Probabilistic Deep Learning for Computer Vision

Qiang Ji
Rensselaer Polytechnic Institute, Troy, NY, USA
jiq@rpi.edu

Introduction

- Probability calculus
- Probabilistic machine learning
- Probabilistic Deep Learning
  - Probabilistic Deep Neural Networks
  - Bayesian Deep Neural Networks

} 1.5 hours

- Deep Probabilistic Graphical Models

} 1.5 hours

  - Deep Boltzmann Machine (DBM)
  - Deep Regression Bayesian Network (DRBNs)
  - Deep Belief Networks (DBNs)

} 1.5 hours

Probability Calculus

- **Random variable**
  - a random variable (RV) is a variable whose value is uncertain, depending on its chance as a result of a random process that maps a RV into a specific value.
- Let capital X represent a RV and lower case $x \in \chi$ represent a particular value of X, where χ *defines the value space*
- *A RV can be discrete or continuous. A discrete RV assumes a finite set of values, while a continuous RV assumes a real value.*

Probability Calculus

- Discrete RV
  - The chance of a discrete RV assuming a particular value is measured by its probability, i.e.,

p(X=x) or p(x) in short , and $0 \le p(x) \le 1$, $\sum_{x\in\chi} p(x) = 1$

- Continuous RV
  - Continuous RV distribution is characterized by its probability density function (pdf)

$$\mathrm{f(x)}: \mathrm{R} -> [0,+\infty),$$

$$\text{and } \; p(X \in A) = \int_A f(x)dx \text{ , where } A \in \chi \text{ is a range of values.}$$

$$\int_{x\in\chi} f(x)dx = 1$$

Probability Calculus (cont'd)

- A **random vector** consists of a vector of RVs. We use bold variable to represent a random vector, i.e., **X**=$(X_1, X_2, \ldots, X_N)^T$, and we use a lower case bold **x** to represent a value vector of **X,** i.e., **X**=**x.** Both **X** and **x** are column vectors.
- p(**X**) represents the probability of the random vector, i.e., the joint probability of all RVs in **X.**

5

Expectation (Mean)

- Expectation

$$E(X) = \sum_{x \in \chi} x \bullet p(x)$$

- Conditional Expectation

$$E(X \mid y) = \sum_{x \in \chi} x \bullet p(x \mid y) dx$$

- For a random vector **X**=$(X_1, X_2, \ldots, X_N)^T$

E(**X**)=$(E(X_1), E(X_2), \ldots, E(X_N))^T$

*Note for continuous RV, sum is replaced by integral and p(x) is replaced by pdf f(x).

6

Variance

- The variance of a RV X

$$Var(X) = \sum_{x \in \chi} (X - E(X))^2 p(x)$$
$$= E[(X - E(X))^2] = E(X^2) - E^2(X)$$

- Standard deviation

$$\sigma_X = \sqrt{Var(X)}$$

- Conditional variance

$$Var(X \mid y) = \sum_{x} (X - E(X \mid y))^2 p(x \mid y)$$
$$= E[(X - E(X \mid y))^2] = E(X^2 \mid y) - E^2(X \mid y)$$

7

Covariance and Covariance Matrix

- Covariance of RVs X and Y

$$\sigma^2{}_{XY} = E[(X - E(X))(Y - E(Y))]$$
$$= E(XY) - E(X)E(Y)$$

- Covariance Matrix-variance of a random vector X=$(X_1, X_2, \ldots, X_N)^t$

$$\Sigma_{\mathbf{X}}^{NxN} = E[(\mathbf{X} - E(\mathbf{X}))(\mathbf{X} - E(\mathbf{X}))^t]$$

Diagonal-variance Off-diagonal-covariance

8

Probability Rules

- Product rule
- Chain rule
- Sum rule
- Conditional probability rule
- Bayes' rule

Product Rule

Given two RVs X and Y, let p(X,Y) represent their joint probability and p(X|Y) represent the conditional probability of X given Y

$$p(X,Y) = p(X|Y)p(Y)$$

i.e., the joint is the product of conditional probability of each variable. The conditional probability can be define as

$$p(X \mid Y) = \frac{p(X,Y)}{p(Y)}$$

Chain Rule

Given three RVs A, B, C ,

- Chain rule :

$$p(A,B,C) = p(A)\,p(B \mid A)\,p(C \mid A,B)$$

Note chain rule expression can vary, depending on the order of the variables.

- Conditional chain rule:

$$p(A,B,C \mid D,E) = p(A \mid D,E)\,p(B \mid A,D,E)\,p(C \mid A,B,D,E)$$

Any assumptions? Chain rule for N variables X_1, X_2, …, X_N?

Sum Rule

- Sum rule via marginalization

$$p(X) = \sum_Y p(X,Y) = \sum_Y \sum_Z p(X,Y,Z)$$

- Sum rule via conditional probability (sum rule + product rule)

$$P(X) = \sum_Y P(X \mid Y)\,p(Y)$$

$$p(X \mid Y) = \sum_Z p(X \mid Y,Z)\,p(Z \mid Y)$$

Bayes' Rule

$$\underbrace{p(X|Y)}_{\text{posterior probability of X}} = \frac{\overbrace{p(X)}^{\text{prior probability of X}}\,\overbrace{p(Y|X)}^{\text{Likelihood of X}}}{p(Y)}$$

p(Y) is a normalization constant and is often called probability of evidence, and

$$p(Y) = \sum_X p(X)\,p(Y|X)$$

Sequential Bayes rule,

$$p(X|Y,Z) = \frac{p(X|Y)\,p(Z|X,Y)}{p(Z|Y)}$$

13

Independence

- If X and Y are marginally independent, then

$$p(X,Y) = p(X)p(Y)$$
$$p(X|Y) = ?$$
$$E(X,Y) = ?$$
$$Cov(X,Y) = ?$$

We denote it as X ⊥Y

14

Conditional Independence

- For three RVs, X, Y, and Z, if X and Y are independent, given Z , we have

$$p(X|Y,Z) = p(X|Z)$$
$$p(X,Y|Z) = ?$$
$$E(XY|Z) = ?$$

We denote it as

$$X \perp Y | Z$$

15

Standard Probability Distributions

- Probability distribution can be discrete or continuous.
- Probability distribution for one RV, i.e., univariate distribution or probability distribution for multiple RVs, i.e., multivariate or joint distribution

16

Discrete Probability Distributions

- Uniform distribution X∈{1,2,..,K}

$$X \sim Uniform(x \mid K) \qquad p(X=k \mid K)=\frac{1}{K}$$

- Bernoulli distribution for a binary RV X∈{0,1}

$$X \sim Ber(x \mid \theta),\ p(X=1)=\theta, p(X=0)=1-\theta$$

Given a Bernoulli RV X with p(X=1)=θ , Y represents the number of times for X=1 out of N trials.

- Binominal distribution
  - Let Y be a positive integer RV. It follows binominal distribution if

$$Y \sim Bin(y \mid N,\theta) \quad p(Y=n_1 \mid N,\theta)=\binom{N}{n_1}\theta^{n_1}(1-\theta)^{N-n_1}$$

17

Discrete Probability Distributions

- Categorical distribution for a discrete RV X∈ $\{1,2,..K\}$.

$$X \sim Cat(x \mid K,\boldsymbol{\theta}), \boldsymbol{\theta} \text{ is a vetor of K probabilties, and}$$

$$p(X=k)=\boldsymbol{\theta}[k] \text{ and } \sum_{k=1}^{K}\boldsymbol{\theta}[k]=1$$

$$p(X)=\prod_{k=1}^{K}[\boldsymbol{\theta}[k]]^{I(X=k)}, \text{where I(X = k) is the indicator function.}$$

- It is an extension to Bernoulli distribution

18

Discrete Probability Distributions

- Multinomial distribution
  - Let **Y**=(Y$_1$,Y$_2$,…,Y$_K$) be K RVs, where Y$_k$ represents a positive integer. **Y** follows a multinomial distribution if

$$\mathbf{Y} \sim Mult(y_1,y_2,...,y_k \mid N,\theta_1,\theta_2,...,\theta_K)=\frac{N!}{y_1!y_2!...y_k!}\theta_1^{y_1}\theta_2^{y_2}...\theta_k^{y_K}$$

$$\sum_{k=1}^{K}\theta_k = 1 \qquad \sum_{k=1}^{K} y_k = N$$

  - **Y** can be generated by repeating a categorical RV X∈ $\{1,2,..,K\}$, with p(X=k)=θ$_k$, N times. θ$_k$ is the number of times X=k out of N trials.
- Binominal distribution is a special case of multinomial distribution, where K=2.

19

Continuous Probability Distributions

- Gaussian distributions
  - Unary:

$$X \sim N(x \mid \mu,\sigma^2)=f(x \mid \mu,\sigma^2)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

  - Multivariate

For a random vector **X** with K dimension:

$$\mathbf{X} \sim N(\mathbf{x} \mid \boldsymbol{\mu},\Sigma)=f(\mathbf{x} \mid \boldsymbol{\mu},\Sigma)=\frac{1}{\sqrt{(2\pi)^k \mid \Sigma \mid}}e^{-\frac{(\mathbf{x}-\boldsymbol{\mu})^t\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})}{2}}$$

*Note we use f(x) to represent pdf for continuous RV

20

Continuous Probability Distributions

- Beta distribution (Probability dist.)
  – For a real RV X $\in$(0,1)

$$\mathbf{X} \sim Beta(x \mid \alpha, \beta) = f(x \mid \alpha, \beta) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1}(1-x)^{\beta-1}$$

$$where\ \ B(\alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\ \ and\ \ \Gamma(n) = (n-1)! \text{ - the gamma function}$$

  - Beta distribution is conjugate to binominal distribution
- Dirichlet distribution (Probability vector dist.)
  – Continuous multivariate probability distributions for K RVs $X_1, X_2, \ldots, X_K$, where $X_i \in (0,1)$ and $\sum_{i=1}^{K} X_i = 1$

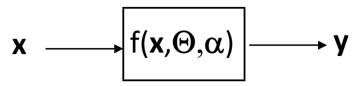
$$\mathbf{X} \sim Dir(x_1, x_2, \ldots, x_K \mid \alpha_1, \alpha_2, \ldots, \alpha_K) = f(x_1, x_2, \ldots, x_K \mid \alpha_1, \alpha_2, \ldots, \alpha_K) = \frac{1}{B(\alpha_1, \alpha_2, \ldots, \alpha_k)} \prod_{i=1}^{K} x_i^{\alpha_i - 1},$$

$$where\ \ B(\alpha_1, \alpha_2, \ldots, \alpha_K) = \frac{\prod_{i=1}^{k} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{k} \alpha_i)}$$

  – Dirichlet distribution is conjugate to multinomial distribution

21

Deterministic Machine Learning



- **x**-input, **y**-output
- f() the mapping function that maps input to output
- Θ and α are the parameters and hyper-parameters of the mapping function
- The goal is to learn the mapping function f() that maps input **X** to output **Y**.

22

Deterministic Machine Learning

- They cannot effectively capture the uncertainties in the data (X) and in the model (Θ).
- They cannot quantify the uncertainty or confidence of their outputs (Y).
- Their prediction is based on point estimation of the parameters (Θ*) and tends to overfit

23

Probabilistic Machine Learning

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

– Generative approach –p(**X**,**Y**| Θ)
  Learn the parameters **Θ** that characterizes the joint probability of **X** and **Y**, and performs the classification/regression using
  Y*=argmax$_{\mathbf{Y}}$ p(**Y**|**X**, **Θ**)

– Discriminative approach – p(Y|X, Θ)
  Learn the parameters **Θ** that characterizes the conditional joint probability of **Y** given **X** , and performs the classification/regression using
  Y*=argmax$_{\mathbf{Y}}$ p(**Y**|**X**, **Θ**)

– Θ are parameters that characterize the probability distributions.

24

Probabilistic Machine Learning v.s. Deterministic Machine Learning

Pros:

- Capable of capturing the input data uncertainties and quantifying the output uncertainties.
- Can identify the input outlier.

Cons:

- Computationally more expensive during both training and prediction.
- Cannot quantify the model uncertainty, i.e., that of Θ.

25

Bayesian Machine Learning

Bayesian Machine models the posterior distribution of the model parameters Θ, P(Q|**D**, **a**), where **D** are the training data and α are the hyper-parameters that specify the prior probability of Q, $p(\Theta \mid \alpha)$.

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

- Empirical inference

  $Y^*=\text{argmax}_Y\ p(\mathbf{Y}|\mathbf{X}, \mathbf{D}, \alpha^*)$
- Full Bayesian inference

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

26

Bayesian Machine Learning (cont'd)

Compared to probabilistic machine learning, Bayesian learning has the following advantages:

- Bayesian inference does not need learn parameters Θ and hence avoids the tedious, time-consuming, and heuristic training process.
- Instead of performing point-based prediction, Bayesian inference performs output prediction using all parameters through parameter integration, hence avoiding the overfitting problem.
- Bayesian inference produces not only outputs but also generates its distribution, based on which we can derive both data and model uncertainties.

27

Bayesian Machine Learning (cont'd)

Bayesian learning has the following disadvantages:

- It needs either 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 as well as the hyper-parameters, which is computationally intractable and cannot scale up well for a large number of parameters.
- Approximated and inaccurate solutions are often used to approximate the parameter integration, hence leading to non-optimal solutions.

28

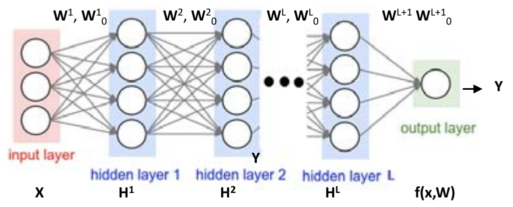
Probabilistic Deep Learning

- Deep Neural Networks
  - Probabilistic Deep Neural Networks
  - Bayesian Deep Neural Networks
- Deep Probabilistic Graphical Models (PGMs)
  - Undirected Deep PGMs
  - Directed Deep PGMs

Deep Neural Networks

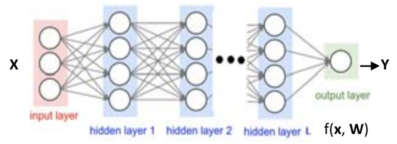
- Conventional deep neural networks (DNNs)
- Probabilistic deep neural networks
- Bayesian deep neural networks

Conventional DNNs



- $\mathbf{X} = (x_1, x_2, \ldots, x_N)^T$ represents input layer with N nodes the
- $\mathbf{Y} = (y_1, y_2, \ldots, y_K)^T$ represents output with K nodes
- $\mathbf{H}^l = (h^l_1, h^l_2, \ldots h^l_{N_l})$ represents *l* th hidden layer with N_l nodes, l=0,1,2,..,L+1. with $\mathbf{H}^0 = \mathbf{X}$ **and** $\mathbf{H}^{L+1} = \mathbf{Y}$
- $\mathbf{W}^l$ is the weight matrix (N_{l-1} x N_l) for the lth hidden layer. $\mathbf{W}^l_0$ the bias vector (N_lx1). $\mathbf{W}^{L+1}$ and $\mathbf{W}^{L+1}_0$ are the weight matrix and bias for the output layer.
- f(**x**,**W**) is the output function

Conventional DNNs



$$H^1 = \phi((\mathbf{W}^1)^T \mathbf{X} + \mathbf{W}_0^1)$$
$$H^2 = \phi((\mathbf{W}^2)^T \mathbf{H}^1 + \mathbf{W}_0^2)$$
....
$$H^L = \phi((\mathbf{W}^L)^t \mathbf{H}^{L-1} + \mathbf{W}_0^L)$$
$$f(\mathbf{X}, \mathbf{W}) = (\mathbf{W}^{L+1})^T \mathbf{H}^L + \mathbf{W}_0^{L+1}$$

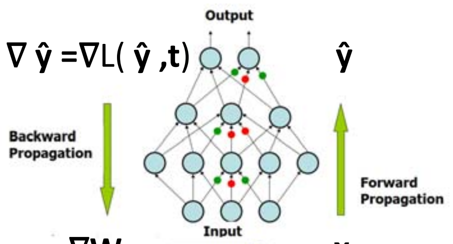
Given an input x, the output is computed through a series of recursive composition from input layer through the hidden layer until the output layer, i.e.,

ϕ() is the activation function (i.e. ReLU)
f() is the output function or discriminant function

W=($\mathbf{W}^l$, $\mathbf{W}^l_0$),l=1,2,..L+1
Regression: **y**=f(**x**, **W**)
Classification: p(**y**|**x**, **W**)=softmax (f(**x**, **W**))

Conventional DNN Training via Backprop

$\nabla \hat{\mathbf{y}} = \nabla L(\hat{\mathbf{y}}, \mathbf{t})$ $\quad \hat{\mathbf{y}}$



$\nabla \mathbf{W}$ $\quad \mathbf{x}$

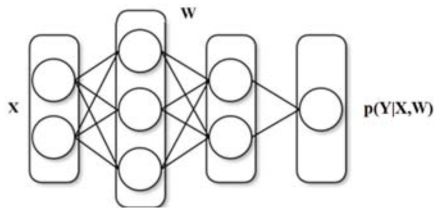
$\mathbf{W}(t) = \mathbf{W}(t-1) - \eta \nabla \mathbf{W}$

Weight Update

Limitations of Conventional DNNs

- Training is time consuming and is full of heuristics for parameter tuning.
- Training yields one set of "best" model parameters **W***, based on which output for new query data is estimated. Output prediction based only on **W** *may overfit.
- Cannot quantify the output uncertainty or confidence.

Probabilistic Deep Neural Networks (PDNNs)



Regression: $p(\mathbf{Y}|\mathbf{X},\mathbf{W}) = \mathbf{N}(u(\mathbf{x}, \mathbf{W}), \Sigma(\mathbf{X}, \mathbf{W}))$, where $u(\mathbf{x}, \mathbf{W})$ and $\Sigma(\mathbf{X}, \mathbf{W})$ are two output functions.
Classification: $p(\mathbf{Y}|\mathbf{X}, \mathbf{W}) = \mathrm{Cat}(\mathbf{y}|K, \lambda)$ $\lambda = \mathrm{softmax}(f(\mathbf{x}, \mathbf{W}))$, where $f(\mathbf{x}, \mathbf{W})$ is the output function.

PDNN Learning-Maximum Likelihood

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

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

where

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

PDNN Learning-Maximum Likelihood (cont'd)

For regression problem, the PDNN outputs $\mu(\mathbf{X},\mathbf{W})$ and $\Sigma(\mathbf{X},\mathbf{W})$ for $p(\mathbf{Y}|\mathbf{X},\mathbf{W})$

$$p(\mathbf{Y}_i|\mathbf{X}_i,\mathbf{W}) = N(\mu(\mathbf{X}_i,\mathbf{W}),\Sigma(\mathbf{X}_i,\mathbf{W}))$$
$$= \frac{1}{\sqrt{2\pi|\Sigma(\mathbf{X}_i,\mathbf{W})|}}\exp(-\frac{[\mathbf{Y}_i-\mu(\mathbf{X}_i,\mathbf{W})]^t\Sigma^{-1}(\mathbf{X}_i,\mathbf{W})[\mathbf{Y}_i-\mu(\mathbf{X}_i,\mathbf{W})]}{2})$$

$$\log p(\mathbf{Y}_i|\mathbf{X}_i,\mathbf{W}) = \frac{-1}{2}\log(2\pi|\Sigma(\mathbf{X}_i,\mathbf{W})|) - \frac{[\mathbf{Y}_i-\mu(\mathbf{X}_i,\mathbf{W})]^t\Sigma^{-1}(\mathbf{X}_i,\mathbf{W})[\mathbf{Y}_i-\mu(\mathbf{X}_i,\mathbf{W})]}{2}$$
$$\mathbf{W}^* = \arg\max_{\mathbf{W}}\sum_{i=1}^{N}\log p(\mathbf{Y}_i|\mathbf{X}_i,\mathbf{W})$$

Gradient ascent through backpropagation can be used to solve for **W***.

37

PDNN Learning-Maximum Likelihood (cont'd)

For classification problem, the DNN outputs $\lambda=\sigma_m(f(\mathbf{X},\mathbf{W}))$ for $p(\mathbf{Y}|\mathbf{X},\mathbf{W})$

$$p(Y_i|\mathbf{X}_i,\mathbf{W}) = \prod_{k=1}^{K}[\sigma_M(f(\mathbf{X}_i,\mathbf{W}))[k]]^{I(Y_i=k)}$$
$$\log p(Y_i|\mathbf{X}_i,\mathbf{W}) = \log\prod_{k=1}^{K}[\sigma_M(f(\mathbf{X}_i,\mathbf{W}_k))[k]]^{I(Y_i=k)}$$
$$= \sum_{k=1}^{K} I(Y_i=k)\log\sigma_M(f(\mathbf{X}_i,\mathbf{W}_k))[k] - \text{Cross}-\text{entropy}$$

Gradient ascent through backpropagation can be used to solve **W***.

38

PDNN Output Prediction (Inference)

Given **W*** and a query input **X**, prediction of output **Y*** can be done via

For regression

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x}, \mathbf{W}^*) = \mu(\mathbf{x}, \mathbf{W}^*)$$
$$\mathrm{Var}(\mathbf{y} \mid \mathbf{x}) = \Sigma(\mathbf{x}, \mathbf{W}^*)$$

For classification

$$k^* = \arg\max_{k} p(y = k \mid \mathbf{x}, \mathbf{W}^*) = \arg\max_{k} \sigma_M(f(\mathbf{x}, \mathbf{W}^*)[k]$$
$$\mathrm{Var}(k^* \mid \mathbf{x}) = \sigma_M(f(\mathbf{x}, \mathbf{W}^*)[k*](1 - \sigma_M(f(\mathbf{x}, \mathbf{W}^*)[k*])$$

39

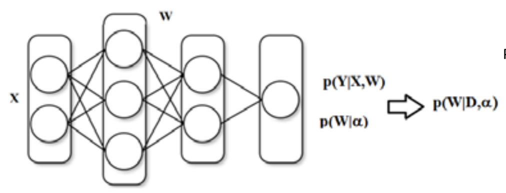
PDNN Summary

- It extends the deterministic DNN to produce output distribution p(y|x).
- It employs maximum likelihood estimation to obtain the best model parameters.
- It still involves a time-consuming training processing and can overfit.
- It can only quantify the data uncertainty and cannot quantify the model uncertainty.

40

Bayesian Deep Neural Networks (BDNNs)

- It extends the probabilistic deep models by capturing both data and model uncertainties
- Instead of estimating the model parameters, BDNN constructs the posterior distribution of the parameters p(**W** |**D**, α) and use it to perform prediction.
- BDNN consists of a likelihood model p(**y**|**x**, **W**) that relates the output **y** to the input **x,** and a prior model p(**W** |α) for the model parameters **W**



Posterior of **W** Prior of **W** Likelihood of **W**

$$p(\mathbf{W}\mid D,\boldsymbol{\alpha})=\frac{p(\mathbf{W}\mid\boldsymbol{\alpha})p(\mathbf{D}\mid\mathbf{W})}{\int p(\mathbf{W}\mid\boldsymbol{\alpha})p(\mathbf{D}\mid\mathbf{W})d\mathbf{W}}$$

BDNN Model Specification

The parameter likelihood function p(**Y**|**X**, **W**) can be specified as follows

- For regression problem
  - Let **X**∋ $\boldsymbol{R}^N$ be input vector, **Y** ∋ $\boldsymbol{R}^K$ be the output vector,

$$p(\mathbf{Y}\mid\mathbf{X},\mathbf{W})=N(\mu(\mathbf{X},\mathbf{W}),\Sigma(\mathbf{X},\mathbf{W}))$$

where m() and S() specify the mean and covariance matrix p(**Y**|**X**).

- For classification
  - Let **X**∋ $\boldsymbol{R}^N$ be the input vector and **Y** ∋ $\{1,2,\dots,K\}$ be the output vector , and σ_m() is the softmax function

$$p(Y\mid\mathbf{X},\mathbf{W})=Cat(y\mid K,\boldsymbol{\lambda}(\mathbf{X},\mathbf{W})),\qquad \boldsymbol{\lambda}(\mathbf{X},\mathbf{W})=\sigma_M(\mathrm{f}(\mathbf{X},\mathbf{W}))$$

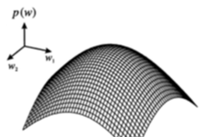
BDNN Model Specification (cont’d)

The prior probability distribution of the parameter p(**W**|α) can be specified as multi-variate Gaussian distribution, i.e.,

$$p(\mathbf{W}\mid\alpha)=N(0,\Sigma)\approx N(0,\sigma^2\mathbf{I})$$

I is a KxK identity matrix, K is dimension of **W**, and

σ^2 is a small positive number (e.g., 0.01)



BDNN Model specification (cont’d)

The posterior probability distribution of the parameter p(Θ |**D**, α) can be specified as follows

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

$$p(\mathbf{W}\mid\mathbf{D},\alpha)=\frac{p(\mathbf{W}\mid\alpha)p(\mathbf{D}\mid\mathbf{W})}{\int p(\mathbf{W}\mid\alpha)p(\mathbf{D}\mid\mathbf{W})d\mathbf{W}}$$

$$=\frac{p(\mathbf{W}\mid\alpha)\prod_i p(\mathbf{Y}_i\mid\mathbf{X}_i,\mathbf{W})}{\int p(\mathbf{W}\mid\alpha)\prod_i p(\mathbf{Y}_i\mid\mathbf{X}_i,\mathbf{W})d\mathbf{W}}$$

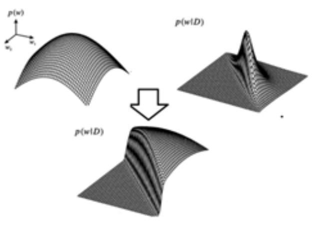
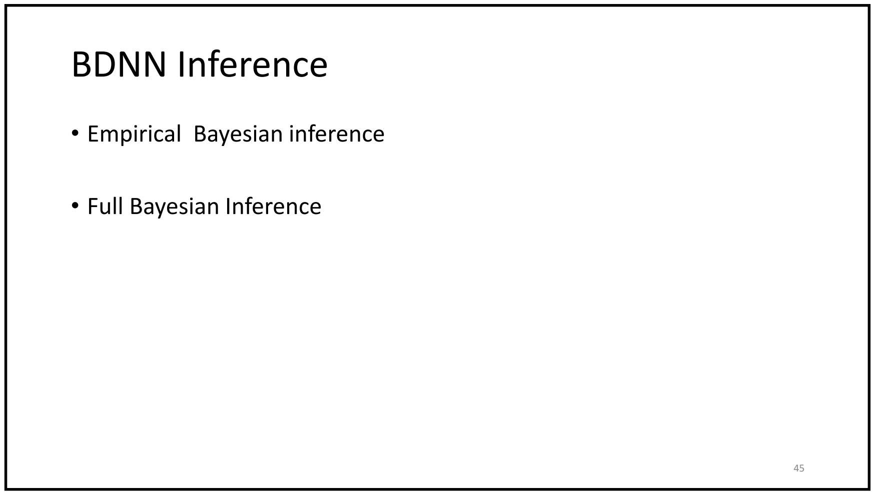
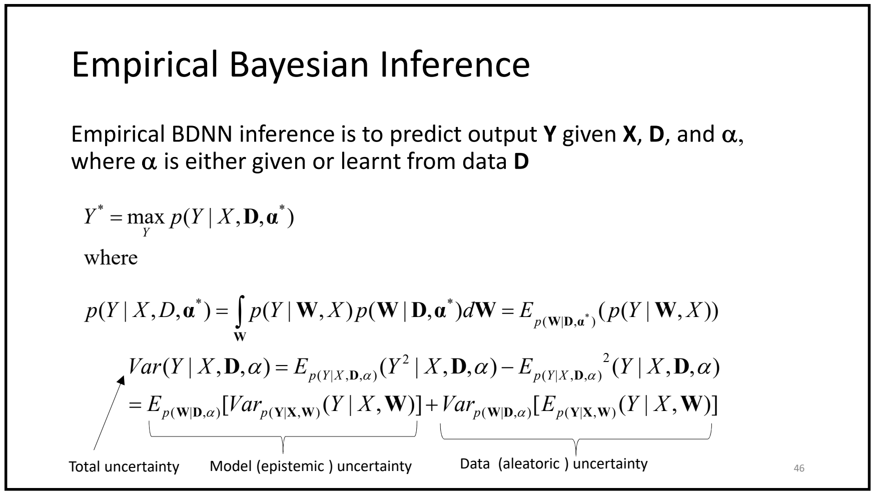


Figure credit: Aaron Courville



BDNN Inference

- Empirical Bayesian inference
- Full Bayesian Inference



Empirical Bayesian Inference

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

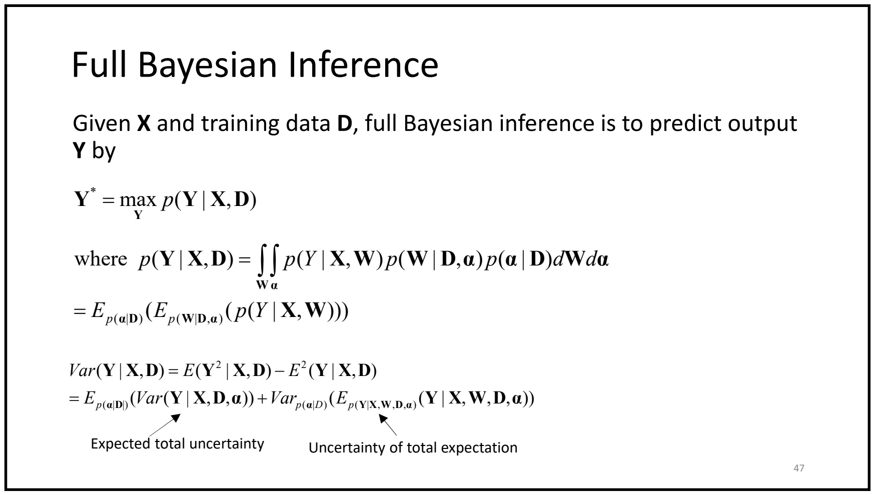
$$Y^* = \max_{Y} p(Y \mid X, \mathbf{D}, \boldsymbol{\alpha}^*)$$

where

$$p(Y \mid X, D, \boldsymbol{\alpha}^*) = \int_{\mathbf{W}} p(Y \mid \mathbf{W}, X) p(\mathbf{W} \mid \mathbf{D}, \boldsymbol{\alpha}^*) d\mathbf{W} = E_{p(\mathbf{W}|\mathbf{D},\boldsymbol{\alpha}^*)}(p(Y \mid \mathbf{W}, X))$$

$$Var(Y \mid X, \mathbf{D}, \alpha) = E_{p(Y|X,\mathbf{D},\alpha)}(Y^2 \mid X, \mathbf{D}, \alpha) - E_{p(Y|X,\mathbf{D},\alpha)}{}^2(Y \mid X, \mathbf{D}, \alpha)$$
$$= \underbrace{E_{p(\mathbf{W}|\mathbf{D},\alpha)}[Var_{p(\mathbf{Y}|\mathbf{X},\mathbf{W})}(Y \mid X, \mathbf{W})]}_{\text{Model (epistemic) uncertainty}} + \underbrace{Var_{p(\mathbf{W}|\mathbf{D},\alpha)}[E_{p(\mathbf{Y}|\mathbf{X},\mathbf{W})}(Y \mid X, \mathbf{W})]}_{\text{Data (aleatoric) uncertainty}}$$

Total uncertainty



Full Bayesian Inference

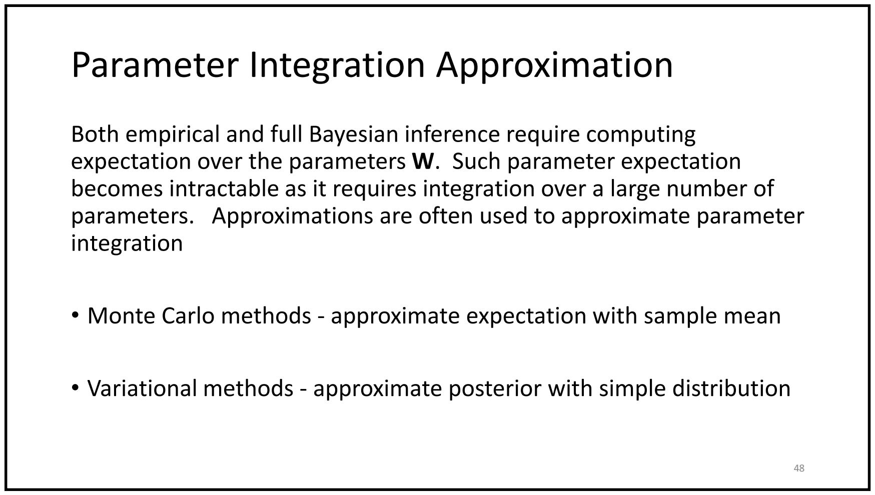
Given **X** and training data **D**, full Bayesian inference is to predict output **Y** by

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

$$\text{where } p(\mathbf{Y} \mid \mathbf{X}, \mathbf{D}) = \int_{\mathbf{W}}\int_{\boldsymbol{\alpha}} p(Y \mid \mathbf{X}, \mathbf{W}) p(\mathbf{W} \mid \mathbf{D}, \boldsymbol{\alpha}) p(\boldsymbol{\alpha} \mid \mathbf{D}) d\mathbf{W} d\boldsymbol{\alpha}$$
$$= E_{p(\boldsymbol{\alpha}|\mathbf{D})}(E_{p(\mathbf{W}|\mathbf{D},\boldsymbol{\alpha})}(p(Y \mid \mathbf{X}, \mathbf{W})))$$

$$Var(\mathbf{Y} \mid \mathbf{X}, \mathbf{D}) = E(\mathbf{Y}^2 \mid \mathbf{X}, \mathbf{D}) - E^2(\mathbf{Y} \mid \mathbf{X}, \mathbf{D})$$
$$= E_{p(\boldsymbol{\alpha}|\mathbf{D}|)}(Var(\mathbf{Y} \mid \mathbf{X}, \mathbf{D}, \boldsymbol{\alpha})) + Var_{p(\boldsymbol{\alpha}|D)}(E_{p(\mathbf{Y}|\mathbf{X},\mathbf{W},\mathbf{D},\boldsymbol{\alpha})}(\mathbf{Y} \mid \mathbf{X}, \mathbf{W}, \mathbf{D}, \boldsymbol{\alpha}))$$

Expected total uncertainty   Uncertainty of total expectation



Parameter Integration Approximation

Both empirical and full Bayesian inference require computing expectation over the parameters **W**. Such parameter expectation becomes intractable as it requires integration over a large number of parameters.   Approximations are often used to approximate parameter integration

- Monte Carlo methods - approximate expectation with sample mean
- Variational methods - approximate posterior with simple distribution

The Monte Carlo Methods

Expected value of a function: E$[f(x)] = \int p(x)f(x)dx$

can be approximated by sample averages

$$\hat{s} = \frac{1}{N}\sum_{n=1}^{N} f(x_n)$$

where x_n~ p(x)

The sample average $\hat{s}$ is an unbiased estimate of E$[f(x)]$ and as **N** approaches to infinite, the sample average $\hat{s}$ approaches E$[f(x)]$.

49

Sampling method

- Empirical Bayesian inference

$$\begin{aligned}
Y^* &= \arg\max_{Y} p(Y \mid X, \mathbf{D}, \alpha) \\
&= \arg\max_{Y} \int_{\mathbf{W}} p(Y \mid X, \mathbf{W})\, p(\mathbf{W} \mid \mathbf{D}, \alpha) \\
&\approx \arg\max_{Y} \frac{1}{N}\sum_{n=1}^{N} p(Y, \mid \mathbf{W}_n, X), \text{ where } \mathbf{W}_n \sim \text{ p}(\mathbf{W} \mid \mathbf{D}, \alpha) \\
&= \frac{1}{N}\sum_{n=1}^{N} \arg\max_{Y_n^*} p(Y_n, \mid \mathbf{W}_n, X)
\end{aligned}$$

50

Sampling method (cont’d)

- Full Bayesian Inference

$$\begin{aligned}
\mathbf{Y}^* &= \arg\max_{Y} p(\mathbf{Y} \mid \mathbf{X}, \mathbf{D}) \\
&= \arg\max_{Y} \int_{\mathbf{W}}\int_{\boldsymbol{\alpha}} p(\mathbf{Y} \mid \mathbf{X}, \mathbf{W}) p(\mathbf{W} \mid \mathbf{D}, \boldsymbol{\alpha})\, p(\boldsymbol{\alpha} \mid \mathbf{D}) d\mathbf{W} d\boldsymbol{\alpha} = E_{p(\boldsymbol{\alpha}\mid\mathbf{D})}(E_{p(\mathbf{W}\mid\mathbf{D},\boldsymbol{\alpha})}(\mathbf{Y} \mid \mathbf{X}, \mathbf{W}))) \\
&\approx \arg\max_{Y} \frac{1}{N_\alpha N_{\mathbf{W}}} \sum_{n_\alpha=1}^{N_\alpha}\sum_{n_{\mathbf{W}}=1}^{N_{\mathbf{W}}} p(Y \mid \mathbf{W}_{n_w n_\alpha}, X), \text{ where } \boldsymbol{\alpha}_{n_\alpha} \sim p(\boldsymbol{\alpha} \mid \mathbf{D}), \mathbf{W}_{n_w n_\alpha} \sim \text{ p}(\mathbf{W} \mid \mathbf{D}, \boldsymbol{\alpha}_{n_\alpha}) \\
&= \frac{1}{N_\alpha N_w} \sum_{n_\alpha=1}^{N_\alpha}\sum_{n_w=1}^{N_w} \arg\max_{Y} p(Y \mid \mathbf{W}_{n_w n_\alpha}, X)
\end{aligned}$$

51

Sampling Challenge

$$\mathbf{W}_s \sim p(\Theta \mid \mathbf{D}, \alpha) = \frac{p(\mathbf{W} \mid \alpha) \prod_i p(\mathbf{Y}_i \mid \mathbf{X}_i, \mathbf{W})}{\int p(\mathbf{W} \mid \alpha) \prod_i p(\mathbf{Y}_i \mid \mathbf{X}_i, \mathbf{W}) d\mathbf{W}}$$

- W is high dimensional, with millions or billons of parameters
- The denominator requires integration over W , which is intractable

52

Efficient Sampling Methods

- Importance sampling

- Markov Chain Monto Carlo (MCMC) Sampling
  - Gibbs sampling
  - Metropolis hastings sampling
  - Hamiltonian Monto Carlo sampling

53

Importance Sampling

Sampling from the true probability distribution p(x) can be challenging. The value we want is:

$$\hat{s} = \frac{1}{N} \sum_{n=1,\, x_n \sim p(x)}^{N} f(x_n) \approx E_{p(x)}(f(x))$$

Instead, we can sample from an alternative distribution q(x), yielding

$$\hat{s}_q = \frac{1}{N} \sum_{n=1,\, x_n \sim q(x)}^{N} \frac{p(x_n) f(x_n)}{q(x_n)} \approx E_{q(x)}(\frac{p(x) f(x)}{q(x)})$$

$\hat{s}_q$ is an unbiased estimator of $\hat{s}$, i.e. $E[\hat{s}_q] = E[\hat{s}] = \mu$. We shall choose the q() that produces the minimum variance of $\hat{s}_q$.

54

Importance Sampling

Empirical Bayesian inference

$$Y^* = \arg\max_Y \frac{1}{N} \sum_{n=1}^{N} p(Y,|\,\mathbf{W}_n, X), \text{where } \mathbf{W}_n \sim \ \mathrm{p}(\mathbf{W} \mid \mathbf{D}, \alpha)$$

$$= \arg\max_Y \frac{1}{N} \sum_{n=1}^{N} \phi(\mathbf{W}_n, X), \text{where } \mathbf{W}_n \sim \ \mathrm{q}(\mathbf{W})$$

$$\text{where} \quad \phi(\mathbf{W}) = \frac{p(Y \mid X, \mathbf{W})\, p(\mathbf{W} \mid D, \alpha)}{q(\mathbf{W})}$$

q(**W**) is chosen to minimize the variance of the sample average .

We can apply the same importance sampling to full Bayesian Inference.

Note importance sampling works well for parameters **W** of low dimension and hence is not suitable for deep neural networks.

55

Markov Chain Monte Carlo (MCMC)

- One of the most powerful methods for sampling in high dimensional parameter space.

- Starting from a random initialization, MCMC sampling follows a Markov chain to generate a sequence of samples $x^{(1)}, \ldots, x^{(n)}$ from the distribution $p(x)$, where each sample $x^{(i)}$ depends (only) on the previous sample $x^{(i-1)}$.

- Independent of starting point, as $n \rightarrow \infty$, sample distribution will approach $p(x)$.

56

Gibbs Sampling

- One of the simplest and most popular MCMC sampling methods
- It samples one parameter x_i at a time, given the current values of other parameters, i.e., sample x_i from $p(x_i|x_{-i})$ where x_{-i} are the remaining parameters .
- At each step, it needs compute $p(x_i|x_{-i})$.

57

Metropolis Hastings sampling

- Gibbs sampling assumethe availability of $p(x_i|x_{-i})$ and its efficient estimation.
- This is NOT possible for BDNNs as it requires integration over all **W**.
- An alternative MCMC method is Metropolis-Hastings (M-H) sampling method. As a rejection sampling method, M-H samples from a simpler proposal distribution q(x) to choose the next $x^{(j)}$, and then decide if to accept the sample using the unnormalized $\bar{p}$(x) (avoiding the parameter integral).

58

Metropolis Hastings Sampling

1. Start with $\mathbf{x}^0 = \{x^0_1, x^0_2, \ldots, x^0_n\}$ for n variables, whose joint distribution is p(x).
2. At iteration t, obtain a new sample of $\mathbf{x}^t$ from $q(X^t \mid x^{t-1})$, where q is a symmetric proposal distribution.
3. Calculate
$$a = \frac{\bar{p}(x^t)}{\bar{p}(x^{t-1})}, \text{ where } \bar{p}(x) \propto p(x)$$
4. Accept x^t with a probability of α. If x^t is rejected, $x^t = x^{t-1}$
5. Repeat steps 2-4 M times to get M samples, with some burn-in.

q() needs be a symmetric distribution and a Gaussian is often chosen.

59

Metropolis Hastings

Metropolis-Hastings is very powerful and widely-used.

It reduces the problem of sampling from a difficult distribution $p(x)$ to **making proposals**: $q(x^{(j)}|x^{(j-1)})$ and **evaluating ratios** of $p(x^{(j)})/p(x^{(j-1)})$.

The proposals can be trivial, e.g. random walk (choose $x^{(j)}$ from a normal distribution centered at $x^{(j-1)}$), or sophisticated. Only **efficiency, not correctness** is affected by the proposal.

Because only ratios $p(x^{(j)})/p(x^{(j-1)})$ are needed, **we don't have to deal with normalizing sums** Z.

Slide credit: Aaron Courville

60

Hamiltonian Monte-Carlo

- M-H proposals use random walks, which are slow in exploration of large parameter space.
- Hamiltonian Monte-Carlo addresses this problem by treating sampling of posterior density as a problem in Hamiltonian dynamics, by adding an independent velocity variable (***v***) to the parameter space **W**, yielding the joint probability p(**W**,**v**).
- Sampling can be performed from p(**W**,**v**) via physics simulation following Hamiltonian dynamics equations. Samples of **W** can be obtained by ignoring samples of **v**.
- Converge faster than M-H method but requires computing gradient each step.

Sampling with MC Dropout

- Dropout is a regularization technique for conventional deep neural networks, where it follows the Bernoulli distribution to decide which link to keep or drop.
- Different DNNS can be obtained by *using dropout for every layer during training as well as testing, and it has been shown that they are* mathematically equivalent to samples from a BDNNs.
- *Pros*:
  - It is easy to turn an existing deep net into a Bayesian one. it is faster than other techniques, and does not require an inference framework.
- *Cons:*
  - Of course, sampling at test time might be too expensive for computationally-demanding (eg real time) applications.

Sampling Methods Summary

- The goal is to approximate the prediction expectation with sample average .
- Importance Sampling is used to estimate means with fewer parameters.
- Monte-Carlo sampling requires either:
  - Closed form local distributions (Gibbs sampling)
  - A simple proposal distribution to generate next samples and use an unnormalized posterior to accept/reject samples.
- Hamiltonian Monte Carlo (HMC) adds a momentum term to allow samples generations based on physics equations.

Variational methods

- Variational inference methods approximate the parameter integration by approximating the posterior parameter distribution p(**W**|**D**,**a**) with a simple and factorized distribution q(**W**|**D**,**b**).

$$p(\mathbf{W} \mid \mathbf{D}, \alpha) \approx q(\mathbf{W} \mid \mathbf{D}, \boldsymbol{\beta})$$

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}} KL(q(\mathbf{W} \mid \mathbf{D}, \boldsymbol{\beta}) \parallel p(\mathbf{W} \mid \mathbf{D}, \boldsymbol{\alpha}))$$

where b are the variational parameters.

- With a factorized distribution, the integration can be performed more easily.

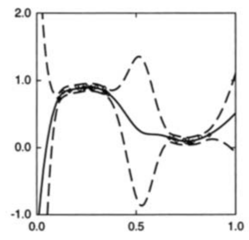
Variational Methods

- Design choice of q(**W**|D)
- Often choose to be simple for tractable inference e.g. mean-field variational inference assumes fully factorized q(**W**|**D**)= $\prod_i \boldsymbol{p}(W_i|\boldsymbol{D})$
  - 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]
- Pros: Fast and easy to implement
- Cons: Inaccurate due to the inherent gap between q() and p(), leading to non-optimal solution. Requiring an optimization procedure to estimate q() that varies with initialization and could be slow for large model.

65

Example o Bayesian Regression

- The figure is an example of the application of Bayesian methods to a regression problem. The data(circles) was generated from the function, h(x) = 0.5 + 0.4sin(2πx)



Bayesian Methods for Neural Networks – p.17/28

66

Example of Bayesian Classification

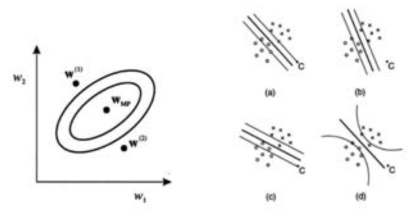


Figure 1 Figure 2

The three lines in Figure 2 correspond to network outputs of 0.1, 0.5, and 0.9. (a) shows the predictions made by $\mathbf{w}_{MP}$. (b) and (c) show the predictions made by the weights $\mathbf{w}^{(1)}$ and $\mathbf{w}^{(2)}$. (d) shows $P(\mathcal{C}_1|\mathbf{x}, \mathcal{D})$, the prediction after marginalizing over the distribution of weights; for point C, far from the training data, the output is close to 0.5.

67

Bayesian Model Selection

- BNN can be used to select best NN models (M) for a given training data **D** in terms of its architecture (number of hidden layers, number of filters, types of activation functions, etc..) using model likelihood, i.e.,

$$p(\mathbf{D}\,|\,M) = \int_{\mathbf{W}} \underbrace{p(\mathbf{D}\,|\,\mathbf{W}, M)}_{\text{Model likelihood}} \underbrace{p(\mathbf{W}\,|\,M)}_{\text{model prior}} d\mathbf{W}$$

$$M^* = \arg\max_M p(\mathbf{D}\,|\,M)$$

The prior term balances between data likelihood term to avoid overly complex models.

68

Model Selection and fusion for Prediction

- **Model selection**
  - Given K models M_k , k=1,2,..K, training data **D**, and an input X, identify the model that produces the output Y with minimum variance, i.e.,

$$M_{k^*} = \arg\min_k Var(Y \mid X, M_k, \mathbf{D})$$
$$Var(Y \mid X, M_k, \mathbf{D}) = \int (y - \mu_y)^2 p(y \mid X, M_k, \mathbf{D}) dy$$
$$y^* = \arg\max_y p(y \mid X, \mathbf{D}, M_{k^*})$$

- We can also combine the results from the K models, i.e.

$$p(Y \mid X, D) = \sum_{i=1}^{M} p(Y \mid X, D, M_i) p(M_i \mid D)$$
$$y^* = \arg\max_y p(y \mid X, D)$$

The result is a weighted average of the probability distributions over the outputs of the models.

69

Bayesian Neural Network Software

The major sampling and variational methods have been implemented in the following software

- TensorFlow probability
  - https://www.tensorflow.org/probability
- Edward
  - http://edwardlib.org/
- Pyro
  - http://docs.pyro.ai/en/0.3.0-release/contrib.bnn.html
- Gen-A General-Purpose Probabilistic Programming language
  - https://dspace.mit.edu/bitstream/handle/1721.1/119255/MIT-CSAIL-TR-2018-020.pdf

70

Sampling and Variational Metod Implementations

Method/Language	Pyro	Tensorflow Probability	Edward	Gen
Metropolis Hastings	Yes	Yes	Yes	Yes
Hamiltonian Monte Carlo	Yes	Yes	Yes	Yes
Variational Inference	Yes	Yes	Yes	Yes

71

Bayesian Neural Networks

- Bayesian Neural Network

https://medium.com/neuralspace/bayesian-neural-network-series-post-1-need-for-bayesian-networks-e209e66b70b2

- 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

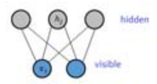
- Tensorflow probability

https://medium.com/tensorflow/introducing-tensorflow-probability-dca4c304e245

72

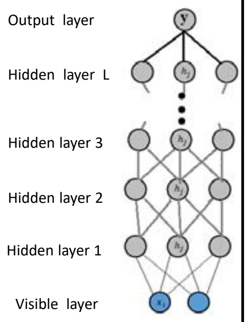
Deep Probabilistic Graphical Models

- Constructed from Probabilistic Graphical models (PGMs) instead of from neural networks.
- PGMs are probabilistic models that capture the joint probabilistic distribution of the data
- The building block of a deep PGM is a two-layer graphical model, with one latent layer **h** and a visible data layer ***x***.



*p(**x**,**h**).*

- A deep PGM can be constructed by stacking the building block on top of each other, capturing *p(**h**^{1}, **h**^{2},..., **h**^{L},**x**, **y**).*

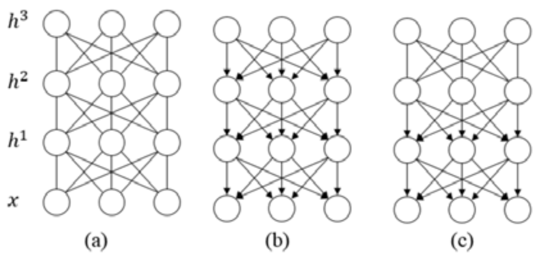


*p(**h**^{1}, **h**^{2},..., **h**^{L},**x**, **y**).*

73

Different Types of Deep PGMs

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



Deep Boltzmann Machine (undirected)

Deep Regression Bayesian Network (directed)

Deep Belief Network (hybrid)

74

Probabilistic Graphical Model (PGM)

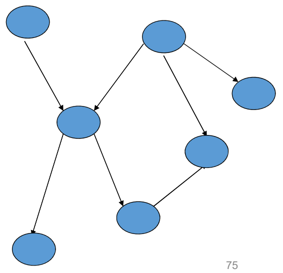
A PGM is a **graphical** model for **compactly** representing the **joint** probabilistic distributions among **random variables**.

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**):



75

Deep Undirected PGMs

- Markov Network
- Restricted Boltzmann Machine
  - Learning and inference with DBM
  - DBM applications
- Deep Boltzmann Machine
  - Learning and inference with DBM
  - DBM applications
- DBM Applications

76

Markov Networks

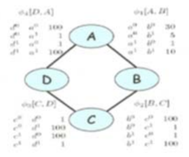
- A Markov Network (MN) *G* is an undirected graph that models the probabilistic dependencies of among a set of random variables.
- Nodes represent RVs and links represent their mutual dependencies or correlation
- A MN concisely encodes the joint probability of all nodes.

$$p(\mathbf{X}) = \frac{1}{Z}\prod_{c\in\mathbf{C}}\Phi(\mathbf{X}_c),\quad \mathbf{X}=\{X_n\}_{n=1}^{N}$$

c – a maximum clique, $\mathbf{X}_c$ varables in c,

and $\Phi()$ potential function, and Z partition function.

$$p(A,B,C,D)=\frac{\phi(A,B)\ \phi(B,C)\ \phi(C,D)\phi(A,D)}{Z}$$



- Markov condition: given its neighbors, a node is independent of all other nodes.

A and C are independent given D and B

Pairwise Markov Network

- A MN with maximum clique size of 2.
- Each node is parameterized by a unary potential function f(X_i) and a pairwise potential function y(X_i,X_j), with its neighbor $X_j \in N_{xi.}$ With a log-linear potential function,

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

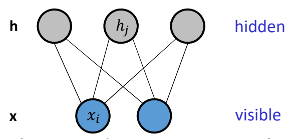
- where E(X_i) and E(X_i,X_j) are the unary and pairwise energy functions.
- The joint probability of all nodes **X**={X_1,X_2,…, X_N} can be represented by the Gibbs distribution

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

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

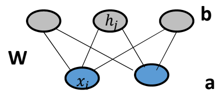
Restricted Boltzmann machines (RBM)

- A pairwise MN with a latent layer **h** of binary variables {1,0} and a 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



- According to the Markov condition, given **x**, elements of **h** are independent of each other, *and vice versa.*

Binary RBM



- For binary $x_i \in \{1,0\}$, its energy function of for **h** and **x** is,

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

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

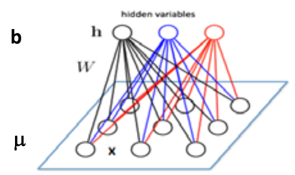
Sigmoid function

$$p(\mathbf{x},\boldsymbol{h}|\Theta) = \frac{1}{Z_\theta}\exp(-E(\mathbf{x},\boldsymbol{h},\Theta)) \quad p(h_j=1\,|\,\mathbf{x}) = \sigma(b_j + \mathbf{x}^t\mathbf{W}_j) \quad p(x_i=1\,|\,\mathbf{h}) = \sigma(a_i + \mathbf{W}_i\mathbf{h})$$

- $\boldsymbol{\Theta} = \{\boldsymbol{W},\boldsymbol{a},\boldsymbol{b}\}$, where $\mathbf{W}^{NxH}$={w_{ij}}, $\mathbf{a}^{Nx1}$={a_i}, and $\mathbf{b}^{Hx1}$={b_j} 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 $Z_\theta=\sum_{x\in\boldsymbol{x}}\sum_{h\in\boldsymbol{h}}\exp(-E(\mathbf{x},\boldsymbol{h}|\Theta))$

Continuous RBMs

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



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

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

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

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

$$p(h_j=1\,|\,\mathbf{x}) = \sigma(b_j + \sum_i w_{ij}\frac{x_i}{\sigma_i})$$

$$p(x_i\,|\,\mathbf{h}) = N(\mu_i + \sum_j h_j w_{ij}, \sigma_i^2)$$

81

RBM Parameter Learning

Given dataset **D**=$\{x(m)\}_{m=1}^{M}$, the parameter learning task is to estimate the parameters Θ={**W**, **a**, **b**} by maximizing the average log marginal likelihood,

$$\Theta^* = \arg\max_{\Theta}\frac{1}{M}\sum_{m=1}^{M}\log p(\mathbf{x}(m)\,|\,\Theta)$$

$$\text{where}\log p(\mathbf{x}(m)\,|\,\Theta) = \log\sum_{\mathbf{h}} p(\mathbf{x}(m),\mathbf{h}\,|\,\Theta) = \log\frac{1}{Z}\sum_{\mathbf{h}}\exp(-E(\mathbf{x}(m),\mathbf{h}\,|\,\Theta))$$

$$\Theta(t) = \Theta(t-1) + \eta\nabla_{\Theta}(\frac{1}{M}\sum_{m=1}^{M}\log p(\mathbf{x}(m)\,|\,\Theta))$$

$$\nabla_{\Theta}(\frac{1}{M}\sum_{m=1}^{M}\log p(\mathbf{x}(m)\,|\,\Theta)) = -\frac{1}{M}\sum_{m=1}^{M}\sum_{\mathbf{h}} p(\mathbf{h}\,|\,\mathbf{x}(m),\Theta)\frac{\partial E(\mathbf{x}(m),\mathbf{h}\,|\,\Theta)}{\partial\Theta} + \sum_{\mathbf{x},\mathbf{h}} p(\mathbf{x},\mathbf{h})\frac{\partial E(\mathbf{x},\mathbf{h}\,|\,\Theta)}{\partial\Theta}$$

$$\nabla_{W}(\frac{1}{M}\sum_{m=1}^{M}\log p(\mathbf{x}(m)\,|\,\Theta)) = -\frac{1}{M}\sum_{m=1}^{M}\sum_{\mathbf{h}} p(\mathbf{h}\,|\,\mathbf{x}(m),\Theta)\mathbf{x}(m)\mathbf{h}^t + \sum_{\mathbf{x},\mathbf{h}} p(\mathbf{x},\mathbf{h})\mathbf{x}\mathbf{h}^t$$

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \eta\nabla_{W}(\frac{1}{M}\sum_{m=1}^{M}\log p(\mathbf{x}(m)\,|\,\Theta))$$

where the first and second terms are data and model expectation (averages), respectively. The second term comes from taking derivative of the log partition function Z. It can be estimated computed by Gibbs sampling p(**x**,**h**) using current parameters - contrastive divergence (CD) method. Both terms become computationally intractable for a large number of hidden nodes.

82

Inference in RBMs

- Posterior probability inference p(**h**|**x**) or p(**x**|**h**)
- MAP inference **h***= argmax$_{\mathbf{h}}$ p(**h**|**x**) or **x***= argmax$_{\mathbf{x}}$ p(**x**|**h**)
- Likelihood inference p(**x**)

83

Inference in RBMs

- Posterior probability inference P(**h**|**x**)

$$p(\mathbf{h}\,|\,\mathbf{x}) = \prod_{j=1}^{H} p(h_j\,|\,\mathbf{x}) = \prod_{j=1}^{H}\sigma(b_j + \mathbf{x}^t\mathbf{W}_j)^{h_j}(1-\sigma(b_j+\mathbf{x}^t\mathbf{W}_j))^{1-h_j}$$

where $\mathbf{W}_j$ is jth column of $\mathbf{W}^{N\text{x}H}$, $\sigma()$ is the sigmoid function.

- Posterior probability inference P(**x**|**h**) for binary **x**

$$p(\mathbf{x}\,|\,\mathbf{h}) = \prod_{i=1}^{N} p(x_i\,|\,\mathbf{h}) = \prod_{i=1}^{N}\sigma(a_i + \mathbf{W}_i\mathbf{h})^{x_i}(1-\sigma(a_i+\mathbf{W}_i\mathbf{h}))^{1-x_i}$$

where $\mathbf{W}_i$ is the ith row of $\mathbf{W}$.

84

Inference in RBMs

- Posterior probability inference p(**x**|**h**) for continuous **x**

$$p(\mathbf{x}\,|\,\mathbf{h}) = \prod_{i=1}^{N} p(x_i\,|\,\mathbf{h})$$

$$= \prod_{i=1}^{N} N(\mu_i + \sum_j W_{ij} h_j, \sigma_i^2)$$

85

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$_i$.

$$h_j^* = \arg\max_{h_j} P(h_j|\boldsymbol{x}) = \begin{Bmatrix} 1 \;\; if\ \sigma(b_j + \boldsymbol{x}^t \boldsymbol{W}_j) > 0.5 \\ 0 \qquad\qquad\qquad\qquad else \end{Bmatrix}$$

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

  For binary **x**

$$x_i^* = \arg\max_{x_i} P(x_i|\boldsymbol{h}) = \begin{Bmatrix} 1 \;\; if\ \sigma(a_i + \boldsymbol{W}_i \boldsymbol{h}) > 0.5 \\ 0 \qquad\qquad\qquad\qquad else \end{Bmatrix}$$

  For continuous **x**

$$x_i^* = \arg\max_{x_i} P(x_i|\boldsymbol{h}) = \mu_i$$

86

Inference in RBMs

- Likelihood inference p(**x**)

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

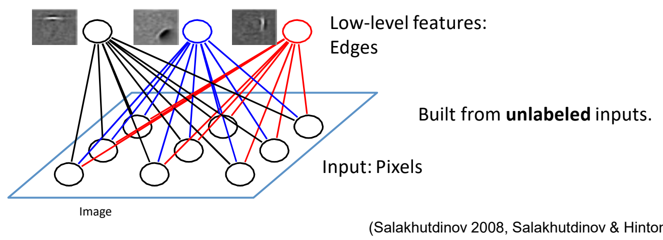
If the number of **h** is too large to enumerate, approximate method such as importance sampling can apply

$$p(\mathbf{x}) = \sum_{\mathbf{h}} p(\mathbf{x},\mathbf{h}) = \sum_{\mathbf{h}} p(\mathbf{h}\,|\,\mathbf{x}) \frac{p(\mathbf{x},\mathbf{h})}{p(\mathbf{h}\,|\,\mathbf{x})} \approx \frac{1}{N} \sum_{n=1}^{N} \frac{p(\mathbf{x},\mathbf{h}_n)}{q(\mathbf{h}_n\,|\,\mathbf{x})}, \mathbf{h}_n \sim p(\mathbf{h}\,|\,\mathbf{x})$$

87

Deep Boltzmann Machines

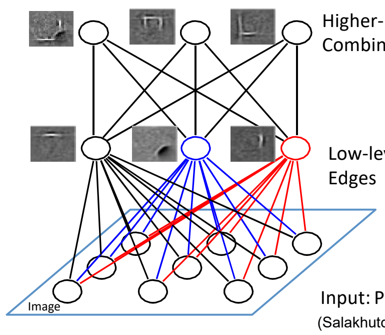
RBMs can be used as building blocks to construct the Deep Boltzmann Machines (DBMs).



(Salakhutdinov 2008, Salakhutdinov & Hinton 2012)

88

Deep Boltzmann Machines



Higher-level features:
Combination of edges

Low-level features:
Edges

Built from **unlabeled** inputs.

Input: Pixels

(Salakhutdinov 2008, Salakhutdinov & Hinton 2012)

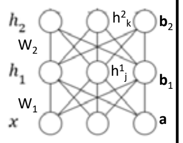
DBM Parameterizations

- The joint probabilities are defined using pairwise and unary potentials:

$$E(\boldsymbol{x}, \boldsymbol{h_1}, \boldsymbol{h}_2) = -\boldsymbol{x}^t \boldsymbol{W}_1 \boldsymbol{h}_1 - \boldsymbol{h}_1^t \boldsymbol{W}_2 \boldsymbol{h}_2 - \boldsymbol{a}^t \boldsymbol{x} - \boldsymbol{b}_1^t \boldsymbol{h}_1 - \boldsymbol{b}_2^t \boldsymbol{h}_2$$

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

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



Deep Boltzmann Macbhine

$$p(h_{1j} = 1 | \mathbf{x}, \mathbf{h}_2) = \sigma(b_{1,j} + \mathbf{x}^t \mathbf{W}_{1,j} + \mathbf{W}_{2,j} \mathbf{h}_2)$$

where $\mathbf{W}_{1,j}$ is jth column of $\mathbf{W}_1$, and $\mathbf{W}_{2,j}$ is jth row of $\mathbf{W}_2$

$$p(h_{2,k} = 1 | \mathbf{h}_1) = \sigma(b_{2,k} + \mathbf{h}_1^t \mathbf{W}_{2,k})$$

where $\mathbf{W}_{2,k}$ is kth column of $\mathbf{W}_2$

Note: 1) $\mathbf{h}_{1,j}$ are no longer independent given **x** and hence p($\mathbf{h}_1$|**x**) can not be factorized as product of p($\mathbf{h}_{1,j}$|**x**).
2) Given $\mathbf{h}_1$, elements of **x** are independent like RBM but given $\mathbf{h}_2$, they are not independent.

DBM Learning

Given the training data **D**,={**x**(m)}, m=1,2,..,M, DBM learning is to learn the parameters Q, weights and bias for all layers.

- Pre-training
  - Layerwise unsupervised learning
- Fine-tuning
  - Unsupervised fine-tuning
  - Supervised fine-tuning

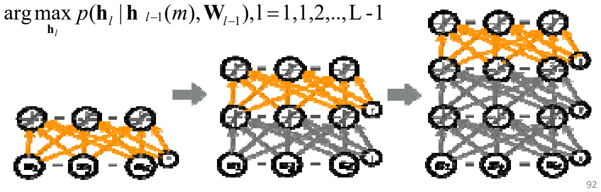
DBM Pre-training

- Learn consecutive layers separately as RBM, while fixing the values for last layer
- Train one layer at a time by maximizing its input log likelihood using CD method
- Learn the parameters for next layer using the output of the previous layer as input

$$\mathbf{W}_l^* = \arg\max_{\mathbf{W}_l} \sum_m \log p(\mathbf{h}_l^*(m) | \mathbf{W}_l)$$

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

Note max p($\mathbf{h}_l$ | $\mathbf{h}_{l-1}$) is difficult as $\mathbf{h}_l$ are no longer independent.



DBM Unsupervised Fine Tuning

Use the pre-training results as initialization and update all parameters at the same time by maximizing the log likelihood of the input data **x**.

$$\Theta^* = \arg\max_{\mathbf{W}} \frac{1}{M}\sum_m \log p(\mathbf{x}(m)\,|\,\Theta) = \arg\max_{\mathbf{W}} \frac{1}{M}\sum_m \log \sum_{h_1,h_2,\ldots,h_L} p(h_1,h_2,\ldots,h_L,\mathbf{x}(m)\,|\,\Theta), \quad \text{where } \Theta=(W_1,W_2,\ldots,W_L)$$

$$\Theta_l^{t+1} = \Theta_l^t + \eta_l \nabla\Theta_l \quad \text{for } \mathrm{l}=1,2,..,\mathrm{L}$$

$$\nabla\Theta_l = \frac{\partial \frac{1}{M}\sum_m \log p(\mathbf{x}(m)\,|\,\Theta)}{\partial \mathbf{W}_l} = -\frac{1}{M}\sum_m \sum_{h^1,h^2,\ldots,h^L} p(h_1,h_2,\ldots,h_L\,|\,\mathbf{x}(m))\frac{\partial E(h_1,h_2,\ldots,h_L,\mathbf{x}(m))}{\partial\Theta_l} + \sum_{\mathbf{x},h^1,h^2,\ldots,h^L} p(h_1,h_2,\ldots,h_L,\mathbf{x})\frac{\partial E(h_1,h_2,\ldots,h_L,\mathbf{x})}{\partial\Theta_l}$$

Challenges and solutions :

1) Exponential number of hidden layer configurations

Solution : Stochastic sampling $\mathrm{p}(\mathbf{h}_1,\mathbf{h}_2,\ldots\mathbf{h}_L,\mathbf{x})$ and use sample average to to approximate the summation.

2) $\mathrm{p}(\mathbf{h}_1,\mathbf{h}_2,\ldots\mathbf{h}_L\,|\,\mathbf{x})$ are no factoriable :

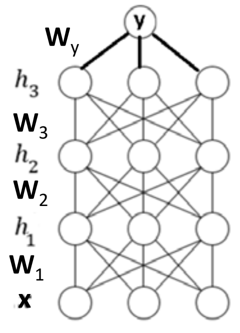
Solution : Variational approximation for $\mathrm{p}(\mathbf{h}_1,\mathbf{h}_2,\ldots\mathbf{h}_L\,|\,\mathbf{x})$ by $\mathrm{q}(\mathbf{h}_1,\mathbf{h}_2,\ldots\mathbf{h}_L\,|\,\mathbf{x})$

93

DBM Supervised Fine-tuning

Once all layers are pre-trained

- Add output layer **y**
- train the whole network using supervised learning by maximizing p(**x**,**y**) (generative learning) or p(**y**|**x**) (discriminative learning)



$$p(\mathbf{y},\mathbf{x}\,|\,\Theta) = \sum_{\mathbf{h}_1,\mathbf{h}_2,\mathbf{h}_3} p(\mathbf{y},\mathbf{x},\mathbf{h}_1,\mathbf{h}_2,\mathbf{h}_3\,|\,\Theta)$$

where

$$p(\mathbf{y},\mathbf{x},\mathbf{h}_1,\mathbf{h}_2,\mathbf{h}_3\,|\,\Theta) = \frac{1}{Z}\exp(\mathbf{x}^t\mathbf{W}_1\mathbf{h}_1 + \mathbf{h}_1^t\mathbf{W}_2\mathbf{h}_2 + \mathbf{h}_2^t\,\mathbf{W}_3\mathbf{h}_3 + \mathbf{h}_3^t\mathbf{W}_y\mathbf{y})$$

Note the bias terms are ignored.

For l = 1 to L

$$\mathbf{W}_l(t) = \mathbf{W}_l(t-1) + \eta\frac{\partial p(\mathbf{y},\mathbf{x},\mathbf{h}_1,\mathbf{h}_2,\mathbf{h}_3\,|\,\Theta)}{\partial \mathbf{W}_l}$$

94

Encounter the same computational challenges and same solutions can be employed.

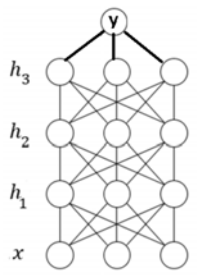
DBM inference

1. Posterior probability inferences

   p(h$_{l,j}$|**x**) and p(**x**$_i$|h$_L$) -both no longer factorize (see fig) .
   - Variational method

     Mean filed p(h$_l$|**x**) ≈q(h$_l$|**x**)=$\prod_{j=1}^{J} q($ h$_{l,j}$) or inference network.
   - Gibbs sampling -obtain samples from p(h$_l$|**x**) and use samples to approximate

     the posterior probability
   - Same approximations can apply to p(**x**$_i$|h$_L$)
2. MAP inferences
   - **h**$_l$*=argmax$_h$ p(**h**$_l$|**x**)
     - Coordinate ascent
     - Variational approximation and perform inference with q(**h**$_l$|x), which is factorable.
   - x*=argmax$_x$ p(**x**|**h**$_l$)
     - Employ the same approximations

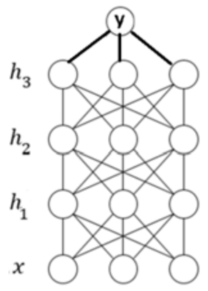


95

DBM inference

3. Likelihood inference
   - $p(\boldsymbol{x}) = \sum_{\boldsymbol{h_1},\, \boldsymbol{h_2}.\, \boldsymbol{h_3}} p(\boldsymbol{x}, \boldsymbol{h_1}, \boldsymbol{h_2}.\, \boldsymbol{h_3})$
     - Use sampling average to approximate the summation.

4. Classification inference
   - **y***=argmax$_y$ p(**y**|**x**)=argmax$_y$ $\Sigma_{\mathrm{h1,h2,\,..,hL}}$ p(**h**$_1$,**h**$_2$, ..,**h**$_L$, **y**|**x**)
     - Employ sampling or variational method to approximate p(**y**|**x**)



Inference challenges with DBM lead to typically only a few hidden layers and limited number of hidden nodes in each layer.

96

Deep Boltzmann Machine Summary

Training	Inference	Structure	Advantages
A layerwise pre-training and fine-tune step, both are implemented in gradient ascent.	Inference includes posterior inference, MAP inference, likelihood inference, and classification inference.	Limited to a few (3) hidden layers	Fully capture the joint probability distribution of the data
Both pre-training and fine tuning include 1) summation over an exponential number of latent layer configurations, and 2). intractable posterior probability inference.	Because of dependences among latent nodes, all inferences are intractable. Approximation inference methods including MCMC sampling and variational methods.	Computationally challenging during both learning and inference for many layers.	

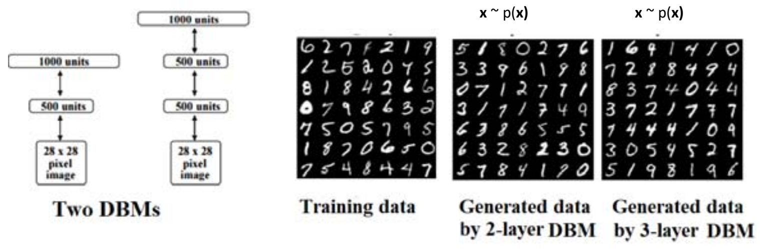
DBM Applications

- Generative data modeling $\mathbf{x} \sim p(\mathbf{x})$
  - Single modality data modeling
  - Multi-modal data modeling
- Feature/representation learning $\mathbf{h} \sim \text{argmax}_h\, p(\mathbf{h}|\mathbf{x})$
- Classification $y^*=\text{argmax}_y\, p(\mathbf{y}|\mathbf{x})$

Generative Model of Handwritten Digits

Generate **x** by sampling p(**x**)

- Employ Gibbs sampling of $p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2, h_3)$ to obtain samples **x** and ignore samples of **h**s.



Generative Model of 3-D Objects

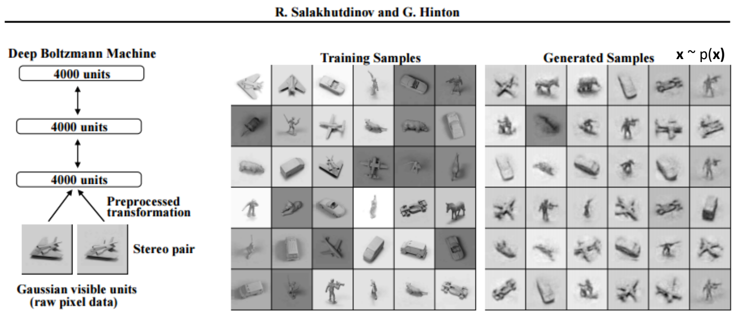
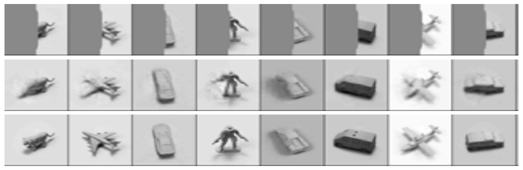


Figure 5: **Left:** The architecture of deep Boltzmann machine used for NORB. **Right:** Random samples from the training set, and samples generated from the deep Boltzmann machines by running the Gibbs sampler for 10,000 steps.

Generate **x** by sampling p(**x**)

Pattern Completion

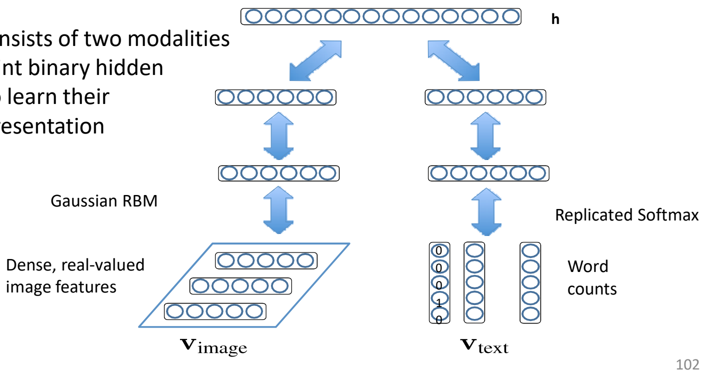
Given an occluded input $\hat{x}$, infer the original pattern **x** by
$\mathbf{h}_L^* = \text{argmax } p(\mathbf{h}_L | \hat{x})$
$\mathbf{x}^* = \text{argmax } p(\mathbf{x} | \mathbf{h}_L^*)$


Occluded pattern ($\hat{x}$)
Recovered pattern ($\mathbf{x}^*$)
Original pattern ($\mathbf{x}$)

101

Multimodal DBM for Image and Text

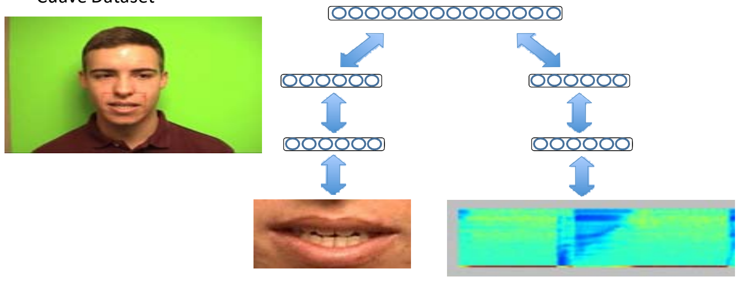
- Input consists of two modalities
- Use a joint binary hidden layer **h** to learn their joint representation


h
Gaussian RBM
Replicated Softmax
Dense, real-valued image features
Word counts
$\mathbf{v}_{\text{image}}$
$\mathbf{v}_{\text{text}}$

102

Multimodal DBM for Video and Audio

Cuave Dataset

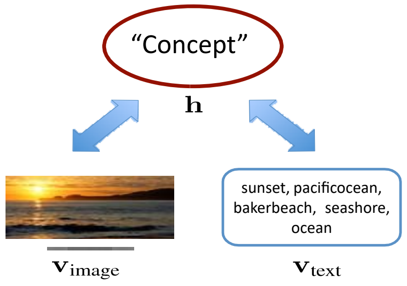


103

Multimodal DBM for Classification

- Learn a joint density model: $P(\mathbf{h}, \mathbf{v}_{\text{image}}, \mathbf{v}_{\text{text}})$.
- **h**: "fused" representation for classification, retrieval.

$P(\mathbf{h} | \mathbf{v}_{\text{image}}, \mathbf{v}_{\text{text}})$


"Concept"
h
sunset, pacificocean, bakerbeach, seashore, ocean
$\mathbf{v}_{\text{image}}$
$\mathbf{v}_{\text{text}}$

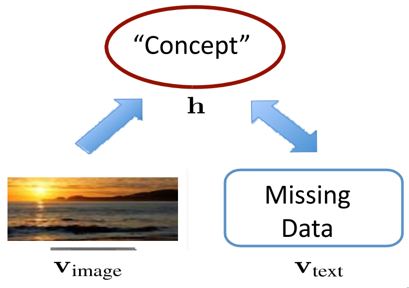
104

Multimodal DBM for Text Generation

Given the multimodal DBM, infer the missing modality given one modality, i.e.,

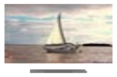
$$\mathbf{v}^*_{text} = \arg\max_{\mathbf{v}_{text}} p(\mathbf{v}_{text} \mid \mathbf{v}_{image})$$

for image annotation



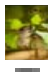
105

Text Generated from Images

Given	Generated	Given	Generated
	dog, cat, pet, kitten, puppy, ginger, tongue, kitty, dogs, furry		insect, butterfly, insects, bug, butterflies, lepidoptera
	sea, france, boat, mer, beach, river, bretagne, plage, brittany		graffiti, streetart, stencil, sticker, urbanart, graff, sanfrancisco
	portrait, child, kid, ritratto, kids, children, boy, cute, boys, italy		canada, nature, sunrise, ontario, fog, mist, bc, morning

106

Images from Text

Given	Retrieved			
water, red, sunset				
nature, flower, red, green				
blue, green, yellow, colors				
chocolate, cake				

107

DBM for Classification

y*=argmax p(**y**|**x**)

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%**

3-D Object Recognition

Learning Algorithm	Error
Logistic regression	22.5%
K-NN (LeCun 2004)	18.92%
SVM (Bengio & LeCun 2007)	11.6%
Deep Belief Net (Nair & Hinton 2009)	9.0%
DBM	**7.2%**

108

Reading materials

- R. Salakhutdinov and G. Hinton, “Deep Boltzmann Machines,” AISTATS 2009
- Deep learning — Deep Boltzmann Machine (DBM) , https://medium.com/datadriveninvestor/deep-learning-deep-boltzmann-machine-dbm-e3253bb95d0f

.

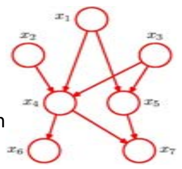
Deep Directed PGMs

- Bayesian Networks (BN)
- Regression Bayesian Network (RBN)
- Deep Regression Bayesian Network (DRBN)
- Deep Belief Networks (DBN)
- DRBN and DBN Applications

Bayesian Networks (BN)

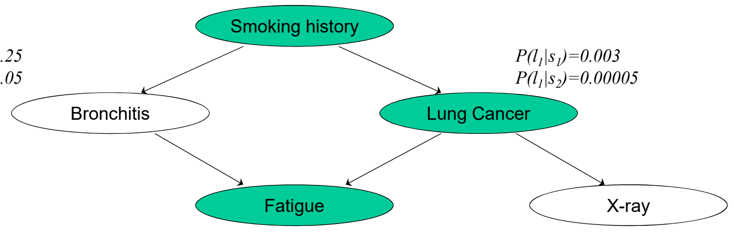
A Bayesian Network is a Directed Acyclic Graph (DAG) that models the dependences among a set of random variables. It consists of:

- the qualitative part, i.e. the graph G=(V,E)
  - Nodes (V)-random variables
  - Edges (E) –capture directed/causal relations
  - No directed cycles
  - Markov condition
    - a node is independent of its non-descendants given its paren
- the quantitative part
  - Conditional probability for each node $p(x_i|p(x_i))$
  - The joint probability



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

An Example Bayesian Network



$P(s_1)=0.2$

$P(b_1|s_1)=0.25$
$P(b_1|s_2)=0.05$

$P(l_1|s_1)=0.003$
$P(l_1|s_2)=0.00005$

$P(f_1|b_1,l_1)=0.75$
$P(f_1|b_1,l_2)=0.10$
$P(f1|b_2,l_1)=0.5$
$P(f1|b_2,l_2)=0.05$

$P(x_1|l_1)=0.6$
$P(x_1|l_2)=0.02$

Total # of parameters is 11 instead of 31

$p(S,B,F,L,X)=p(S)p(B|S)p(L|S)p(F|B,L)p(X|L)$

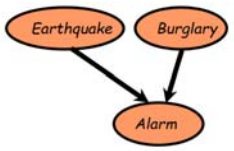
R.E. Neapolitan, Learning Bayesian Networks (2004)

Issues in BN

- Learning
  - Learn the parameters (CPD) and the structure of the BN from training data.
- Inference
  - Given the observation of some nodes, infer the probability or values of other nodes.

Explain-Away

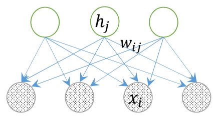
- Two random variables are marginally independent but become dependent given their common child – the V structure .



- This is a unique property for BN. It increases BN's representation power but also increases its inference complexity.

Regression Bayesian Network

- Regression Bayesian networks (RBNs):
  - Two layer BN-latent layer (**h**) and visible layer (**x**)
  - Latent variables are binary, are the parents of visible variables, and are connected to all visible variables.
  - No connections among nodes in the same layer
  - Each link is associated with a weight parameter w_{ij}
  - Given **x**, **h**s are no longer independent of each other because of explain-away (different from RBM!) . But given **h**, **x**s are still independent of each other .
- RBN is the building block for deep directed model.

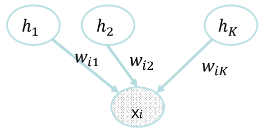


Regression Bayesian Network

Binary Regression Bayesian Network (Sigmoid Belief Network)

Parameterization of RBNs with binary input variable x_i ∈{+1,0}

Prior of h_j	$p(h_j = 1) = \sigma(b_j)$
Conditional prob of x_i	$p(x_i = 1 \mid \mathbf{h}) = \sigma(a_i + W_i \mathbf{h})$



where s is a sigmoid function and $\mathbf{W}_i$ is the ith row of **W** matrix. Each visible node x_i is a linear combination of its parents **h**, with its probability compactly represented by $\mathbf{W}_i$ and a_i.

Continuous RBN (Factor Analyzer)

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

Prior of h_j $\qquad p(h_j = 1) = \sigma(b_j)$

Conditional prob of x_i $\qquad p(x_i \mid \mathbf{h}) \sim N(\mathbf{W}_i\mathbf{h} + a_i, \sigma_i^2)$

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 follows a Gaussian distribution, with its mean specified by the weight vector $\mathbf{W}_i$ and bias vector a_i.

117

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

$$p(\mathbf{x},\mathbf{h}) = \prod_j (\sigma(b_j))^{h_j}(1-\sigma(b_j))^{1-h_j}\prod_i (\sigma(\mathbf{W}_i\mathbf{h}+a_i))^{x_i}(1-\sigma(\mathbf{W}_i\mathbf{h}+a_i))^{1-x_i}$$

$$= \frac{e^{\mathbf{b}^t\mathbf{h}+\mathbf{a}^t\mathbf{x}+\mathbf{x}^t\mathbf{W}\mathbf{h}}}{\prod_j (1+e^{b_j})\prod_i (1+e^{\mathbf{W}_i\mathbf{h}+a_i})}$$

$$= \frac{e^{\mathbf{b}^t\mathbf{h}+\mathbf{a}^t\mathbf{x}+\mathbf{x}^t\mathbf{W}\mathbf{h}+\sum_i \log(\mathbf{W}_i\mathbf{h}+a_i)}}{\prod_j (1+e^{b_j})} = \frac{\exp(-E(\mathbf{x},\mathbf{h}\mid\Theta))}{\prod_j (1+e^{b_j})} = \frac{\exp(-E(\mathbf{x},\mathbf{h}\mid\Theta))}{Z}$$

where $E(\mathbf{x},\mathbf{h}\mid\Theta) = -\mathbf{b}^t\mathbf{h} - \mathbf{a}^t\mathbf{x} - \mathbf{x}^t\mathbf{W}\mathbf{h} + \sum_i \log(1+\exp(\mathbf{W}_i\mathbf{h}+a_i))$ is the the energy functiuon and Z is the normalizat ion constant.

Compared to RBM, RBN energy function is more complex, but its normalization Z is simple. Similarly, we can derive joint probability distributions for continuous RBNs.

118

RBN Parameter Learning-Direct Method

Given dataset $\boldsymbol{D} = \{\boldsymbol{x}(m)\}_{m=1}^{M}$, the objective function for maximum likelihood learning is to find Q=(**a**,**b**, **W**) that maximizes the log marginal likelihood of the data, i.e.,

$$\Theta^* = \underset{\Theta}{\operatorname{argmax}} \log p(\mathbf{D}|\Theta) = \arg\max_{\Theta} \sum_m \log \sum_{\boldsymbol{h}} P(\boldsymbol{x}(m), \boldsymbol{h}|\Theta)$$

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

$$\nabla\Theta = \frac{\partial \sum_m \log \sum_{\mathbf{h}} p(\mathbf{x}(m),\mathbf{h}\mid\Theta)}{\partial\Theta} = -\sum_m\sum_{\mathbf{h}}[p(\mathbf{h}\mid\mathbf{x}(m),\Theta)\frac{\partial E(\mathbf{x}(m),\mathbf{h}\mid\Theta)}{\partial\Theta} + \frac{\partial \log Z}{\partial\Theta}]$$

$$\Theta^{t+1} = \Theta^t + \eta\nabla\Theta$$

Difficulties:

- Exponential number of terms in the sum over **h** (sampling, max, ..)
- Highly nonlinear and non-convex function and difficult to optimize
- Difficulty with inference p(**h**|**x**)

119

RBN Parameter Learning- EM Method

2) EM algorithm maximizes expected log joint likelihood, which is the lower bound of the marginal log-likelihood, i.e.,

Jensen inequality

$$\log p(\boldsymbol{x}(m)|\Theta) = log \sum_{\boldsymbol{h}} P(\boldsymbol{h}|\boldsymbol{x}(m),\Theta^t)\frac{p(\boldsymbol{x}(m),\boldsymbol{h}|\Theta)}{p(\boldsymbol{h}|\boldsymbol{x}(m),\Theta^t)} \geq \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{x}(m),\Theta^t)\log\frac{p(\boldsymbol{x}(m)|,\boldsymbol{h},\Theta)}{p(\boldsymbol{h}|\boldsymbol{x}(m),\Theta^t)}$$

$$\approx \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{x}(m),\Theta^t)\log p(\boldsymbol{x}(m)|,\boldsymbol{h},\Theta)$$

Expected log Likelihood

- EM algorithm:
- Start with an initialization of the parameters, iteratively perform:
  - E-step: compute $p(\boldsymbol{h}|\boldsymbol{x}(m),\Theta^t)$ using the current parameters for every possible **h**
  - M-step: find the parameter by solving,

$$\Theta^* = \arg\max_{\Theta}\sum_m\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{x}(m),\Theta^t)\log p(\boldsymbol{x}(m),\boldsymbol{h}|\Theta)$$

- Each step is convex and always improves the likelihood. Difficult to compute p(**h**|**x**), and exponential number of **h** configurations.

120

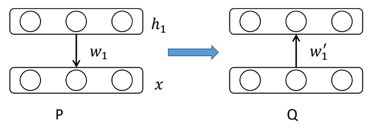
Inference in RBNs

The inference tasks include:

- Posterior probability inference P(**h**|**x**).
- Map inference- $\boldsymbol{h}^* = \arg\max_{\boldsymbol{h}} P(\boldsymbol{h}|\boldsymbol{x})$.
- Likelihood inference p(**x**)

RBN Inferences

- Posterior inference p(**h**|**x**)
  - p(**h**|**x**) is difficult since it cannot factorize over **h** because of explaining-away
  - Pesuido-likelihhood
    - ✓ $p(\boldsymbol{h}|\boldsymbol{x}) = \prod_j p(h_j|x, h_{-j})$
    - ✓ Variational methods
      - ✓ Mean field $p(\boldsymbol{h}|\boldsymbol{x}) \approx q(\boldsymbol{h}|\boldsymbol{x}) = \prod_j q(h_j|\boldsymbol{x})$
      - ✓ Inference network
- Posterior inference p(**x**|**h**)
  - p(**x**|**h**)=$\prod_i p(x_j|$**h**)



RBN Inferences

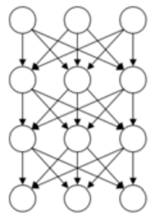
- MAP inference **h***=argmax $_{\mathbf{h}}$ p(**h**|**x**)
  - Coordinate ascent h_j*=argmax $_{hj}$ p(h_j |**x, h**$_{-j}$)
  - Variational method **h***=argmax $_{\mathbf{h}}$ q(**h**|**x**)
- MAP inference **x***=argmax $_{\mathbf{x}}$ p(**x**|**h**)
  - x_i*=argmax $_{\mathbf{x}}$ p(x_i|**h**)

- Likelihood inference p(**x**)
  - p(**x**)=$\sum_{\boldsymbol{h}} p(\boldsymbol{x}, \boldsymbol{h})$ for small **h**
  - Importance sampling p(**x**)=$\sum_{\boldsymbol{h}} p(\boldsymbol{x}, \boldsymbol{h}) = \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{x}) \frac{p(\boldsymbol{x},\boldsymbol{h})}{q(\boldsymbol{h}|\boldsymbol{x})} \approx \sum_{n=1}^{N} \frac{p(\boldsymbol{x},\boldsymbol{h}_n)}{q(\boldsymbol{h}_n|x)}$, $\boldsymbol{h}_n \sim q(\boldsymbol{h}|\boldsymbol{x})$

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 Deep Sigmoid Belief Network
- For continuous RBN, DRBN becomes Deep Factor Analyzers (DFAs)


Deep RBN-DRBN

Deep Regression Bayesian Networks

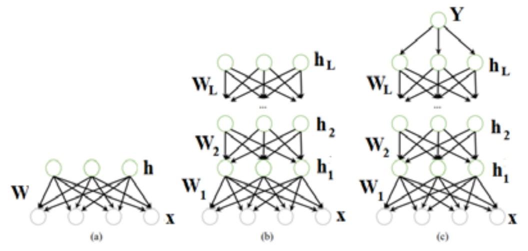


Fig. 2: Graph representation of (a) RBN, (b) DRBN without labels, and (c) DRBN with labels.

$$p(\mathbf{x},\mathbf{h}_1,\mathbf{h}_2,...,\mathbf{h}_L) = p(\mathbf{x}\,|\,\mathbf{h}_1)\,p(\mathbf{h}_1\,|\,\mathbf{h}_2)..p(\mathbf{h}_{L-1}\,|\,\mathbf{h}_L)\,p(\mathbf{h}_L)$$

125

Deep Regression Bayesian Network Learning

•**Pre-training**: layer-wise training follows RBN training method, with the output of previous layer as input of current layer.

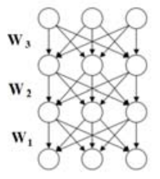
$$\mathbf{W}_l^* = \arg\max_{\mathbf{W}_l} \sum_m \log p(\mathbf{h}_l^*(m) \ \ |\, \mathbf{W}_l),$$

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

•**Unsupervised joint training** to refine the parameters for all layers simultaneously

$$\mathbf{W}^* = \arg\max_{\mathbf{W}} \sum_m \log p(\mathbf{x}(m)\,|\,\mathbf{W}) = \sum_m \log \sum_{\mathbf{h}_1,\mathbf{h}_2,...,\mathbf{h}_L} p(\mathbf{h}_1,\mathbf{h}_2,...,\mathbf{h}_L,\mathbf{x}(m)\,|\,\mathbf{W})$$

$$\text{where } \mathbf{W} = (\mathbf{W}_1,\mathbf{W}_2,...,\mathbf{W}_L)$$

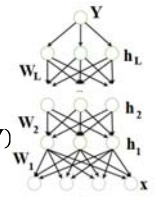


DRBN

•**Challenges**: exponential number of hidden layer configurations, and intractable inference with p($\mathbf{h}_l$|**x**).

126

DRBN Supervised Fine Tuning



Given training data **D**={**x**(m), **y**(m)}, m=1,2,…,M

1) Maximize the joint likelihood p(**x**,**y**) for fine-tuning,

$$\boldsymbol{W}^* = \arg\max_{\boldsymbol{W}} \sum_m \log \sum_{\boldsymbol{h}_1,\boldsymbol{h}_2,..,\boldsymbol{h}_L} p(\boldsymbol{x}(m),\boldsymbol{h}_1,\boldsymbol{h}_2,..,\boldsymbol{h}_L,\boldsymbol{y}(m)|\boldsymbol{W})$$

- Use gradient ascent algorithm with the pre-training result as an initialization for the parameters.

2) Maximize the conditional likelihood P(y|x) for discriminative learning,

$$\boldsymbol{W}^* = \arg\max_{\boldsymbol{W}} \sum_m \log p(\boldsymbol{y}(m)|\boldsymbol{x}(m),\boldsymbol{W})$$

$$= \arg\max_{\boldsymbol{W}} \textstyle\sum_m \log \sum_{\boldsymbol{h}_1,\boldsymbol{h}_2,..,\boldsymbol{h}_L} p(\boldsymbol{y}(m),\boldsymbol{h}_1,\boldsymbol{h}_2,..,\boldsymbol{h}_L|\boldsymbol{x}(m),\mathbf{W})$$

Optimization through gradient ascent.

- Same computational challenges: exponential number of hidden configurations and inference challenges.

127

DRBN Inference

- Posterior inference:
  - p($\mathbf{h}^l$|**x**) –approximated with sampling or pseudo-likelihood or or variational inference with either inference network or mean field method
  - p(**x**|$\mathbf{h}_L$)- sampling to approximate summation and variational methods to break down the dependences.
- MAP inference
  - $\mathbf{h}_l$*=argmax$_{\mathbf{h}l}$ p($\mathbf{h}_l$|**x**) – coordinate ascent or variational inference or their combination –augmented coordinate ascent.
- Likelihood inference
  - p(**x**) –intractable due to exponential number of hidden node configurations and approximated by sampling or variational inference or max operation.

128

Deep Directed Networks

Training	Inference	Structure	Advantages
Layerwise and fine tuning using direct or EM methods	Posterior, MAP, and Likelihood inferences	Limited to a few hidden layers	Fully probabilistic
1. Exponential numbers of latent configurations 1. Inference challenges	Intractable inference due to dependencies among latent variables. Solutions include variational methods and coordinate ascent.	Because of learning and inference complexity	Capture the dependencies among latent variables.

129

DRBN Application: Image inpainting

Given an **x** and a 2 layer DRBN,
1)**h*** = argmax$_x$ p(**h**|**x**)
2) $\hat{x}$=argmax$_x$ p(**x**|**h***)

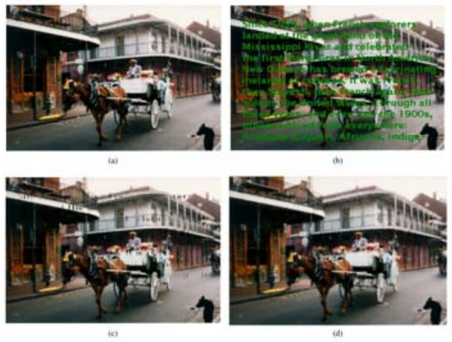


Fig. 4: An example of the image inpainting experiment, (a) original image, (b) corrupted image, (c) GMM, and (d) DRBN. Images collected from [19]

130

DRBN Application: Image Restoration

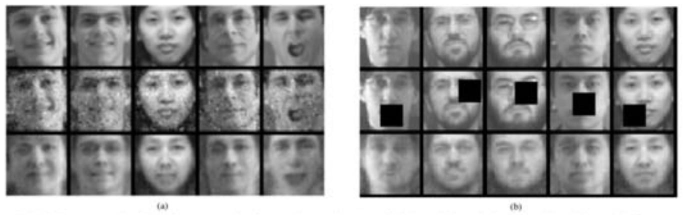


Fig. 5: Some examples of the face restoration from random noise (a) and block occlusion (b). Images collected from [9]. From top to bottom, the rows represent original images, corrupted images, and reconstructed images.

1)**h*** = argmax$_x$ p(**h**|**x**)
2) $\hat{x}$=argmax$_x$ p(**x**|**h***)

131

DRBN Application: Image synthesis

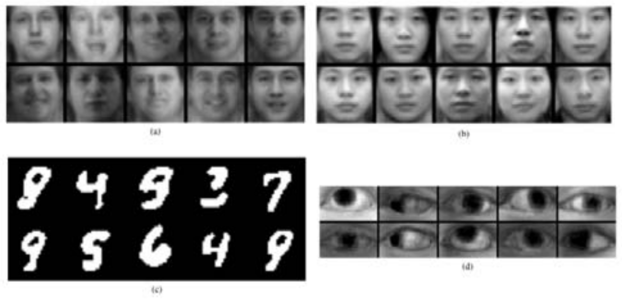


Fig. 6: Examples of the image reconstruction for (a) Multi-PIE, (b) CAS-PEAL, (c) MNIST, (d) UnityEye.

1)**h*** ~ p(**h**)
2) $\hat{x}$=argmax$_x$ p(**x**|**h***)

132

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

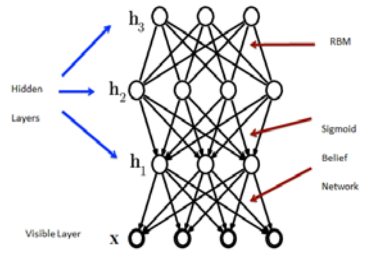
1) Train one DRBN p(**x**|c) for each class c
2) given a query image **x**, c*=argmax$_c$ p(**x**|c) , i.e., compute the likelihood for each model and identify the model with the highest likelihood.

133

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 **x** and hidden variables ($\{\mathbf{h}_1, \dots, \mathbf{h}_l\}$), $p(\mathbf{x}, \mathbf{h}_1, \dots, \mathbf{h}_l; \boldsymbol{W})$
- top 2 layers' distribution $p(\mathbf{h}_2, \mathbf{h}_3)$ is an RBM!
- other layers form a Bayesian network with conditional distributions:



25

Courtesy of Russ Salakhutdinov of CMU

Deep Belief Network

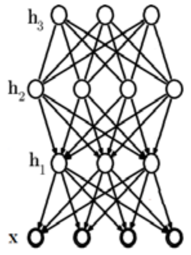
- The joint distribution of a DBN is as follows

$$p(\mathbf{x},\mathbf{h}_1,\mathbf{h}_2,\mathbf{h}_3) = p(\mathbf{h}_2,\mathbf{h}_3)p(\mathbf{h}_1 \mid \mathbf{h}_2)p(\mathbf{x} \mid \mathbf{h}_1)$$

$$p(\mathbf{h}_2,\mathbf{h}_3) = \exp(-E(\mathbf{b}_2^t\mathbf{h}_2 + \mathbf{b}_3^t\mathbf{h}_3 + \mathbf{h}_2\mathbf{W}\mathbf{h}_3))/Z$$

$$p(\mathbf{h}_1 \mid \mathbf{h}_2) = \prod_j p(\mathbf{h}_{1,j} \mid \mathbf{h}_2)$$

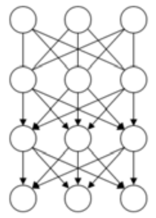
$$p(\mathbf{x} \mid \mathbf{h}_1) = \prod_i p(x_i \mid \mathbf{h}_2)$$



- 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.

DBN Training

- 1. Greedy layerwise pre-training
- 2. Parameter fine-tuning:

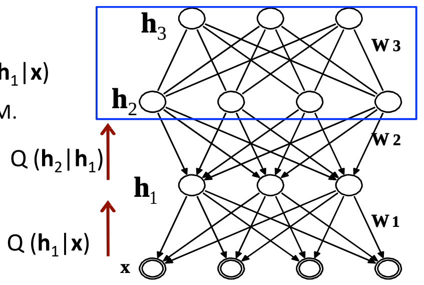


Deep belief network

137

DBN Layer-wise Training

- Every two layers form an RBM, due to the complementary prior, thus can be trained using CD method.
- Approximate the inferred values
- $\mathbf{h}_1$*=argmax Q ($\mathbf{h}_1$|$\mathbf{x}$)≈ argmax$p$($\mathbf{h}_1$|$\mathbf{x}$)
- as the data for training 2nd-layer RBM.
- Learn and freeze 2nd layer RBM.
- Proceed to the next layer.



Courtesy of Russ Salakhutdinov of CMU

DBN Unsupervised Fine-tuning

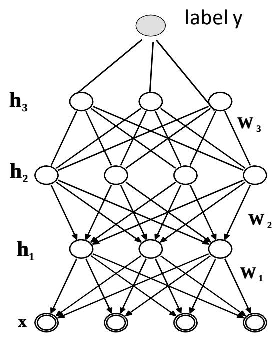
After pre- training, we can fine tune the model parameters in top-down manner

1. Sample the top layer $\mathbf{h}_L$ to obtain samples for $\mathbf{h}^*_L$
2. Adjust the parameters $\mathbf{W}_L$ to maximize p($\mathbf{h}^*_L$| $\mathbf{W}_L$)
3. Fix $\mathbf{W}_L$ and perform a top-down sampling to obtain $\mathbf{h}^*_{L-1}$ and repeat this until the first hidden layer
4. Update $\mathbf{W}_1$ to maximize p($\mathbf{h}^*_{1,}$ $\mathbf{x}$| $\mathbf{W}_1$)
5. Using the updated parameters **W** to perform a bottom-up layerwise learning
6. Repeat steps 1-5 until convergence

DBN and supervised fine-tuning

- Discriminative fine-tuning
  - Initializing with neural nets + backpropagation
  - Maximizes $\log P(Y|X)$ (X: data Y: label)
- Generative fine-tuning
  - Up-down algorithm
  - Maximizes $\log P(Y,X)$ (joint likelihood of data and labels)

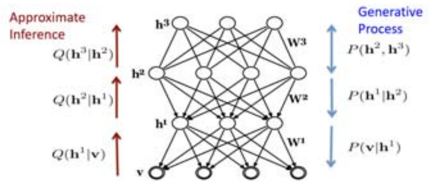
Similar top-down methods for unsupervised fine tuning can apply.



Courtesy of Russ Salakhutdinov of CMU

Deep Belief Network Inference

- Posterior inference
  - Bottom-up inference p($\mathbf{h}_l$|$\mathbf{x}$) - variational inference p($\mathbf{h}_l$|$\mathbf{x}$) = q($\mathbf{h}_l$|$\mathbf{x}$)
  - Top-down p($\mathbf{x}$|$\mathbf{h}_L$) - ancestral sampling
- MAP inference
  - Bottom-up inference $\mathbf{h}^*_l$=argmax$_{hl}$ p($\mathbf{h}_l$|$\mathbf{x}$)
    - variational method $\mathbf{h}^*_l$=argmax$_{hl}$ q($\mathbf{h}_l$|$\mathbf{x}$)
  - Top-down $\mathbf{x}^*$=argmax$_x$ p($\mathbf{x}$|$\mathbf{h}_L$)



DRBN v.s. DBN v.s. DBM

DRBN
- Deep directed models have no assumption about the prior of the latent variables.
- Constructed from layers of RBNs.
- Parameterized with CPDs

DBN
- Hybrid models, with RBM as the top two layers and RBN for the remaining layers.
- Uses "complementary priors" to eliminate the explaining-away effects and effectively convert the DBN to DBM during learning.
- Parameterized with potentials for top two layers and CPDs for remaining layers

DBM
- Undirected deep model and constructed using RBMs
- Parameterized with pairwise and unary potentials.

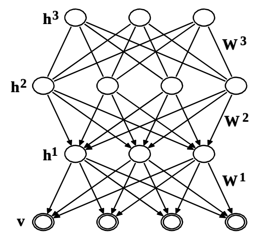
DBM, DRBN, and DBN Comparison

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

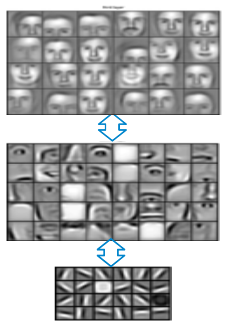
- In general, DRBN has better representation but more complex inference. DBM/DBN are more efficient in inference but are less powerful in representation.
- DBM/DBN must deal with partition function, while DRBN must deal with hidden nodes dependencies

DBN Representation Learning

Convolutional DBN



Faces



Groups of parts.

h^*_3=argmax $_{h3}$ p(h_3|v)

Object Parts

h^*_2=argmax $_{h2}$ p(h_2|v)

Edges

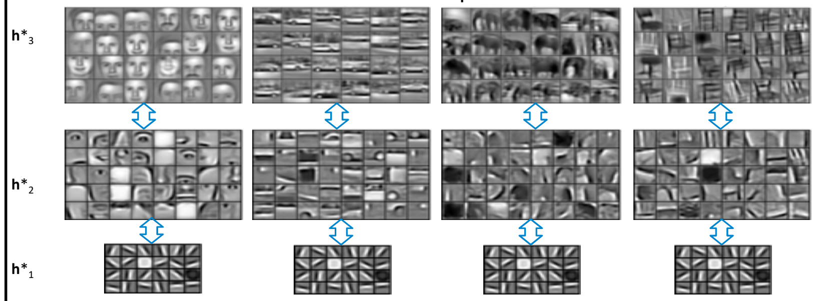
h^*_1=argmax $_{h1}$ p(h_1|v)

Courtesy of Russ Salakhutdinov of CMU

Lee et.al., ICML 2009

DBN Representation Learning

Faces Cars Elephants Chairs

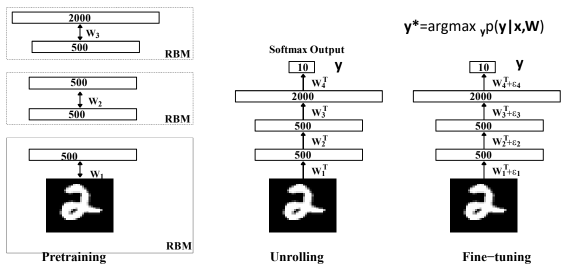


Courtesy of Russ Salakhutdinov of CMU

Lee et.al., ICML 2009

DBNs for Classification

$y^*=\text{argmax}_{y}\,p(\mathbf{y}|\mathbf{x},\mathbf{W})$



•After layer-by-layer **unsupervised pretraining**, discriminative fine-tuning achieves an error rate of 1.2% on MNIST.

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

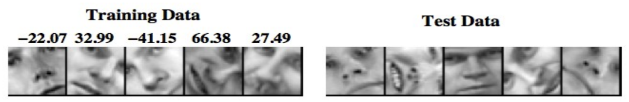
(Hinton and Salakhutdinov, Science 2006)

Courtesy of Russ Salakhutdinov of CMU

DBNs for Regression

Predicting the orientation of a face patch

$y^*=\text{argmax}_{y}\,p(\mathbf{y}|\mathbf{x},\mathbf{W})$



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. Achieves a RMSE (root mean squared error) of 16.33 degree.

(Salakhutdinov and Hinton, NIPS 2007)

Courtesy of Russ Salakhutdinov of CMU

Summary and Conclusions

Probabilistic deep learning models include

- Probabilistic deep neural networks
- Bayesian deep neural networks
- Deep probabilistic graphical models
  - Deep Boltzmann Machine
  - Deep Regression Bayesian networks
  - Deep Belief Networks

163

Summary and Conclusions

- Advantages:
  - Probabilistic deep learning has the abilities to capture data, model, and output uncertainties.
  - Probabilistic deep learning can quantify prediction uncertainty, which can be used for low quality and outlier data detection, quantification of output confidence, active learning, and combination of different models.
  - Bayesian deep learning can avoid the learning process and performs prediction using all parameters, hence avoiding overfitting problem.
  - Deep probabilistic graphical models fully model and capture the input, model, and output uncertainties.
- Challenges:
  - Probabilistic deep learning suffers from intractable inference problems. It hence cannot scale up well.
  - MCMC sampling and variational methods are currently two most popular methods to address the inference complexities. These techniques require further developments.
- Future
  - Probabilistic deep learning still has many theoretical challenges to overcome before it can be applied in large scale as deep neural networks.
  - It has great potential for many applications, it represents a AI frontier research, and its success could lead to another AI revolution.

164