainties in the data and in the model
- They cannot quantify the uncertainty or confidence of their outputs.
- They are based on point estimation and they tend to overfit

Introduction

- Probabilistic machine learning
- Bayesian Neural Networks
- Deep Boltzmann Machine (DBM)
- Deep Regression Bayesian Network (DRBNs)
- Deep Belief Networks (DBNs)

Probabilistic Machine Learning

Probabilistic machine learning constructs the probability distributions of X and Y, and use the distribution to perform the classification.

- Generative approach $-p(X,Y|\Theta)$

  Learn the parameters Θ that characterizes the joint probability of X and Y, and performs the classification/regression using

  $Y=\text{argmax}_Y\, p(Y|X, \Theta)$
- Discriminative approach $- p(Y|X, \Theta)$

  Learn the parameters Θ that characterizes the conditional joint probability of Y given X , and performs the classification/regression using

  $Y=\text{argmax}_Y\, p(Y|X, \Theta)$
- Θ is learnt through maximum likelihood estimation, i.e., $\Theta^* = \text{argmax}_\Theta\, p(D|\Theta)$

Bayesian Machine Learning

As an extension to probabilistic machine learning, Bayesian Machine learning includes a prior on the model parameters, i.e., $P(\Theta|\boldsymbol{\alpha})$, where $\boldsymbol{\alpha}$ are the hyper-parameters that specify the prior probability of Θ.

- As Θ are treated as RV, there is no parameter learning.
- Hyper-parameters $\boldsymbol{\alpha}$ are either manually specified or are learned by maximizing tis likelihood, i.e.,

  $\alpha^* = \text{argmax}_\alpha\ p(\mathbf{D}|\alpha)$

  where **D** is the training data.

Bayesian Machine Learning (cont'd)

Compared to non-Bayesian learning, Bayesian learning has the following advantages:

- Bayesian inference does not need estimate parameters Θ, i.e., inference is independent of parameters Θ. This is accomplished by marginalizing out Θ. This is fundamentally different from maximum likelihood learning, which performs a point estimate of Θ via maximization. It hence replaces maximization by marginalization.
- Maximum likelihood inference uses the learnt parameters Θ^* to perform prediction, i.e., $p(y|x, \Theta^*)$, while Bayesian inference, through its integration, uses the prediction results from all parameters. Bayesian learning hence avoids the over-fitting problem with maximum likelihood learning.
- Bayesian inference produces not only outputs but also generates its distribution, based on which we can quantify the output confidence (or uncertainty).

Bayesian Machine Learning (cont'd)

Given an input X, inference of output Y can be done through empirical or full Bayesian

- Empirical Bayesian inference

  $Y^* = \text{argmax}_Y\ p(Y|X, D, \alpha^*)$

  where $p(Y \mid X, D, \alpha^*) = \int p(Y, \Theta \mid X, D, \alpha^*) d\Theta = \int p(Y \mid X, \Theta)\, p(\Theta \mid D, \alpha^*) d\Theta$

- Full Bayesian inference

  $Y^* = \text{argmax}_Y\ p(Y|X, D)$

  where $p(Y \mid X, D) = \int\int p(Y, \Theta, \alpha \mid X, D) d\Theta d\alpha = \int\int p(Y \mid X, \Theta)\, p(\Theta \mid D, \alpha)\, p(\alpha \mid D) d\Theta d\alpha$

where $p(Y|\Theta,X)$ is the parameter likelihood, $p(\Theta|D,\alpha)$ is the the posterior distribution of Θ, and $p(\alpha|D)$ is the prior distribution of the hyper-parameters.

Bayesian Machine Learning (cont'd)

Bayesian learning has the following disadvantages:

- It needs manually specify or learn the hyper-parameters. Manual specification of the hyper-parameters is inaccurate, while automatic hyper-parameter learning is computationally complex.
- Bayesian inference requires integration over all parameters, which is computationally intractable and cannot scale up well. Approximated and inaccurate solutions are often used to approximate the parameter integration.

Bayesian Neural Networks (BNNs)

- Combine strengths of neural networks with power of stochastic modeling
- Produce not only output but its probability distribution, providing probabilistic guarantee (confidence) on the output, and allowing online performance assessment
- Deal with data, model (parameters and structures), and output uncertainties
- Replace maximization by marginalization, hence avoiding point estimation of parameters and overfitting

See this https://arxiv.org/ftp/arxiv/papers/1801/1801.07710.pdf

BNN Model specification (cont'd)

The parameter likelihood function $p(Y|X, \Theta)$ can be specified as follows

- For regression problem

  Let $\mathbf{X} \ni \boldsymbol{R}^N$ be input vector, $\mathbf{Y} \ni \boldsymbol{R}^K$ be the output vector, $f(X,\boldsymbol{\Theta})$ be the discriminant function outputted by the last layer

  $$\mathbf{Y} = f(\mathbf{X},\Theta) + \boldsymbol{\varepsilon},\ \boldsymbol{\varepsilon} \sim N(0,\Sigma) \qquad p(\mathbf{Y} \mid \mathbf{X},\Theta) = N(f(\mathbf{X},\Theta),\Sigma)$$

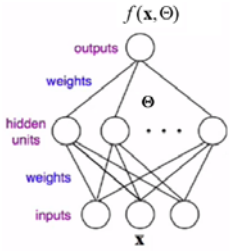
- For classification

  Let $\mathbf{X} \ni \boldsymbol{R}^N$ be the input vector and $\mathbf{Y} \ni \boldsymbol{B}^K$ and $B \ni \{0,1\}$ be the output vector that follows 1-of-K encoding, and $\sigma_m()$ is the softmax function

$$p(\mathbf{Y} \mid \mathbf{X},\Theta) \sim Cat(\sigma_M(f(\mathbf{X},\Theta))) \qquad p(\mathbf{Y} \mid \mathbf{X},\Theta) = \prod_{k=1}^{K} [\sigma_M(f(\mathbf{X},\Theta))[k]]^{Y[k]}$$, where Cat() represents categorical distribution.

BNN Model specification

- Use a NN to capture the relationships between input X and output Y, i.e., $f(X,\Theta)$: X -> Y.
- Specify the prior distribution of the model parameters Θ, e.g. $p(\Theta|\boldsymbol{\alpha}) \sim N(0,I)$, non-informative prior, where I is identity matrix and α is hyper-parameters.
- Specify the parameter likelihood function as $p(Y|X, \Theta)$



BNN Model specification (cont'd)

The posterior probability distribution of the parameter $p(\Theta \mid \mathbf{D}, \alpha)$ can be specified as follows

$$\text{Given training data } \mathbf{D} = (\mathbf{X}_i, \mathbf{Y}_i), i = 1,2,..,N$$

$$p(\Theta \mid \mathbf{D},\alpha) \propto p(\Theta,\mathbf{D},\alpha) = p(\mathbf{D} \mid \Theta,\alpha)\,p(\Theta,\alpha)$$

$$= \prod_i p(\mathbf{Y}_i \mid \mathbf{X}_i,\Theta)\,p(\mathbf{X}_i \mid \Theta,\alpha)\,p(\Theta \mid \alpha)p(\alpha)$$

$$\propto p(\Theta \mid \alpha)\prod_i p(\mathbf{Y}_i \mid \mathbf{X}_i,\Theta)$$

BNN Model Learning –Maximum Likelihood Learning

Given the training data **D**=($\mathbf{X}_i$, $\mathbf{Y}_i$), i=1,2,..N, learn the parameters by maximizing the log parameter likelihood (same as minimizing the negative loglikeligood), i.e

$$\Theta^* = \arg\max_{\Theta} \log p(\mathbf{D} \mid \Theta)$$

where

$$\log p(\mathbf{D} \mid \Theta) = \sum_{i=1}^{N} \log p(\mathbf{Y}_i \mid \mathbf{X}_i, \Theta)$$

BNN Model Learning –Maximum Likelihood Learning (cont'd)

For multi-class classification problem,

$$p(\mathbf{Y}_i \mid \mathbf{X}_i, \Theta) = \prod_{k=1}^{K} [\sigma_M(f(\mathbf{X}_i, \Theta))[k]]^{\mathbf{Y}_i[k]}$$

$$\log p(\mathbf{D} \mid \Theta) = \sum_{i=1}^{N} \log\ p(\mathbf{Y}_i \mid \mathbf{X}_i, \Theta) = \sum_{i=1}^{N} \log \prod_{k=1}^{K} [\sigma_M(f(\mathbf{X}_i, \Theta_k))[k]]^{\mathbf{Y}_i[k]}$$

$$= \sum_{i=1}^{N} \sum_{k=1}^{K} \mathbf{Y}_i[k] \log \sigma_M(f(\mathbf{X}_i, \Theta_k))[k]$$

BNN Model Learning –Maximum Likelihood Learning (cont'd)

For regression problem,

$$p(\mathbf{Y}_i \mid \mathbf{X}_i, \Theta) = N(f(\mathbf{X}_i, \Theta), \Sigma(\mathbf{X}_i, \Theta))$$

$$= \frac{1}{\sqrt{2\pi \mid \Sigma(\mathbf{X}_i, \Theta) \mid}} \exp\left(-\frac{[\mathbf{Y}_i - f(\mathbf{X}_i, \Theta)]^t \Sigma^{-1}(\mathbf{X}_i, \Theta)[\mathbf{Y}_i - f(\mathbf{X}_i, \Theta)]}{2}\right)$$

$$\log p(\mathbf{D} \mid \Theta) = \sum_{i=1}^{N} \log p(\mathbf{Y}_i \mid \mathbf{X}_i, \Theta)$$

$$= \frac{-N}{2} \log(2\pi \mid \Sigma(\mathbf{X}_i, \Theta) \mid) - \sum_{i=1}^{N} \frac{[\mathbf{Y}_i - f(\mathbf{X}_i, \Theta)]^t \Sigma^{-1}(\mathbf{X}_i, \Theta)[\mathbf{Y}_i - f(\mathbf{X}_i, \Theta)]}{2}$$

BNN Model Learning –MAP Learning

Learn the parameters by maximizing the posterior probability distribution of the parameter p(Θ |D, α), i.e.,

$$\Theta^* = \arg\max_{\Theta} \log p(\Theta \mid \mathbf{D}, \alpha)$$

where

$$\log p(\Theta \mid \mathbf{D}, \alpha) = \log p(\Theta \mid \alpha) + \sum_{i=1}^{N} \log p(\mathbf{Y}_i \mid \mathbf{X}_i, \Theta)$$

BNN Inference with point estimation

Given Θ^* and input **X**, inference of **Y*** based on point estimation can be accomplished via

For regression

$$\mathbf{Y}^* = \arg\max_{\mathbf{Y}} p(\mathbf{Y} \mid \mathbf{X}, \Theta^*)$$

$$\mathrm{Var}(\mathbf{Y}^* \mid X) = \Sigma(X, \Theta^*)$$

For classification

$$\mathbf{Y}^*[k] = \begin{cases} 1 \text{ if } k = k^* = \arg\max_k p(\mathbf{Y}[k] \mid X), \text{ where } p(\mathbf{Y}[k] \mid X) = \boldsymbol{\sigma}_M(f(X, \Theta^*)) \\ 0 \text{ else} \end{cases}$$

$$E(\mathbf{Y}^*) = \boldsymbol{\sigma}_M(f(X, \Theta^*))$$

$$\Sigma_{(\mathbf{Y}^* \mid X)} = -[\boldsymbol{\sigma}_M(f(X, \Theta^*))][\boldsymbol{\sigma}_M(f(X, \Theta^*))]^t + diag[\boldsymbol{\sigma}_M(f(X, \Theta^*))]$$

Uncertainty of $\mathbf{Y}^* = \mathrm{trace}(\Sigma_{(\mathbf{Y}^* \mid X)})$ or by its entropy

$$\mathrm{H}(\mathbf{Y}^* \mid X) = -\sum_{k=1}^{K} p(\mathbf{Y}^*[k]) \log p(\mathbf{Y}^*[k])$$

Empirical Bayesian Inference

Empirical BNN inference is to predict output Y given X, D, and α, where α is either given or learnt from data D

$$Y^* = \max_Y p(Y \mid X, D, \boldsymbol{\alpha})$$

where

$$p(Y \mid X, D, \boldsymbol{\alpha}) = \int_\Theta p(Y, \Theta \mid X, D, \boldsymbol{\alpha}) d\Theta$$

$$= \int_\Theta p(Y \mid \Theta, X)\, p(\Theta \mid D, \boldsymbol{\alpha}) d\Theta$$

Bayesian BNN Inference

- Empirical Bayesian inference
- Full Bayesian Inference

Full Bayesian Inference

Given X and training data D, full BNN inference is to predict output Y by

$$Y^* = \max_Y p(Y \mid X, D)$$

where

$$p(Y \mid X, D) = \int_\Theta \int_{\boldsymbol{\alpha}} p(Y, \Theta, \boldsymbol{\alpha} \mid X, D) d\Theta d\boldsymbol{\alpha}$$

$$= \int_\Theta \int_{\boldsymbol{\alpha}} p(Y \mid \Theta, X)\, p(\Theta \mid \boldsymbol{\alpha}, D)\, p(\boldsymbol{\alpha} \mid D) d\Theta d\boldsymbol{\alpha}$$

Parameter Integration Approximation

Both empirical and full Bayesian inference require integration over the parameters. Such parameter integration becomes intractable for a large number of parameters. Approximations are often used to approximate parameter integration

- Sampling method
- Variational method

Sampling method (cont'd)

- Full Inference

$$Y^* = \arg\max_Y p(Y \mid X, D)$$
$$= \arg\max_Y \iint_{\Theta\ \alpha} p(Y \mid X, \Theta) p(\Theta \mid D, \alpha)\, p(D \mid \alpha) d\Theta d\alpha$$
$$\approx \arg\max_Y \frac{1}{S_\alpha S_\Theta} \sum_{s_\alpha=1}^{S_\alpha} \sum_{s_\Theta=1}^{S_\Theta} p(Y, \mid \Theta_s, X), \text{ where } \Theta_s \sim \mathrm{p}(\Theta \mid D, \alpha_s), \alpha_s \sim p(\alpha \mid D)$$
$$= \frac{1}{S_\alpha S_\Theta} \sum_{s_\alpha=1}^{S_\alpha} \sum_{s_\Theta=1}^{S_\Theta} \arg\max_{Y_{s_\alpha s_\theta}^*} p(Y_{s_\alpha s_\theta}, \mid \Theta_s, X)$$

$$Var(Y \mid X, D) = E(Y^2 \mid X, D) - E^2(Y \mid X, D)$$
$$= E_{p(D\mid\alpha)}\{E_{p(\Theta\mid D,\alpha)}[Var\,(Y^2 \mid X, \Theta)]\} + E_{p(D\mid\alpha)}\{Var_{p(\Theta\mid D,\alpha)}[E(Y \mid X, \Theta)]\} + Var_{p(D\mid\alpha)}\{E_{p(\Theta\mid D,\alpha)}[E\,(Y \mid X, \Theta)]\}$$

Sampling method

- Empirical integration

$$Y^* = \arg\max_Y p(Y \mid X, D, \alpha)$$
$$= \arg\max_Y \int_\Theta p(Y \mid X, \Theta) p(\Theta \mid D, \alpha)$$
$$\approx \arg\max_Y \frac{1}{S} \sum_{s=1}^{S} p(Y, \mid \Theta_s, X), \text{ where } \Theta_s \sim \mathrm{p}(\Theta \mid D, \alpha)$$
$$= \frac{1}{S} \sum_{s=1}^{S} \arg\max_{Y_s^*} p(Y_s, \mid \Theta_s, X)$$
$$Var(Y \mid X, D, \alpha) = E(Y^2 \mid X, D, \alpha) - E^2(Y \mid X, D, \alpha)$$
$$= E_{p(\Theta\mid D,\alpha)}[E(Y^2 \mid X, \Theta) - E^2_{p(\Theta\mid D,\alpha)}[E(Y \mid X, \Theta)]$$
$$= E_{p(\Theta\mid D,\alpha)}[Var(Y \mid X, \Theta)] + E^2(Y \mid X, \Theta) - E^2_{p(\Theta\mid D,\alpha)}[E(Y \mid X, \Theta)]$$
$$= E_{p(\Theta\mid D,\alpha)}[Var(Y \mid X, \Theta)] + E^2_{p(\Theta\mid D,\alpha)}[E(Y \mid X, \Theta)] - E^2_{p(\Theta\mid D,\alpha)}[E(Y \mid X, \Theta)]$$
$$= E_{p(\Theta\mid D,\alpha)}[Var(Y \mid X, \Theta)] + Var_{p(\Theta\mid D,\alpha)}[E(Y \mid X, \Theta)]$$

Sampling Methods

- Metropolis hastings sampling
- Hamiltonian Monto Carlo sampling
- Stochastic Gradient Hamiltonian Monte Carlo
- No-U-turn sampler (NUTs)

Check out the latest Tensorflow probability (TFP) that includes some of the sampling functions

Variational methods

Variational inference methods approximate the parameter integration by approximating the posterior parameter distribution p(Θ|D,α) with a simple and factorized distribution q(Θ|D). The mean field is the widely used method, which assumes q(Θ|D) is fully factorizable, i.e.,

$$p(\Theta \mid D, \alpha) \approx q(\Phi \mid D) = \prod_i p(\Phi_i \mid D)$$

$$\Phi^* = \arg\min_{\Phi} KL(q(\Phi \mid D) \,||\, p(\Theta \mid D, \alpha))$$

With fully factorized distribution, the integration can be performed by integrating each parameter independently.

Bayesian Neural Networks

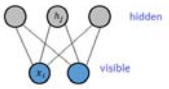
- Introduction to full Bayesian approach
  https://www.youtube.com/watch?v=n6um8qhYLFw&t=16s
- Full Bayesian Learning
  https://www.youtube.com/watch?v=7BR31sfTP60
- The Bayesian interpretation of weight decay
  https://www.youtube.com/watch?v=vEPQNwxd1Y4
- Bayesian Optimization of Hyper-parameters
  https://www.youtube.com/watch?v=cWQDeB9WqvU&t=24s
- Tensorflow probability
  https://medium.com/tensorflow/introducing-tensorflow-probability-dca4c304e245

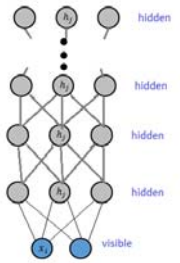
Variational Methods

- Design choice of $q(\Phi|D)$
  - Often choose to be simple for tractable inference e.g. mean-field variational inference assumes fully factorized $q(\Phi|D) = \prod_i p(\Phi_i|D)$
  - More sophisticated structure or model can be used to define q to better reflect the posterior
- Representative variants of variational inference methods
  - Mean-field variational inference [Jordan 1999]
  - Stochastic variational inference [Graves 2011, Paisley 2012, Hoffman 2013, Ranganath 2014]
  - Inference networks [Minh 2014, Kingma 2014, Rezende 2014]
  - Normalizing flows [Rezende 2016]

Probabilistic Deep Models

- Constructed from Probabilistic Graphical models (PGMs), they are usually generative and capture the joint probabilistic distribution of the data (instead of a deterministic mapping function)
- The building block of a deep probabilistic model contains a latent layer **h** and a visible data layer ***x***. *it captures p(h,x).* A deep probabilistic model can be constructed by stacking the building block on top of each other

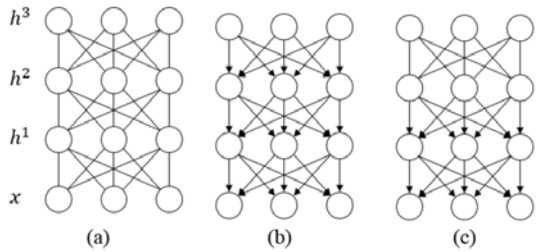




42

Different Deep Probabilistic Models

Depending on the types of the building block, they can be divided into undirected deep models, directed deep models, and hybrid deep models



Deep Boltzmann Machine (undirected)

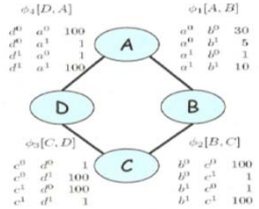
Deep Regression Bayesian Network (directed)

Deep Belief Network (hybrid)

Markov Networks

- A Markov Network (MN) G is an undirected graph that models the probabilistic dependencies of among a set of random variables.
- Nodes represent RVs and links represent mutual dependencies or correlation
- A MN concisely encodes the joint probability of all nodes. Dependencies are captured by the potential function $\phi()$

Alice and Bob study together
Bob and Charles study together
Charles and Debbie study together
Debbie and Alice study together
No interaction between Alice and Charles
No interaction between Dob and Debbie



Probabilistic Graphical Model (PGM)

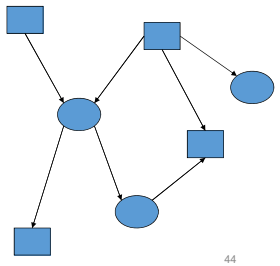
A PGM is a **graphical** model for **compactly** representing **joint** probabilistic distributions among **random variables**.

Random variables are represented by nodes:

Probabilistic interactions among RVs are represented by links:

Undirected links capture correlations between variables (**Undirected graphical model, e.g Markov Network**):

Directed links capture typically causal relationships between variables (**Directed Graphical Model, e.g Bayesian Network**):



Pairwise Markov Network

- Each node is parameterized by a unary potential function $\phi(X_i)$ and a pairwise potential function $\phi(X_i, X_j)$, with its neighbor $X_j \in N_{Xi}$
- Let $E(X_i)$ and $E(X_i, X_j)$ be unary and pairwise energy function, and the exponential potential function can be

$$\phi(X_i) = \exp(-E(X_i)) \quad \phi(X_i, X_j) = \exp(-E(X_i, X_j)),$$

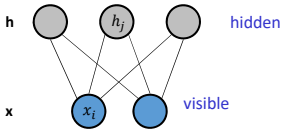
- The joint probability of all nodes $\mathbf{X}=\{X_1, X_2, \ldots, X_N\}$ can be represented by the Gibbs distribution

$$P(\mathbf{X}) = \frac{1}{Z}\exp(-E(\mathbf{X}, \boldsymbol{\Theta})), \text{ where } E(\mathbf{X}, \boldsymbol{\Theta}) = \sum_{i \in G} w_i E_i(X_i) + \sum_{i,j \in G} w_{ij} E(X_i, X_j)$$

$Z = \sum_{\mathbf{X}} \exp\left(-\sum_{i,j \in G} w_{ij}\phi(X_i, X_j) - \sum_{i \in G} w_i \phi(X_i)\right)$ is the normalization term (partition function) and $\Theta=\{w_{ij}, w_i\}$ are the MN parameters

Restricted Boltzmann machines (RBM)

- A undirected graph with a latent layer **h** of binary variables and visible layer of data variables **x**, capturing p(**x**,**h**)
- Each latent variable is connected to each visible variable
- No interactions among nodes in the same layer

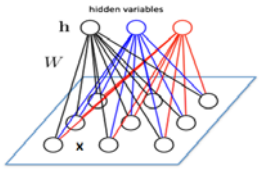


- *RBM is the building block for deep Boltzmann machine*

47

Continuous RBMs

For continuous **x**, we have Gaussian-Bernoulli RBM



Unary x Unary h Pairwise

$$E(\mathbf{x},\mathbf{h}) = -\sum_{x_i \in \mathbf{x}} a_i \frac{(x_i-\mu_i)^2}{2\sigma_i^2} - \sum_{h_j \in \mathbf{h}} b_j h_j - \sum_{i,j} w_{ij} \frac{x_i}{\sigma_i} h_j$$

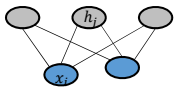
$$= -\mathbf{a}^t(\mathbf{x}-\boldsymbol{\mu}) - \mathbf{b}^t\mathbf{h} - \mathbf{x}^t\mathbf{W}\mathbf{h}$$

$$p(\mathbf{x},\mathbf{h}) = \frac{1}{Z}\exp(-E(\mathbf{x},\mathbf{h}))$$

$$Z = \int_{\mathbf{x}} \sum_{\mathbf{h}} \exp(-E(\mathbf{x},\mathbf{h}))d\mathbf{x}$$

$$\boldsymbol{\Theta} = \{\mathbf{W}, \mathbf{a}, \mathbf{b}, \boldsymbol{\mu}\}$$

Discrete RBM



- For discrete **x**, its energy function of for **h** and **x** is,

Unary x Unary h Pair-wise

$$E(\boldsymbol{x},\boldsymbol{h}) = -\sum_i x_i a_i - \sum_j h_j b_j - \sum_{i,j} x_i h_j w_{ij}$$

$$= -\mathbf{a}^t\mathbf{x} - \boldsymbol{b}^t\boldsymbol{h} - \mathbf{x}^t\boldsymbol{W}\boldsymbol{h}$$

$$p(\mathbf{x},\boldsymbol{h}) = \frac{1}{Z_\theta}\exp(-E(\mathbf{x},\boldsymbol{h},\theta))$$

- $\boldsymbol{\Theta} = \{\boldsymbol{W},\boldsymbol{a},\boldsymbol{b}\}$, where **W**=$\{w_{ij}\}$, **a**=$\{a_i\}$, and **b**=$\{b_j\}$. They are the model parameters, which respectively represent the visible-to-hidden node interactions, the visible node bias and the hidden node bias.

- Z_θ is the partition function, and $Z_\theta = \sum_{x\in\boldsymbol{x}}\sum_{h\in\boldsymbol{h}}\exp(-E(\mathbf{x},\boldsymbol{h}))$

48

RBM Parameter learning

- Given dataset $\{\mathbf{x}^m\}_{m=1}^M$, the parameter learning task is to estimate the parameters $\boldsymbol{\theta}$ by maximizing the log marginal likelihood,

$$\theta^* = \arg\max_\theta \sum_m \log p(\mathbf{x}^m) = \sum_m log \sum_{\boldsymbol{h}} P(\boldsymbol{x}^m,\boldsymbol{h})$$

- Method: gradient ascent with contrastive divergence (CD),

$$\frac{\partial \sum_m logp(\mathbf{x}^m)}{\partial\theta} = \frac{1}{M}\sum_m\sum_h \frac{\partial E(\mathbf{x}^m,\boldsymbol{h})}{\partial\theta} - \sum_{\boldsymbol{x},\boldsymbol{h}} p(\boldsymbol{x},\boldsymbol{h})\frac{\partial E(\boldsymbol{x}^m,\boldsymbol{h})}{\partial\theta}$$

where the first and second terms are data and model expectation (averages), respectively. The second term comes from taking derivative of the partition function. It is computed by sampling using current parameters-contrastive divergence (CD) method

$$\theta_{t+1} = \theta_t + \eta \frac{\partial logp(\mathbf{X};\theta)}{\partial\theta}$$

50

Inference in RBMs

Posterior probability inference P(**h**|**x**) or p(x|h)

1) Bottom up inference

$$p(\mathbf{h}\mid\mathbf{x})=\prod_{j=1}^{H}p(h_j\mid\mathbf{x}),\ \text{where}\ p(h_j\mid\mathbf{x})=\frac{p(h_j,\mathbf{x})}{\sum_{h_j}p(h_j,\mathbf{x})}=\frac{\sum_{\mathbf{h}_{-j}}p(h_j,\mathbf{h}_{-j},\mathbf{x})}{\sum_{\mathbf{h}_{-j}}\sum_{h_j}p(h_j,\mathbf{h}_{-j},\mathbf{x})}$$

2) Top down inference

$$p(\mathbf{x}\mid\mathbf{h})=\prod_{i=1}^{N}p(x_i\mid\mathbf{h}),\ \text{where}\ p(x_i\mid\mathbf{h})=\frac{p(x_i,\mathbf{h})}{\sum_{x_i}p(x_i,\mathbf{h})}=\frac{\sum_{\mathbf{x}_{-i}}p(x_i,\mathbf{x}_{-i},\mathbf{h})}{\sum_{x_i}\sum_{\mathbf{x}_{-i}}p(x_i,\mathbf{x}_{-i},\mathbf{h})}$$

51

Inference in RBMs

- MAP inference $\boldsymbol{h}^{*}=\arg\max_{h}P(\boldsymbol{h}|\boldsymbol{x})$

Because the latent variables are conditionally independent, it can be performed individually for h_i.

$$h_j^{*}=\arg\max_{h_j}P(h_j|\boldsymbol{x})$$

52

Inference in RBMs

- Likelihood inference p(**x**)

$$p(\mathbf{x})=\sum_{\mathbf{h}}p(\mathbf{x},\mathbf{h})$$

The number of **h** is too large to enumerate. Approximate method may be used via sampling

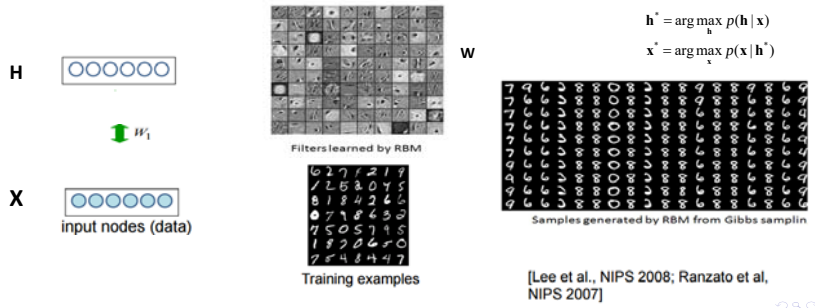
$$p(\mathbf{x})\approx\sum_{\mathbf{h}^{S}\in S}p(\mathbf{x},\mathbf{h}^{S})$$

A better approach is Annealed Importance Sampling (AIS)*

53

RBM for Feature Learning and Data Representation

Modeling Handwritten Digits



[Lee et al., NIPS 2008; Ranzato et al, NIPS 2007]

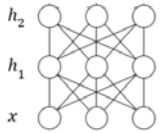
Building Deep Structures

- RBMs can be used as building blocks to construct Deep Boltzmann machines (DBMs).
- A DBM can be constructed by stacking RBM on top of each other
- The joint Probabilities are defined using pairwise and unary potentials:

$$E(\boldsymbol{x}, \boldsymbol{h}_1, \boldsymbol{h}_2) = -\boldsymbol{x}^T \boldsymbol{W}_1 \boldsymbol{h}_1 - \boldsymbol{h}_1^T \boldsymbol{W}_2 \boldsymbol{h}_2 - \boldsymbol{a}^T \boldsymbol{x} - \boldsymbol{b}_1^T \boldsymbol{h}_1 - \boldsymbol{b}_2^T \boldsymbol{h}_2$$

$$p(\boldsymbol{x}, \boldsymbol{h}_1, \boldsymbol{h}_2) = \frac{1}{Z_\theta} \exp(-E(\boldsymbol{x}, \boldsymbol{h}_1, \boldsymbol{h}_2))$$

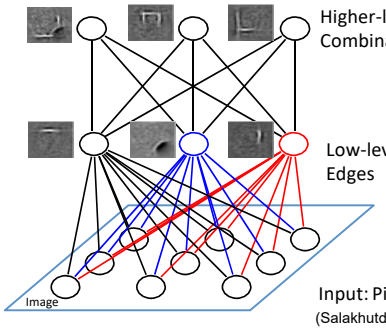
$\boldsymbol{\theta}=\{\mathbf{W}_1, \mathbf{W}_2, \mathbf{a}, \mathbf{b}_1, \mathbf{b}_2\}$



Deep Boltzmann Machine

55

Deep Boltzmann Machines



Higher-level features: Combination of edges

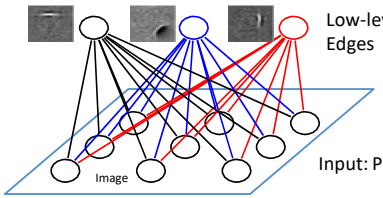
Low-level features: Edges

Built from **unlabeled** inputs.

Input: Pixels

(Salakhutdinov 2008, Salakhutdinov & Hinton 2012)

Deep Boltzmann Machines



Low-level features: Edges

Built from **unlabeled** inputs.

Input: Pixels

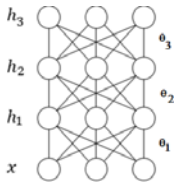
(Salakhutdinov 2008, Salakhutdinov & Hinton 2012)

Parameter learning for DBMs

There are two steps for DBM parameter learning:

1. Layerwise pre-training to obtain initial value for the parameters
   - Learn consecutive layers separately as RBM

$$\theta_l^* = \arg\max_{\theta_l} \sum_m \log p(\mathbf{h}_l^{*m} \mid \theta_l), \text{where } \mathbf{h}_l^{*m} = \arg\max_{\mathbf{h}_l} p(\mathbf{h}_l \mid \mathbf{h}^m{}_{l-1}), l = 1, 1, 2, .., L-1$$



Deep Boltzmann machine

58

Parameter learning for DBMs (cont'd)

2. Joint un-supervised parameter refining using the pre-training results as initialization and update all parameters at the same time. Let

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \sum_m \log p(\mathbf{x}^m \mid \boldsymbol{\theta}) = \sum_m \log \sum_{h_1,h_2,\ldots,h_L} p(h_1,h_2,\ldots,h_L,\mathbf{x}^m \mid \boldsymbol{\theta}) \qquad \boldsymbol{\theta} = \{\theta_1,\theta_2,\ldots,\theta_L\}$$

$$\nabla\boldsymbol{\theta} = \frac{\partial \sum_m \log p(\mathbf{x}^m \mid \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{1}{M}\sum_m \sum_{h^1,h^2,\ldots,h^L} \frac{\partial E(h_1,h_2,\ldots,h_L,\mathbf{x}^m)}{\partial \boldsymbol{\theta}} - \sum_{h^1,h^2,\ldots,h^L} p(h_1,h_2,\ldots,h_L,\mathbf{x}^m)\frac{\partial E(h_1,h_2,\ldots,h_L,\mathbf{x}^m)}{\partial \boldsymbol{\theta}}$$

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t + \eta\nabla\boldsymbol{\theta}$$

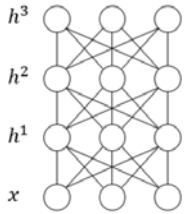
Like RBM learning, it can be solved with CD

Deep Boltzmann Machine Summary

Training	Inference	Structure	Advantages
Need to estimate the gradient of the partition function	Intractable and requires approximation methods: Sampling or variational	Limited to a few (3) hidden layers	Fully capture the joint probability distribution of the data
The gradient of a parameter consists of two parts, the unnormalized likelihood and the partition function. $\nabla_\theta \log P(x)$ $= \nabla_\theta \log P^*(x)$ $- \nabla_\theta \log Z$	Because in a multi-layer DBM, the latent variables are no longer independent given the observations.	Computationally changing during both learning and inference	

DBM inference

1. Posterior probability inference
   - $p(h_l|\mathbf{x})$ and $p(\mathbf{x}|h_l)$ -both can no longer factorize (see fig) . They can be approximated by variational method or pseudo-likelihood or Gibbs sampling
2. MAP inference
   - $h^*=\text{argmax}_h\, p(h|x)$-coordinate ascent



Generative Model of Handwritten Digits

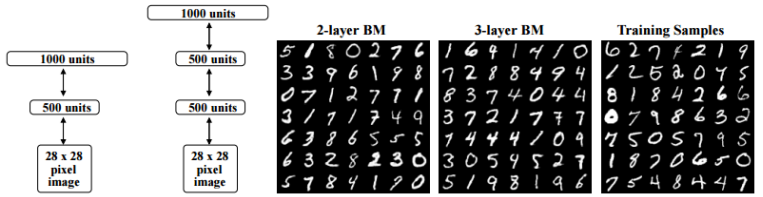


Figure 4: **Left:** Two deep Boltzmann machines used in experiments. **Right:** Random samples from the training set, and samples generated from the two deep Boltzmann machines by running the Gibbs sampler for 100,000 steps. The images shown are the *probabilities* of the binary visible units given the binary states of the hidden units.

Generative Model of 3-D Objects

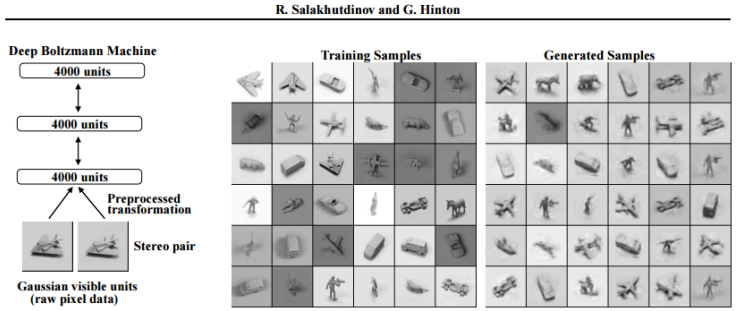
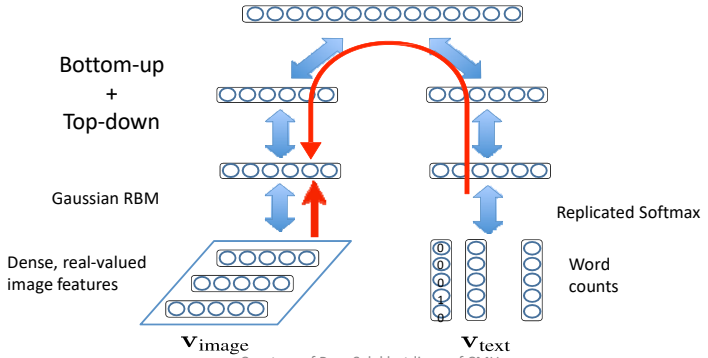


Figure 5: **Left:** The architecture of deep Boltzmann machine used for NORB. **Right:** Random samples from the training set, and samples generated from the deep Boltzmann machines by running the Gibbs sampler for 10,000 steps.

Multimodal DBM



Courtesy of Russ Salakhutdinov of CMU

Handwriting Recognition

MNIST Dataset
60,000 examples of 10 digits

Learning Algorithm	Error
Logistic regression	12.0%
K-NN	3.09%
Neural Net (Platt 2005)	1.53%
SVM (Decoste et.al. 2002)	1.40%
Deep Autoencoder (Bengio et. al. 2007)	1.40%
Deep Belief Net (Hinton et. al. 2006)	1.20%
DBM	**0.95%**

Optical Character Recognition 42,152 examples of 26 English letters

Learning Algorithm	Error
Logistic regression	22.14%
K-NN	18.92%
Neural Net	14.62%
SVM (Larochelle et.al. 2009)	9.70%
Deep Autoencoder (Bengio et. al. 2007)	10.05%
Deep Belief Net (Larochelle et. al. 2009)	9.68%
DBM	**8.40%**

Permutation-invariant version.

Courtesy of Russ Salakhutdinov of CMU

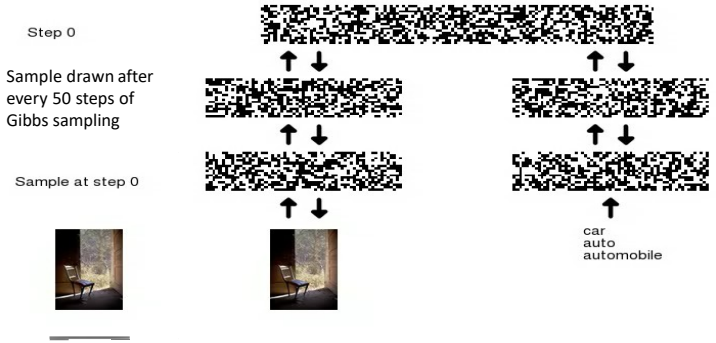
Text Generated from Images

Given	Generated
	dog, cat, pet, kitten, puppy, ginger, tongue, kitty, dogs, furry
	sea, france, boat, mer, beach, river, bretagne, plage, brittany
	portrait, child, kid, ritratto, kids, children, boy, cute, boys, italy

Given	Generated
	insect, butterfly, insects, bug, butterflies, lepidoptera
	graffiti, streetart, stencil, sticker, urbanart, graff, sanfrancisco
	canada, nature, sunrise, ontario, fog, mist, bc, morning

Courtesy of Russ Salakhutdinov of CMU

Images from Text



Courtesy of Russ Salakhutdinov of CMU

Reading materials

G.E. Hinton, S. Osindero, and Y. Teh, "A Fast Learning Algorithm for Deep Belief Nets," Neural Computation, Vol. 18, pp. 1527-1544, 2006.

H.Larochelle and Y. Bengio, "Classification using Discriminative Restricted Boltzmann Machines", ICML 2008.

G. E. Hinton, "Products of Experts," Proc. Int'l Conf. Artificial Neural Networks, 1999.

R. Salakhutdinov and G. Hinton, "Deep Boltzmann Machines," AISTATS 2009.

Images from Text

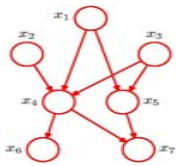
Courtesy of Russ Salakhutdinov of CMU

Bayesian Networks (BN)

A Bayesian Network is a Directed Acyclic Graph (DAG) that models the dependences among a set of random variables. It consists of:

•the qualitative part, i.e. the graph G=(V,E)

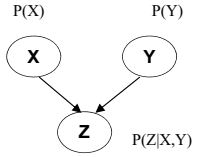
- Nodes (V)-random variables
- Edges (E) –capture directed/causal relations
- No directed cycles



138

Bayesian Networks (cont'd)

- Quantitative part
  - Conditional Probability Distribution (CPD) specifications
    - the conditional probability of each variable given its parents $p(x|\pi(x))$
    - for the root node, specify its prior probability

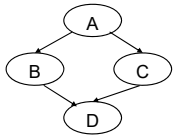


142

Bayesian Networks (cont'd)

- Quantitative part
  - Joint probability

$$P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} p(X_i \mid \pi(X_i))$$

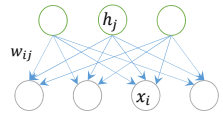


p(A,B,C,D)=p(A)p(B|A)p(C|A)p(D|B,C)

143

Regression Bayesian Network

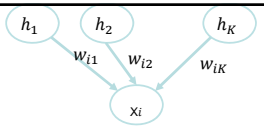
- Regression Bayesian networks (RBNs):
  - Two layer directed graph-latent layer (h) and visible layer (x)
  - Latent variables are binary and are the parents of visible variables
  - No connections among nodes in the same layer
  - Each link is associated with a weight parameter conditional probability is a function of the weights
- Building block for directed deep model



Regression Bayesian network

146

Binary Regression Bayesian Network



Parameterization of RBNs with binary input variable x

$$p(h_j = 1) = \sigma(d_j) \Rightarrow p(h_j) = \frac{e^{h_j d_j}}{1 + e^{d_j}}$$

$$p(x_i = 1 \mid \mathbf{h}) = \sigma(\sum_j h_j w_{ij} + b_i) \Rightarrow p(x_i \mid \mathbf{h}) = \frac{e^{x_i(\sum_j h_j w_{ij} + b_i)}}{1 + e^{\sum_j h_j w_{ij} + b_i}}$$

where σ is a sigmoid function. Each visible node x_i is a linear combination of its parents **h**, with its probability compactly represented by w_{ij} and b_i.

147

Multi-Class Regression BN

Parameterization of RBNs with multi-class (softmax) input variable x∈{1,2,..,K}

$$p(h_j) = \frac{e^{h_j d_j}}{1 + e^{d_j}}$$

$$p(x_i \mid \mathbf{h}) = \sigma_M(\mathbf{W}_i{}^t \mathbf{h} + \mathbf{b}_i)$$

$$p(x_i = k \mid \mathbf{h}) = \frac{\exp(\sum_j h_j w_{ijk} + b_{ik})}{\sum_{k=1}^{K} \exp(\sum_j h_j w_{ijk} + b_{ik})}, k = 1,2,.., K$$

where h_j is the jth the parent of node X_i. The child is linear combination of parents. The child's probability is parameterized with a softmax function of the weights.

Joint Probability of RBN

$$p(\mathbf{x}, \mathbf{h}) = p(\mathbf{h})\, p(\mathbf{x} \mid \mathbf{h}) = \prod_j p(h_j) \prod_i p(x_i \mid \mathbf{h})$$

Binary RBN

$$p(\mathbf{x}, \mathbf{h}) = \prod_j \frac{e^{h_j d_j}}{1 + e^{d_j}} \prod_i \frac{e^{x_i (\sum_j h_j w_{ij} + b_i)}}{1 + e^{\sum_j h_j w_{ij} + b}}$$

$$= \frac{e^{\sum_j h_j d_j + \sum_i x_i (\sum_j h_j w_{ij} + b_i)}}{\prod_j (1 + e^{d_j}) \prod_i (1 + e^{\sum_j h_j w_{ij} + b})} = \frac{e^{\mathbf{d}^t \mathbf{h} + \mathbf{b}^t \mathbf{x} + \mathbf{x}^t \mathbf{W}_{xh} \mathbf{h}}}{\prod_j (1 + e^{d_j}) \prod_i (1 + e^{\sum_j h_j w_{ij} + b})}$$

Similarly, we can derive joint probability distributions for discrete and continuous RBNs

Continuous Regression BN

For continuous input x, its CPT for each node is can be specified as a linear Gaussian

$$p(h_j) = \frac{e^{h_j d_j}}{1 + e^{d_j}}$$

$$p(x_i \mid \mathbf{h}) \sim N(\sum_j w_{ij} h_j + b_i, \sigma_i^2)$$

where h_j is the jth the parent of node X_i. The child is linear combination of parents. The conditional probability for node x_i is compactly represented by the regression parameters w_{ijk} and b_{ik}.

Parameter learning for RBNs

- Given dataset $D = \{x^m\}_{m=1}^M$, the objective function for ML learning is to find θ that maximizes the log marginal likelihood of the data, i.e.,

$$\theta^* = \arg\max_\theta \sum_m \log \sum_h P(x^m, h)$$

- Directly maximizing the marginal log-likelihood through gradient ascent.

$$\nabla\boldsymbol{\theta} = \frac{\partial \sum_m \log \sum_{\mathbf{h}} p(\mathbf{x}^m, \mathbf{h}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{\sum_m \frac{\partial \log \sum_{\mathbf{h}} p(\mathbf{x}^m, \mathbf{h}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}}{\sum_{\mathbf{h}} p(\mathbf{x}^m, \mathbf{h}, \boldsymbol{\theta})} \text{ and } \boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t + \eta \nabla \boldsymbol{\theta}$$

- Difficulty:
  - Exponential number of terms in the sum over h
  - Sampling, variational, and max out.

RBN Inferences

- Posterior inference p(h|x)
  - p(h|x) is difficult since it cannot factorize over h
  - Pesuido-likelihhood
    - ✓ $p(h|x) = \prod_j P(h_j|x, h_{-j})$
    - ✓ Variational method- $p(h|x) \approx q(h)$
- MAP inference-coordinate ascent
  - Initialize h and update one latent variable with others fixed until congerence.
  - $h_j^{t+1} = \arg\max_{h_j} P(h_j|x, h_{-j}^t)$
- Likelihood inference
  - $p(x) = \sum_h p(h, x) \approx max_h\, p(h, x)$

161

Deep Regression Bayesian Networks

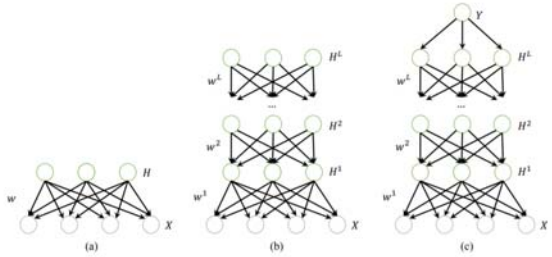
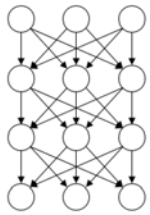


Fig. 2: Graph representation of (a) RBN, (b) DRBN without labels, and (c) DRBN with labels.

Deep Regression Bayesian Network (DRBN)

(Nie, Zheng, and Ji, IEEE SPM, 2018, https://arxiv.org/abs/1710.04809)

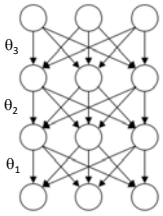
- Constructed by stacking RBNs on top of each other, i.e., every two layers forms a RBN.
- DRBN remains a directed deep model
- For binary RBN, DRBN becomes Sigmoid Belief Network
- For continuous RBN, DRBN becomes Deep Factor Analyzers (DFAs)



Deep RBN-DRBN

166

Deep Regression Bayesian Network Learning



DRBN

- Pre-training: layer-wise training for each layer using an RBN training method

$$\theta_l^* = \arg\max_{\theta_l} \sum_m \log p(\mathbf{h}_l^{*m} \mid \theta_l), \text{where } \mathbf{h}_l^{*m} = \arg\max_{\mathbf{h}_l} p(\mathbf{h}_l \mid \mathbf{h}^m{}_{l-1}), l = 1,1,2,..,L-1$$

- Unsupervised joint training to refine the parameters for all layers simultaneously

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \sum_m \log p(\mathbf{x}^m \mid \boldsymbol{\theta}) = \sum_m \log \sum_{h_1, h_2, ..., h_L} p(h_1, h_2, ..., h_L, \mathbf{x}^m \mid \boldsymbol{\theta})$$

- Supervised joint training, given training data {x^m, y^m},

$$\boldsymbol{\Theta}^* = \underset{\Theta}{\text{argmax}} \sum \log P(y^m|\boldsymbol{x}^m, \boldsymbol{\Theta}) = \underset{\Theta}{\text{argmax}} \sum_m \log \sum_{\boldsymbol{h}} P(y^m, \boldsymbol{h}|\boldsymbol{x}^m, \boldsymbol{\Theta})$$

168

Supervised Fine Tuning

1) One option is to put target variable y at the top and maximize the joint likelihood P(x,y) for fine-tuning,

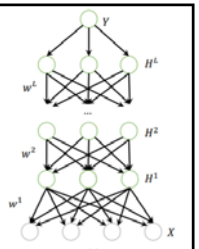
$$\theta^* = \arg\max_{\theta} \sum_i \log \sum_h P(x^i, h, y^i)$$

- The gradient descent algorithm uses the pre-training result as an initialization for the parameters.

2) Alternatively, we can maximize the posterior probability P(y|x) for discriminative learning,

$$\theta^* = \arg\max_{\theta} \sum_i P(y^i|x^i)$$

Optimization through gradient descent.



Deep regression Bayesian network with labels

169

DRBN Application: Image inpainting

Fig. 4: An example of the image inpainting experiment, (a) original image, (b) corrupted image, (c) GMM, and (d) DRBN. Images collected from [19]

DRBN Inference

- p(**h**^{l}|**x**) –approximated with sampling or variational inference or pseudo-likelihood
  - ➢ **h**^{*l}=argmax$_{\mathbf{h}^l}$ p(**h**^{l}|**x**) – feature learning, coordinate ascent method
- p(x|**h**)- reconstruction
- p(x) – likelihood learning for classification, intractable and approximated by sampling or variational inference

DRBN Application: Image Restoration

Fig. 5: Some examples of the face restoration from random noise (a) and block occlusion (b). Images collected from [9]. From top to bottom, the rows represent original images, corrupted images, and reconstructed images.

DRBN Application: Image synthesis

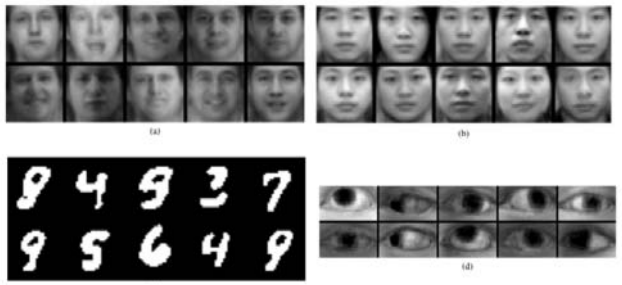


Fig. 6: Examples of the image reconstruction for (a) Multi-PIE, (b) CAS-PEAL, (c) MNIST, (d) UnityEye.

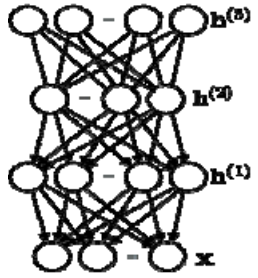
Deep Belief Network (Hinton et al. 2006)

Deep Belief Networks:

- it is a generative model that mixes undirected and directed connections between variables. It models the joint distribution of observed data $\mathbf{x}$ and hidden variables $(\{\mathbf{h}^1, \ldots, \mathbf{h}^l\}), P(\mathbf{x}, \mathbf{h}^1, \ldots, \mathbf{h}^l; \theta)$
- top 2 layers' distribution $p(\mathbf{h}^{(2)}, \mathbf{h}^{(3)})$ is an RBM!
- other layers form a Bayesian network with conditional distributions:

$p(h_j^{(1)} = 1|\mathbf{h}^{(2)}) = \text{sigm}(\mathbf{b}^{(1)} + \mathbf{W}^{(2)}\ \mathbf{h}^{(2)})$

$p(x_i = 1|\mathbf{h}^{(1)}) = \text{sigm}(\mathbf{b}^{(0)} + \mathbf{W}^{(1)}\ \mathbf{h}^{(1)})$



RBM

BN

25

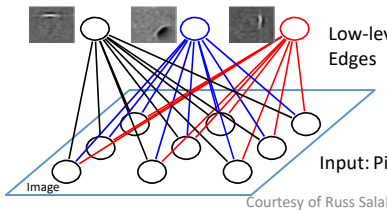
Courtesy of Russ Salakhutdinov of CMU

DRBN Application: Head pose estimation

TABLE IV: MAE of head pose angles in the BU data set.

Method	Yaw	Pitch	Roll
DMF	5.2	4.5	2.6
3D-Deform	4.3	6.2	3.2
MHPE	5.0	3.7	2.9
DVF+CNN	4.3	3.7	2.6
DRBN (IN) [21]	4.8	3.8	3.7
DRBN (CA)	5.4	5.8	3.5
DRBN (AugCA)	4.6	3.5	3.3

Deep Belief Network



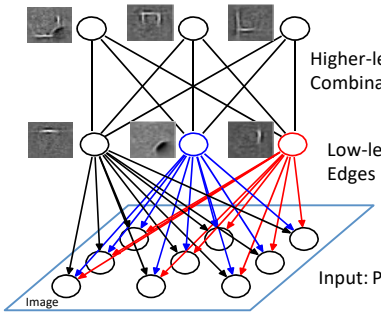
Low-level features: Edges

Built from **unlabeled** inputs.

Input: Pixels

Courtesy of Russ Salakhutdinov and Hinton et al. Neural Computation 2006)

Deep Belief Network



Internal representations capture higher-order statistical structure

Higher-level features: Combination of edges

Low-level features: Edges

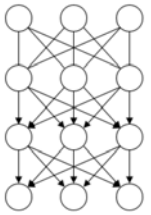
Built from **unlabeled** inputs.

Input: Pixels

(Hinton et.al. Neural Computation 2006)

Parameter learning for DBNs

- Two steps:
  - 1. Greedy layerwise pre-training
    - Every two layers form an RBM, due to the complementary prior, thus can be trained using CD method.
    - Theoretical guarantee to improve log-likelihood.
  - 2. Parameter fine-tuning:
    - Unsupervised: when generating the samples to estimate model expectation, consider both upper and lower layers
    - Supervised: treat target variables as top layer with fixed states, and perform back-propagation.



Deep belief network

183

Deep Belief Network

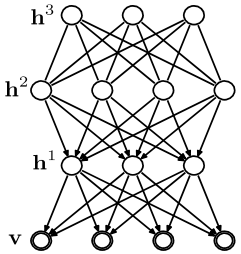
- The joint distribution of a DBN is as follows

$$p(\mathbf{x},\mathbf{h}^1,\mathbf{h}^2,\mathbf{h}^3) = p(\mathbf{h}^2,\mathbf{h}^3)\,p(\mathbf{h}^1 \mid \mathbf{h}^2)\,p(\mathbf{x} \mid \mathbf{h}^1)$$

$$p(\mathbf{h}^2,\mathbf{h}^3) = \exp(-E(\mathbf{b}^2\mathbf{h}^2 + \mathbf{b}^3\mathbf{h}^3 + \mathbf{h}^2\mathbf{W}\mathbf{h}^3))/Z$$

$$p(\mathbf{h}^1 \mid \mathbf{h}^2) = \prod_j p(\mathbf{h}^1_j \mid \mathbf{h}^2)$$

$$p(\mathbf{x} \mid \mathbf{h}^1) = \prod_i p(x_i \mid \mathbf{h}^2)$$



- DBNs use the "complementary priors" to render the latent nodes independent of each other. The details of the complementary prior can be found in Hinton et al. 2006.
- With the complementary priors, DBN becomes DBM.

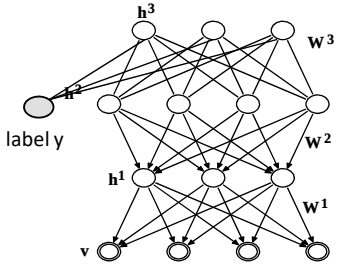
Supervised Learning with DBNs

- If we have access to label information, we can train the joint generative model by maximizing the joint log-likelihood of data and labels

$$\log P(\mathbf{y}, \mathbf{v})$$

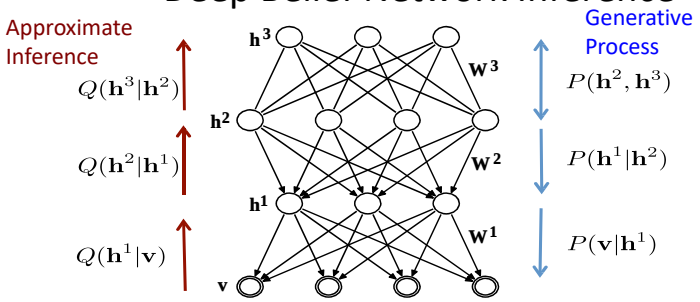
- Discriminative fine-tuning:
  - Use DBN to initialize a multilayer neural network.
  - Maximize the conditional distribution:

$$\log P(\mathbf{y}|\mathbf{v})$$



Courtesy of Russ Salakhutdinov of CMU

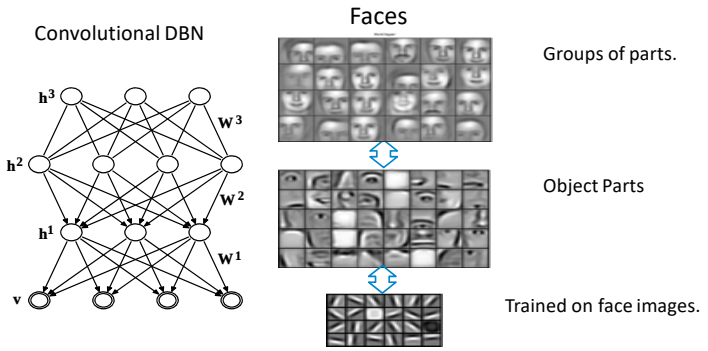
Deep Belief Network Inference



$$Q(\mathbf{h}^t|\mathbf{h}^{t-1}) = \prod_j \sigma\left(\sum_i W^t h_i^{t-1}\right) \qquad P(\mathbf{h}^{t-1}|\mathbf{h}^t) = \prod_j \sigma\left(\sum_i W^t h_i^t\right)$$

Courtesy of Russ Salakhutdinov of CMU

DBN Representation Learning



Courtesy of Russ Salakhutdinov of CMU

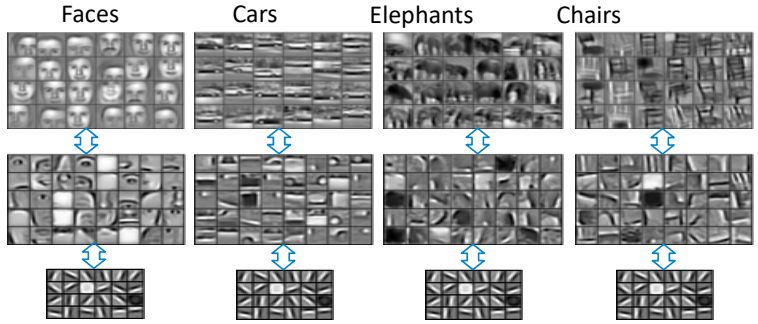
Lee et.al., ICML 2009

DBM, DRBN, and DBN Comparison

	Structure	Parameterization	Learning	Inference
DBM	Undirected	Energy functions	Layerwise + fine tuning with CD	Variational or sampling
DRBN	Directed	Conditional prob.	Layerwise + fine tuning +sampling	Varational or coordinate ascent
DBN	Hybrid	Energy function + Condition prob.	Layerwise + fine tuning with CD	Variational or sampling

- In general, DRBN has better representation but more complex inference. DBM/DBN more efficient in inference but less powerful in representation.
- DBM/DBN must deal with partition function, while DRBN must deal with hidden nodes dependencies

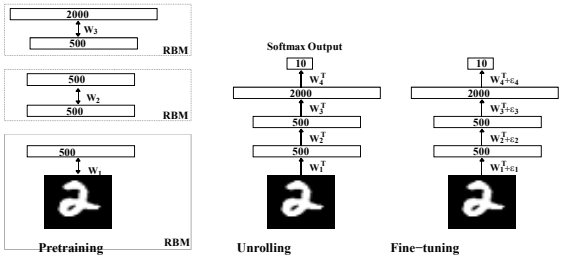
DBN Representation Learning



Courtesy of Russ Salakhutdinov of CMU

Lee et.al., ICML 2009

DBNs for Classification



•After layer-by-layer **unsupervised pretraining**, discriminative fine-tuning by backpropagation achieves an error rate of 1.2% on MNIST. SVM's get 1.4% and randomly initialized backprop gets 1.6%.

•Clearly unsupervised learning helps generalization. It ensures that most of the information in the weights comes from modeling the input data.

Courtesy of Russ Salakhutdinov and (Hinton and Salakhutdinov, Science 2006)

DBNs for Regression

Predicting the orientation of a face patch

Training Data: 1000 face patches of 30 training people.

Test Data: 1000 face patches of **10 new people**.

Regression Task: predict orientation of a new face.

Gaussian Processes with spherical Gaussian kernel achieves a RMSE (root mean squared error) of 16.33 degree.

(Salakhutdinov and Hinton, NIPS 2007)

Courtesy of Russ Salakhutdinov of CMU