Bayesian Deep Learning and Restricted Boltzmann Machines

Narada Warakagoda

Forsvarets Forskningsinstitutt

ndw@ffi.no

November 1, 2018

Overview

1. Probability Review
2. Bayesian Deep Learning
3. Restricted Boltzmann Machines

Probability Review

Probability and Statistics Basics

- Normal (Gaussian) Distribution

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\} = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

- Categorical Distribution

$$P(x) = \prod_{i=1}^{k} p_i^{[x=i]}$$

- Sampling

$$\boldsymbol{x} \sim p(\boldsymbol{x})$$

- Independent variables

$$p\left(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_k\right) = \prod_{i=1}^{k} p\left(\boldsymbol{x}_i\right)$$

- Expectation

$$\mathbb{E}_{p(\boldsymbol{x})} f\left(\boldsymbol{x}\right) = \int f\left(\boldsymbol{x}\right) p\left(\boldsymbol{x}\right) dx$$

or for discrete variables

$$\mathbb{E}_{p(\boldsymbol{x})} f\left(\boldsymbol{x}\right) = \sum_{i=1}^{k} f\left(\boldsymbol{x}_i\right) P\left(\boldsymbol{x}_i\right)$$

Kullback Leibler Distance

$$KL\left(q\left(\boldsymbol{x}\right)||p\left(\boldsymbol{x}\right)\right) = \mathbb{E}_{q(\boldsymbol{x})} \log\left[\frac{q\left(\boldsymbol{x}\right)}{p\left(\boldsymbol{x}\right)}\right]$$
$$= \int \left[q\left(\boldsymbol{x}\right)\log q\left(\boldsymbol{x}\right) - q\left(\boldsymbol{x}\right)\log p\left(\boldsymbol{x}\right)\right] d\boldsymbol{x}$$

For the discrete case

$$KL\left(Q\left(\boldsymbol{x}\right)||P\left(\boldsymbol{x}\right)\right) = \sum_{i=1}^{k} \left[Q\left(\boldsymbol{x}_i\right)\log Q\left(\boldsymbol{x}_i\right) - Q\left(\boldsymbol{x}_i\right)\log P\left(\boldsymbol{x}_i\right)\right]$$

Bayesian Deep Learning

Bayesian Statistics

- Joint distribution

$$p(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{x}|\boldsymbol{y})\, p(\boldsymbol{y})$$

- Marginalization

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}, \boldsymbol{y})\, d\boldsymbol{y}$$

$$P(\boldsymbol{x}) = \sum_{\boldsymbol{y}} P(\boldsymbol{x}, \boldsymbol{y})$$

- Conditional distribution

$$p(\boldsymbol{x}|\boldsymbol{y}) = \frac{p(\boldsymbol{x}, \boldsymbol{y})}{p(\boldsymbol{y})} = \frac{p(\boldsymbol{y}|\boldsymbol{x})\, p(\boldsymbol{x})}{\int p(\boldsymbol{y}|\boldsymbol{x})\, p(\boldsymbol{x})\, d\boldsymbol{x}}$$

Statistical view of Neural Networks

- Prediction

$$p\left(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{w}\right)=\mathcal{N}\left(\boldsymbol{f}_{\boldsymbol{w}}\left(\boldsymbol{x}\right),\boldsymbol{\Sigma}\right)$$

- Classification

$$P\left(y|\boldsymbol{x},\boldsymbol{w}\right)=\prod_{i=1}^{k}\boldsymbol{f}_{\boldsymbol{w}}^{i}\left(\boldsymbol{x}\right)^{[y=i]}$$

Training Criteria

- Maximum Likelihood(ML)

$$\widehat{\boldsymbol{w}} = \arg\max_{\boldsymbol{w}} p\left(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{w}\right)$$

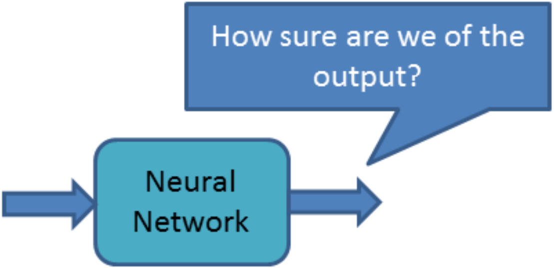
- Maximum A-Priori (MAP)

$$\widehat{\boldsymbol{w}} = \arg\max_{\boldsymbol{w}} p\left(\boldsymbol{Y}, \boldsymbol{w}|\boldsymbol{X}\right) = \arg\max_{\boldsymbol{w}} p\left(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{w}\right) p(\boldsymbol{w})$$

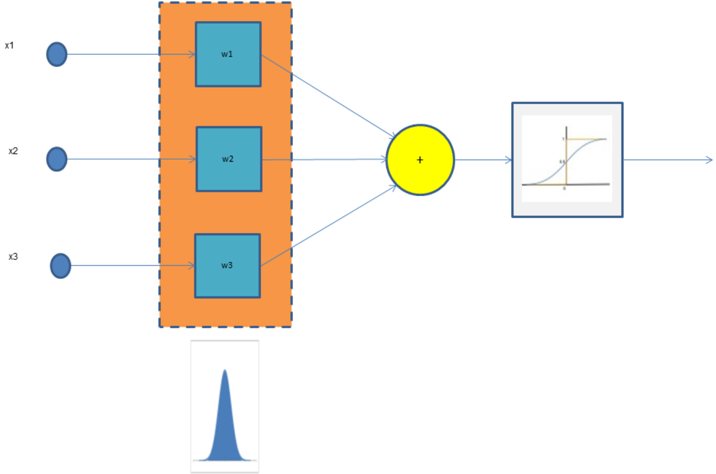
- Bayesian

$$p\left(\boldsymbol{w}|\boldsymbol{Y}, \boldsymbol{X}\right) = \frac{p\left(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{w}\right) p\left(\boldsymbol{w}\right)}{P\left(\boldsymbol{Y}|\boldsymbol{X}\right)} = \frac{p\left(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{w}\right) p\left(\boldsymbol{w}\right)}{\int P\left(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{w}\right) p\left(\boldsymbol{w}\right) d\boldsymbol{w}}$$

Motivation for Bayesian Approach



Motivation for Bayesian Approach



Uncertainty with Bayesian Approach

- Not only prediction/classification, but their uncertainty can also be calculated
  - Since we have $p(\boldsymbol{w}|\boldsymbol{Y},\boldsymbol{X})$ we can sample $\boldsymbol{w}$ and use each sample as network parameters in calculating the prediction/classification $p(\widehat{y}|\widehat{x},\boldsymbol{w}))$ (i.e.network output for a given input).
  - Prediction/classification is the mean of $p(\widehat{y}|\widehat{x},\boldsymbol{w})$

$$p_{out} = p(\widehat{y}|\widehat{x},\boldsymbol{Y},\boldsymbol{X}) = \int p(\widehat{y}|\widehat{x},\boldsymbol{w})\, p(\boldsymbol{w}|\boldsymbol{Y},\boldsymbol{X})\, d\boldsymbol{w}$$

  - Uncertainty of prediction/classification is the variance of $p(\widehat{y}|\widehat{x},\boldsymbol{w})$

$$\mathrm{Var}(p(\widehat{y}|\widehat{x},\boldsymbol{w})) = \int [p(\widehat{y}|\widehat{x},\boldsymbol{w}) - p_{out}]^2\, p(\boldsymbol{w}|\boldsymbol{Y},\boldsymbol{X})\, d\boldsymbol{w}$$

- Uncertainty is important in safety critical applications (eg: self-driving cars, medical diagnosis, military applications

Other Advantages of Bayesian Approach

- Natural interpretation for regularization
- Model selection
- Input data selection (active learning)

Main Challenge of Bayesian Approach

- We calculate
  - For continuous case:

$$p(\boldsymbol{w}|\boldsymbol{Y},\boldsymbol{X}) = \frac{p(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{w})\, p(\boldsymbol{w})}{\int P(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{w})\, p(\boldsymbol{w})\, d\boldsymbol{w}}$$

  - For discrete case:

$$P(\boldsymbol{w}|\boldsymbol{Y},\boldsymbol{X}) = \frac{p(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{w})\, P(\boldsymbol{w})}{\sum_{\boldsymbol{w}} p(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{w})\, P(\boldsymbol{w})}$$

- Calculating denominator is often intractable
  - Eg: Consider a weight vector $\boldsymbol{w}$ of 100 elements, each can have two values. Then there are $2^{100} = 1.2 \times 10^{30}$ different weight vectors. Compare this with universe's age 13.7 billion years.
- We need approximations

Different Approaches

- Monte Carlo techniques (Eg: Markov Chain Monte Carlo -MCMC)
- Variational Inference
- Introducing random elements in training (eg: Dropout)

Advantages and Disadvantages of Different Approaches

- Markov Chain Monte Carlo - MCMC
    - Asymptotically exact
    - Computationally expensive
- Variational Inference
    - No guarantee of exactness
    - Possibility for faster computation

Monte Carlo Techniques

- We are interested in

$$p_{out} = \text{Mean}(p(\widehat{y}|\widehat{x}, \boldsymbol{w})) = p(\widehat{y}|\widehat{x}, \boldsymbol{Y}, \boldsymbol{X}) = \int p(\widehat{y}|\widehat{x}, \boldsymbol{w})\, p(\boldsymbol{w}|\boldsymbol{Y}, \boldsymbol{X})\, d\boldsymbol{w}$$

$$\text{Var}(p(\widehat{y}|\widehat{x}, \boldsymbol{w})) = \int [p(\widehat{y}|\widehat{x}, \boldsymbol{w}) - p_{out}]^2\, p(\boldsymbol{w}|\boldsymbol{Y}, \boldsymbol{X})\, d\boldsymbol{w}$$

- Both are integrals of the type

$$I = \int F(\boldsymbol{w})\, p(\boldsymbol{w}|\mathcal{D})\, d\boldsymbol{w}$$

where $\mathcal{D} = (\boldsymbol{Y}, \boldsymbol{X})$ is training data.

- Approximate the integral by sampling $\boldsymbol{w}_i$ from $p(\boldsymbol{w}|\mathcal{D})$

$$I \approx \frac{1}{L} \sum_{i=1}^{L} F(\boldsymbol{w}_i).$$

Monte Carlo techniques

- Challenge: We don't have the posterior

$$p(\boldsymbol{w}|\mathcal{D}) = p(\boldsymbol{w}|\boldsymbol{Y}, \boldsymbol{X}) = \frac{p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{w})\, p(\boldsymbol{w})}{\int P(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{w})\, p(\boldsymbol{w})\, d\boldsymbol{w}}$$

- "Solution": Use importance sampling by sampling from a proposal distribution $q(\boldsymbol{w})$

$$I = \int F(\boldsymbol{w}) \frac{p(\boldsymbol{w}|\mathcal{D})}{q(\boldsymbol{w})} q(\boldsymbol{w})\, d\boldsymbol{w} \approx \frac{1}{L} \sum_{i=}^{L} F(\boldsymbol{w}_i) \frac{p(\boldsymbol{w}_i|D)}{q(\boldsymbol{w}_i)}$$

- Problem: We still do not have $p(\boldsymbol{w}|\mathcal{D})$

Monte Carlo Techniques

- Problem: We still do not have $p\left(\boldsymbol{w}|\mathcal{D}\right)$
- Solution: use unnormalized posterior $\tilde{p}\left(\boldsymbol{w}|\mathcal{D}\right) = p\left(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{w}\right)p\left(\boldsymbol{w}\right)$ where normalization factor $Z = \int P\left(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{w}\right)p\left(\boldsymbol{w}\right)d\boldsymbol{w}$ such that

$$p\left(\boldsymbol{w}|\mathcal{D}\right) = \frac{\tilde{p}\left(\boldsymbol{w}|\mathcal{D}\right)}{Z}$$

- Integral can be calculated with:

$$I \approx \frac{\sum_{i=1}^{L} F\left(\boldsymbol{w}_i\right)\tilde{p}\left(\boldsymbol{w}_i|D\right)/q\left(\boldsymbol{w}_i\right)}{\sum_{i=1}^{L}\tilde{p}\left(\boldsymbol{w}_i|D\right)/q\left(\boldsymbol{w}_i\right)}$$

Weakness of Importance Sampling

- Proposal distribution must be close to the non-zero areas of original distribution $p(\boldsymbol{w}|\mathcal{D})$.
- In neural networks, $p(\boldsymbol{w}|\mathcal{D})$ is typically small except for few narrow areas.
- Blind sampling from $q(\boldsymbol{w})$ has a high chance that they fall outside non-zero areas of $p(\boldsymbol{w}|\mathcal{D})$
- We must actively try to get samples that lie close to $p(\boldsymbol{w}|\mathcal{D})$
- Markov Chain Monte Carlo (MCMC) is such technique.

Metropolis Algorithm

- Metropolis algorithm is an example of MCMC
- Draw samples repeatedly from random walk $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon}$ is a small random vector, $\boldsymbol{\epsilon} \sim q(\boldsymbol{\epsilon})$ (eg: Gaussian noise)
- Drawn sample at $t = t$ is either accepted based on the ratio $\frac{\tilde{p}(\boldsymbol{w}_t|\mathcal{D})}{\tilde{p}(\boldsymbol{w}_{t-1}|\mathcal{D})}$
  - If $\tilde{p}\left(\boldsymbol{w}_t|\mathcal{D}\right) > \tilde{p}\left(\boldsymbol{w}_{t-1}|\mathcal{D}\right)$ accept sample
  - If $\tilde{p}\left(\boldsymbol{w}_t|\mathcal{D}\right) < \tilde{p}\left(\boldsymbol{w}_{t-1}|\mathcal{D}\right)$ accept sample with probability $\frac{\tilde{p}(\boldsymbol{w}_t|\mathcal{D})}{\tilde{p}(\boldsymbol{w}_{t-1}|\mathcal{D})}$
  - If sample accepted use it for calculating I
- Can use the same formula for calculating I

$$I \approx \frac{\sum_{i=1}^{L} F\left(\boldsymbol{w}_i\right) \tilde{p}\left(\boldsymbol{w}_i|D\right)/q\left(\boldsymbol{w}_i\right)}{\sum_{i=1}^{L} \tilde{p}\left(\boldsymbol{w}_i|D\right)/q\left(\boldsymbol{w}_i\right)}$$

Other Monte Carlo and Related Techniques

- Hybrid Monte Carlo (Hamiltonian Monte Carlo)
  - Similar to Metropolis algorithm
  - But uses gradient information rather than a random walk.
- Simulated Annealing

Variational Inference

- Goal: computation of posterior $p\left(\boldsymbol{w}|\mathcal{D}\right)$, i.e. the parameters of the neural network $\boldsymbol{w}$ given data $\mathcal{D} = (\boldsymbol{Y}, \boldsymbol{X})$
- But this computation is often intractable
- Idea: find a distribution $q(\boldsymbol{w})$ from a family of distributions Q such that $q(\boldsymbol{w})$ can closely approximate $p(\boldsymbol{w}|\mathcal{D})$
- How to measure the distance between $q(\boldsymbol{w})$ and $p(\boldsymbol{w}|\mathcal{D})$?
  - Kullback-Leibler Distance $\mathrm{KL}\big(q(\boldsymbol{w})||p(\boldsymbol{w}|\mathcal{D})\big)$
- The problem can be formulated as

$$\hat{p}(\boldsymbol{w}|\mathcal{D}) = \arg\min_{q(\boldsymbol{w})} \mathrm{KL}\big(q(\boldsymbol{w})||p(\boldsymbol{w}|\mathcal{D})\big)$$

Minimizing KL Distance

- Using the definition of KL distance

$$\mathrm{KL}\big(q(\boldsymbol{w})||p(\boldsymbol{w}|\mathcal{D})\big) = \int q\left(\boldsymbol{w}\right) \ln \frac{q\left(\boldsymbol{w}\right)}{p\left(\boldsymbol{w}|\mathcal{D}\right)} d\boldsymbol{w}$$

- Cannot minimize this directly, because we do not know $p\left(\boldsymbol{w}|\mathcal{D}\right)$
- But we can manipulate it further, and transform it to another equivalent optimization problem involving a quantity known as Evidence Lower Bound (ELBO)

Evidence Lower Bound (ELBO)

$$
\begin{aligned}
\mathrm{KL}\big(q(\boldsymbol{w})||p(\boldsymbol{w}|\mathcal{D})\big) &= \int q(\boldsymbol{w}) \ln \frac{q(\boldsymbol{w})}{p(\boldsymbol{w}|\mathcal{D})} d\boldsymbol{w} \\
&= \int q(\boldsymbol{w}) \ln \frac{q(\boldsymbol{w})\, p(\mathcal{D})}{p(\boldsymbol{w}, \mathcal{D})} d\boldsymbol{w} \\
&= \int q(\boldsymbol{w}) \ln \frac{q(\boldsymbol{w})}{p(\boldsymbol{w}, \mathcal{D})} d\boldsymbol{w} + \int q(\boldsymbol{w}) \ln p(\mathcal{D}) d\boldsymbol{w} \\
&= \mathbb{E}_{q(\boldsymbol{w})} \ln \frac{q(\boldsymbol{w})}{p(\boldsymbol{w}, \mathcal{D})} + \ln p(\mathcal{D}) \int q(\boldsymbol{w})\, d\boldsymbol{w} \\
\ln p(\mathcal{D}) &= \boxed{\mathbb{E}_{q(\boldsymbol{w})} \ln \frac{p(\boldsymbol{w}, \mathcal{D})}{q(\boldsymbol{w})}} + \mathrm{KL}\big(q(\boldsymbol{w})||p(\boldsymbol{w}|\mathcal{D})\big)
\end{aligned}
$$

- Since $\ln p(\mathcal{D})$ is constant, minimizing $\mathrm{KL}\big(q(\boldsymbol{w})||p(\boldsymbol{w}|\mathcal{D})\big)$ is equivalent to maximizing ELBO

Another Look at ELBO

$$
\begin{aligned}
\text{ELBO} &= \mathbb{E}_{q(\boldsymbol{w})} \ln \frac{p(\boldsymbol{w}, \mathcal{D})}{q(\boldsymbol{w})} \\
&= \int q(\boldsymbol{w}) \ln p(\boldsymbol{w}, \mathcal{D}) d\boldsymbol{w} - \int q(\boldsymbol{w}) \ln q(\boldsymbol{w}) d\boldsymbol{w} \\
&= \int q(\boldsymbol{w}) \ln [p(\mathcal{D}|\boldsymbol{w}) p(\boldsymbol{w})] d\boldsymbol{w} - \int q(\boldsymbol{w}) \ln q(\boldsymbol{w}) d\boldsymbol{w} \\
&= \int q(\boldsymbol{w}) \ln p(\mathcal{D}|\boldsymbol{w}) d\boldsymbol{w} - \int q(\boldsymbol{w}) \ln \frac{q(\boldsymbol{w})}{p(\boldsymbol{w})} d\boldsymbol{w} \\
&= \mathbb{E}_{q(\boldsymbol{w})} p(\mathcal{D}|\boldsymbol{w}) - \text{KL}\big(q(\boldsymbol{w}) || p(\boldsymbol{w})\big)
\end{aligned}
$$

- We maximize ELBO with respect to $q(\boldsymbol{w})$
- First term $\mathbb{E}_{q(\boldsymbol{w})} p(\mathcal{D}|\boldsymbol{w})$ is equivalent to maximizing $q(\boldsymbol{w})$'s ability explain training data
- Second term $\text{KL}\big(q(\boldsymbol{w}) || p(\boldsymbol{w})\big)$ is equivalent to minimizing $q(\boldsymbol{w})$'s distance to $p(\boldsymbol{w})$

Outline of Procedure with ELBO

- Start with ELBO

$$\text{ELBO} = \mathcal{L} = \mathbb{E}_{q(\boldsymbol{w})} \ln \frac{p\left(\boldsymbol{w}, \mathcal{D}\right)}{q\left(\boldsymbol{w}\right)} = \mathbb{E}_{q(\boldsymbol{w})}\left[\ln p\left(\boldsymbol{w}, \mathcal{D}\right) - \ln q\left(\boldsymbol{w}\right)\right]$$

- Rewrite with parameter λ of $q\left(\boldsymbol{w}\right)$ and expand expectation

$$\mathcal{L}(\lambda) = \int \ln[p\left(\boldsymbol{w}, \mathcal{D}\right)] q\left(\boldsymbol{w}, \lambda\right) d\boldsymbol{w} - \int \ln[q\left(\boldsymbol{w}, \lambda\right)] q\left(\boldsymbol{w}, \lambda\right) d\boldsymbol{w}$$

- Maximize $\mathcal{L}(\lambda)$ with respect to λ

$$\lambda^\star = \arg\max_{\lambda} \mathcal{L}(\lambda)$$

- Use the optimized q witn respect to λ as posterior

$$q\left(\boldsymbol{w}, \lambda^\star\right) = p(\boldsymbol{w}, \mathcal{D})$$

How to Maximize ELBO

- Analytical methods are not practical for deep neural networks
- We resort to gradient methods with Monte Carlo sampling
- We discuss two methods:
  - Black box variational inference: Based on log derivative trick
  - Bayes by Backprop: Based on re-parameterization trick

Black Box Variational Inference

- Start with ELBO:

$$\mathcal{L}(\lambda) = \int \ln[p(\mathbf{w}, \mathcal{D})] q(\mathbf{w}, \lambda)\, d\mathbf{w} - \int \ln[q(\mathbf{w}, \lambda)] q(\mathbf{w}, \lambda)\, d\mathbf{w}$$

- Differentiate with respect to λ.

$$\begin{aligned}\nabla_\lambda \mathcal{L}(\lambda) = & \int \ln[p(\mathbf{w}, \mathcal{D})] \nabla_\lambda [q(\mathbf{w}, \lambda)] d\mathbf{w} \\ & - \int \ln[q(\mathbf{w}, \lambda)] \nabla_\lambda [q(\mathbf{w}, \lambda)] d\mathbf{w} \\ & - \int \nabla_\lambda \Big[\ln[q(\mathbf{w}, \lambda)] \Big] q(\mathbf{w}, \lambda)\, d\mathbf{w}\end{aligned}$$

- The last term is zero (Can you prove it?)

Black Box Variational Inference

- Now we have

$$\begin{aligned}\nabla_\lambda \mathcal{L}(\lambda) &= \int \ln[p\,(\boldsymbol{w}, \mathcal{D})] \nabla_\lambda [q\,(\boldsymbol{w}, \lambda)] d\boldsymbol{w} \\ &\quad - \int \ln[q\,(\boldsymbol{w}, \lambda)] \nabla_\lambda [q\,(\boldsymbol{w}, \lambda)] d\boldsymbol{w} \\ &= \int \Big[[p\,(\boldsymbol{w}, \mathcal{D})] - \ln[q\,(\boldsymbol{w}, \lambda)] \Big] \nabla_\lambda [q\,(\boldsymbol{w}, \lambda)] dw\end{aligned}$$

- We want to write this as an expectation with respect to q
- Use the log derivative trick

$$\nabla_\lambda [q\,(\boldsymbol{w}, \lambda)] = \nabla_\lambda [\ln q\,(\boldsymbol{w}, \lambda)] q\,(\boldsymbol{w}, \lambda)$$

Black Box Variational Inference

- Now we get

$$\nabla_\lambda \mathcal{L}(\lambda) = \int \ln[p(\boldsymbol{w}, \mathcal{D})] \nabla_\lambda [\ln q(\boldsymbol{w}, \lambda)] q(\boldsymbol{w}, \lambda)\, d\boldsymbol{w} - \int \ln[q(\boldsymbol{w}, \lambda)] \nabla_\lambda [\ln q(\boldsymbol{w}, \lambda)] q(\boldsymbol{w}, \lambda)\, d\boldsymbol{w}$$

- Rearranging terms

$$\nabla_\lambda \mathcal{L}(\lambda) = \int \Big[\ln[p(\boldsymbol{w}, \mathcal{D})] - \ln q(\boldsymbol{w}, \lambda)\Big] \nabla_\lambda [\ln q(\boldsymbol{w}, \lambda)] q(\boldsymbol{w}, \lambda)\, d\boldsymbol{w}$$

- This is the same as Expectation with respect to q

$$\nabla_\lambda \mathcal{L}(\lambda) = \mathbb{E}_{q(\boldsymbol{w}, \lambda)} \Big[\ln[p(\boldsymbol{w}, \mathcal{D})] - \ln q(\boldsymbol{w}, \lambda)\Big] \nabla_\lambda [\ln q(\boldsymbol{w}, \lambda)]$$

BBVI optimization procedure

- Assume a distribution $q(\boldsymbol{w}, \lambda)$ parameterized by λ.
- Draw S samples of $\boldsymbol{w}$ from the distribution using the current value of $\lambda = \lambda_t$
- Estimate the gradient of ELBO using the sample values:

$$\nabla_\lambda \hat{\mathcal{L}}(\lambda) = \frac{1}{S} \sum_{s=1}^{S} \left[\ln[p(\boldsymbol{w}^s, \mathcal{D})] - \ln q(\boldsymbol{w}^s, \lambda) \right] \nabla_\lambda [\ln q(\boldsymbol{w}^s, \lambda)]$$

- Update λ

$$\lambda_{t+1} = \lambda_t + \rho \nabla_\lambda \hat{\mathcal{L}}(\lambda)$$

- repeat from step 2

Bayes by Backprop

- Try to approximate ELBO directly by sampling from the $q(\boldsymbol{w}, \lambda)$

$$\text{ELBO} = \mathcal{L}(\lambda) = \mathbb{E}_{q(\boldsymbol{w},\lambda)}\left[\ln p\left(\boldsymbol{w}, \mathcal{D}\right) - \ln q\left(\boldsymbol{w}, \lambda\right)\right]$$

with

$$\hat{\mathcal{L}}(\lambda) = \frac{1}{S}\sum_{s=1}^{S}\left[\ln p\left(\boldsymbol{w}^{s}, \mathcal{D}\right) - \ln q\left(\boldsymbol{w}^{s}, \lambda\right)\right]$$

- But we need $\nabla_{\lambda}\hat{\mathcal{L}}(\lambda)$ and we can not differentiate $\hat{\mathcal{L}}(\lambda)$ because it is not a smooth function of λ
- Use the re-parameterization trick

$$\boldsymbol{w}^{s} = \boldsymbol{w}(\lambda, \boldsymbol{\epsilon}^{s})$$

where $\boldsymbol{\epsilon}^{s}$ is drawn from for example a standard Gaussian distribution.

Bayes by BackProp (BbB)

- The estimated ELBO now

$$\hat{\mathcal{L}}(\lambda) = \frac{1}{S}\sum_{s=1}^{S}\left[\ln p\left(\boldsymbol{w}(\lambda, \boldsymbol{\epsilon}^s), \mathcal{D}\right) - \ln q\left(\boldsymbol{w}(\lambda, \boldsymbol{\epsilon}^s), \lambda\right)\right]$$

- Now this is a smooth function of λ and can differentiate

$$\nabla_\lambda \hat{\mathcal{L}}(\lambda) = \frac{1}{S}\sum_{s=1}^{S}\left[\frac{\partial \hat{\mathcal{L}}_s}{\partial \boldsymbol{w}}\frac{\partial \boldsymbol{w}}{\partial \lambda} + \frac{\partial \hat{\mathcal{L}}_s}{\partial \lambda}\right]$$

  where $\hat{\mathcal{L}}_s = \ln p\left(\boldsymbol{w}(\lambda, \boldsymbol{\epsilon}^s), \mathcal{D}\right) - \ln q\left(\boldsymbol{w}(\lambda, \boldsymbol{\epsilon}^s), \lambda\right)$
- Once the gradients are known, optimum $\lambda^\star$ and hence $q(\boldsymbol{w}, \lambda^\star)$ can be found by gradient descent.

Performance of BBVI and BbB

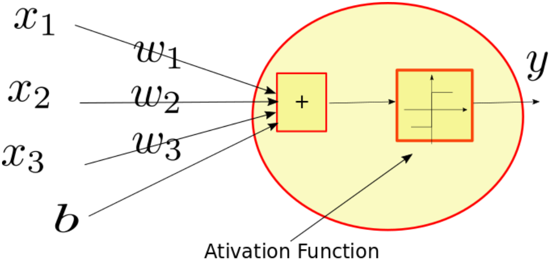
- Both methods estimate approximate gradients by sampling
- High variance of the estimated gradients is a problem
- In practice, these algorithms need modifications to tackle high variance
- BbB tends to have a lower variance estimates than BBVI

Bayesian Deep Learning through Randomization in Training

- Stochastic gradient descent and Dropout can be given Bayesian interpretations
- Dropout procedure in testing can be used for estimating the uncertainty of model outputs (Monte Carlo Dropout).
  - Enable dropout and feed the network S times with data and collect the outputs $f(s)$, $s = 1, 2, \cdots, S$
  - Output variance $= \frac{1}{S} \sum_s (f(s) - \bar{f}(s))^2$ where $\bar{f}(s) = \frac{1}{S} \sum_s f(s)$

Restricted Boltzmann Machines

Stochastic Neurons



Deterministic Neuron $y = \sigma(b + \sum_i w_i x_i)$

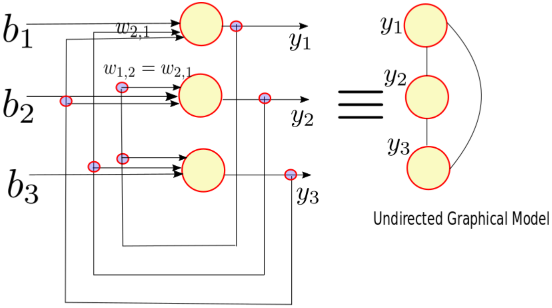
Stochastic Neuron $p(y = 1) = \sigma(b + \sum_i w_i x_i)$

- We consider stochastic binary neurons, i.e. y can be either 1 or 0

$$p(y = 1) = \sigma(b + \sum_i w_i x_i)$$

$$p(y = 0) = 1 - p(y = 1)$$

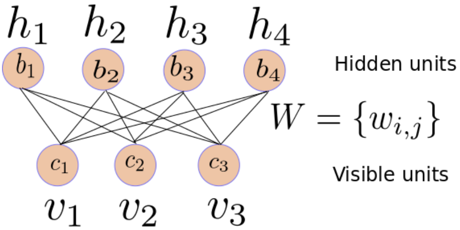
Boltzmann Machine



Stochastic Recurrent Neural Network

- A Boltzmann machine is a recurrent network with stochastic neurons
- Weights are symmetrical
- At the equilibrium, the relationships of the neuron outputs can be represented using an undirected graphical model

Restricted Boltzmann Machine (RBM)



- Neurons are divided into two groups: Visible and Hidden
- Restricted architecture: No connections within visible group or hidden group
- Network parameters:
  - Bias vector hidden units, $\boldsymbol{b} = [b_1, b_2, \cdots, b_H]$
  - Bias vector visible units, $\boldsymbol{c} = [c_1, c_2, \cdots, c_V]$
  - Connection weights, $W = \{w_{i,j}\}$
- Network values are binary random vectors: $\boldsymbol{v} = [v_1, v_2, \cdots, v_V]$ and $\boldsymbol{h} = [h_1, h_2, \cdots, h_H]$

How the network parameters and values are related?

- Through the definition of an Energy function
- In RBM, the energy function is defined as

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{h}^T \boldsymbol{W} \boldsymbol{v} - \boldsymbol{c}^T \boldsymbol{v} - \boldsymbol{b}^T \boldsymbol{h}$$

- We assign probabilities to $(\boldsymbol{v}, \boldsymbol{h})$ based on Boltzmann distribution

$$p(\boldsymbol{v}, \boldsymbol{h}) = \frac{\exp(-E(\boldsymbol{v}, \boldsymbol{h}))}{Z}$$

where

$$Z = \sum_{\boldsymbol{v}', \boldsymbol{h}'} \exp(-E(\boldsymbol{v}', \boldsymbol{h}'))$$

What can we do with RBM?

Assume that the network parameters $\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c}$ are known.

- Can we calculate the probability of a given pair of vectors $(\hat{\boldsymbol{v}}, \hat{\boldsymbol{h}})$?
  - This is generally not tractable, because calculating Z requires to sum all combinations v and h values.
- Can we calculate the probability of $\boldsymbol{h}$ given $\boldsymbol{v}$ or vice-versa?
  - Yes, this is "inference" and possible.

Assume that a data set of $\boldsymbol{v}$ vectors given.

- Can we estimate the network parameters $\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c}$?
  - Yes, this is training and possible

Inference

We want to find $p(\boldsymbol{h}|\boldsymbol{v})$ assuming $\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c}$ are known.

- We start with the Bayes rule

$$
\begin{aligned}
p(\boldsymbol{h}|\boldsymbol{v}) &= \frac{p(\boldsymbol{h}|\boldsymbol{v})}{\sum_{\boldsymbol{h}'} p(\boldsymbol{h}', \boldsymbol{v}')} \\
&= \frac{\exp\left(\boldsymbol{h}^T \boldsymbol{W} \boldsymbol{v} + \boldsymbol{c}^T \boldsymbol{v} + \boldsymbol{b}^T \boldsymbol{h}\right)/Z}{\sum_{\boldsymbol{h}' \in \{0,1\}^H} \exp\left(\boldsymbol{h}'^T \boldsymbol{W} \boldsymbol{v}' + \boldsymbol{c}^T \boldsymbol{v}' + \boldsymbol{b}^T \boldsymbol{h}'\right)/Z}
\end{aligned}
$$

- Canceling common factors and expanding vector-matrix multiplication as a summation

$$
p(\boldsymbol{h}|\boldsymbol{v}) = \frac{\exp\left(\sum_j \left(h_j \boldsymbol{W}_j \boldsymbol{v} + b_j hj\right)\right)}{\sum_{h'_1 \in \{0,1\}} \sum_{h'_2 \in \{0,1\}} \cdots \sum_{h'_H \in \{0,1\}} \exp(\sum_j (h'_j \boldsymbol{W}_j \boldsymbol{v} + b_j h'_j))}
$$

Inference

We want to find $p(\boldsymbol{h}|\boldsymbol{v})$ assuming $\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c}$ are known.

- Writing exponential of sums as product of exponentials

$$
\begin{aligned}
p(\boldsymbol{h}|\boldsymbol{v}) &= \frac{\prod_j \left(\exp\left(h_j \boldsymbol{W}_j \boldsymbol{v} + b_j hj\right)\right)}{\sum_{h'_1 \in \{0,1\}} \sum_{h'_2 \in \{0,1\}} \cdots \sum_{h'_H \in \{0,1\}} \prod_j (\exp(h'_j \boldsymbol{W}_j \boldsymbol{v} + b_j h'_j))} \\
&= \frac{\prod_j \left(\exp\left(h_j \boldsymbol{W}_j \boldsymbol{v} + b_j hj\right)\right)}{(\sum_{h'_1 \in \{0,1\}} \exp(h'_1 \boldsymbol{W}_1 \boldsymbol{v} + b_1 h'_1)) \ldots (\sum_{h'_H \in \{0,1\}} \exp(h'_H \boldsymbol{W}_H \boldsymbol{v} + b_H h'_H))} \\
&= \frac{\prod_j \left(\exp\left(h_j \boldsymbol{W}_j \boldsymbol{v} + b_j hj\right)\right)}{\prod_j (\sum_{h'_j \in \{0,1\}} \exp(h'_j \boldsymbol{W}_j \boldsymbol{v} + b_j h'_j))} \\
&= \frac{\prod_j \left(\exp\left(h_j \boldsymbol{W}_j \boldsymbol{v} + b_j hj\right)\right)}{\prod_j \exp(\boldsymbol{W}_j \boldsymbol{v} + b_j)} \\
&= \prod_j \frac{\left(\exp\left(h_j \boldsymbol{W}_j \boldsymbol{v} + b_j hj\right)\right)}{\exp(\boldsymbol{W}_j \boldsymbol{v} + b_j)}
\end{aligned}
$$

- This implies that calculation of $p(\boldsymbol{h}|\boldsymbol{v})$ is tractable

Inference

- Let's try to interpret

$$p(\boldsymbol{h}|\boldsymbol{v}) = \prod_j \frac{(\exp{(h_j \boldsymbol{W}_j \boldsymbol{v} + b_j hj))}}{\exp(\boldsymbol{W}_j \boldsymbol{v} + b_j)}$$

- Consider the quantity $q(h_j) = \dfrac{\exp{(h_j \boldsymbol{W}_j \boldsymbol{v} + b_j hj)}}{\exp(\boldsymbol{W}_j \boldsymbol{v} + b_j)}$
- $q(h_j)$ takes two values, $q(0)$ and $q(= 1)$. And sum of these values are 1. Therefore it is a probability measure of h_j.
- Since we assumed $\boldsymbol{v}$ is given, $q(h_j)$ is actually $p(h_j|\boldsymbol{v})$
- A simple manipulation shows that $\boxed{p(h_j = 1) = \sigma(\boldsymbol{W}_j \boldsymbol{v} + b_j)}$ i.e. The activation function of a stochastic neuron.

Training

- We consider maximum likelihood training with a given dataset $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_N\}$ with respect to the log likelihood $L = \log \prod_i^N p(\boldsymbol{v}_i) = \sum_i^N \log p(\boldsymbol{v}_i)$
- We use gradient descent and therefore calculate $\dfrac{\partial L}{\partial \theta}$ the gradient of L with respect to a model parameter θ
- Derive the gradient for a single sample $\dfrac{\partial \log \left(p\left(\boldsymbol{v}\right)\right)}{\partial \theta}$

Gradients

- By definition we know that

$$p(\boldsymbol{v}, \boldsymbol{h}) = \frac{\exp(-E(\boldsymbol{v}, \boldsymbol{h}))}{Z} \tag{1}$$

where

$$Z = \sum_{\boldsymbol{v}', \boldsymbol{h}'} \exp(-E(\boldsymbol{v}', \boldsymbol{h}')) \tag{2}$$

- Therefore

$$p(\boldsymbol{v}) = \sum_{\boldsymbol{h}} p(\boldsymbol{v}, \boldsymbol{h}) = \sum_{\boldsymbol{h}} \frac{\exp(-E(\boldsymbol{v}, \boldsymbol{h}))}{Z} \tag{3}$$

- Take log and differentiate wrt θ

$$\frac{\partial \log p(\boldsymbol{v})}{\partial \theta} = \frac{\partial \log \sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}))}{\partial \theta} - \frac{\partial \log Z}{\partial \theta} \tag{4}$$

Gradients

- Consider the first term

$$\frac{\partial \log \sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h}))}{\partial\theta} = -\frac{\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h}))\frac{\partial(E(\boldsymbol{v},\boldsymbol{h}))}{\partial\theta}}{\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h}))} \tag{5}$$

$$= -\sum_{\boldsymbol{h}} \frac{\exp(-E(\boldsymbol{v},\boldsymbol{h}))}{\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h}))}\frac{\partial(E(\boldsymbol{v},\boldsymbol{h}))}{\partial\theta} \tag{6}$$

- But dividing equation 1 by equation 3 we get

$$\frac{p(\boldsymbol{v},\boldsymbol{h})}{p(\boldsymbol{v})} = p(\boldsymbol{h}|\boldsymbol{v}) = \frac{\exp(-E(\boldsymbol{v},\boldsymbol{h}))}{\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h}))} \tag{7}$$

- Substitute equation 7 in equation 6

$$\frac{\partial \log \sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h}))}{\partial\theta} = -\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v})\frac{\partial(E(\boldsymbol{v},\boldsymbol{h}))}{\partial\theta} \tag{8}$$

Gradients

- Consider the second term in equation 4 and substitute for Z from equation 2

$$\frac{\partial \log Z}{\partial \theta} = \frac{\partial \log \sum_{\boldsymbol{v}', \boldsymbol{h}'} \exp(-E(\boldsymbol{v}', \boldsymbol{h}'))}{\partial \theta} \tag{9}$$

$$= -\frac{\sum_{\boldsymbol{v}', \boldsymbol{h}'} \exp(-E(\boldsymbol{v}', \boldsymbol{h}')) \frac{\partial (E(\boldsymbol{v}', \boldsymbol{h}'))}{\partial \theta}}{\sum_{\boldsymbol{v}', \boldsymbol{h}'} \exp(-E(\boldsymbol{v}', \boldsymbol{h}'))} \tag{10}$$

$$= -\sum_{\boldsymbol{v}, \boldsymbol{h}} \frac{\exp(-E(\boldsymbol{v}, \boldsymbol{h}))}{\sum_{\boldsymbol{v}', \boldsymbol{h}'} \exp(-E(\boldsymbol{v}', \boldsymbol{h}'))} \frac{\partial (E(\boldsymbol{v}, \boldsymbol{h}))}{\partial \theta} \tag{11}$$

- From equations 1 and 2 it is clear that $\frac{\exp(-E(\boldsymbol{v}, \boldsymbol{h}))}{\sum_{\boldsymbol{v}', \boldsymbol{h}'} \exp(-E(\boldsymbol{v}', \boldsymbol{h}'))}$ is $p(\boldsymbol{v}, \boldsymbol{h})$
- Therefore

$$\frac{\partial \log Z}{\partial \theta} = -\sum_{\boldsymbol{v}, \boldsymbol{h}} p(\boldsymbol{v}, \boldsymbol{h}) \frac{\partial (E(\boldsymbol{v}, \boldsymbol{h}))}{\partial \theta} \tag{12}$$

Gradients

- From equations 4, 8 and 12

$$\frac{\partial \log p(\boldsymbol{v})}{\partial \theta} = -\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) \frac{\partial (E(\boldsymbol{v},\boldsymbol{h}))}{\partial \theta} + \sum_{\boldsymbol{v},\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h}) \frac{\partial (E(\boldsymbol{v},\boldsymbol{h}))}{\partial \theta} \quad (13)$$

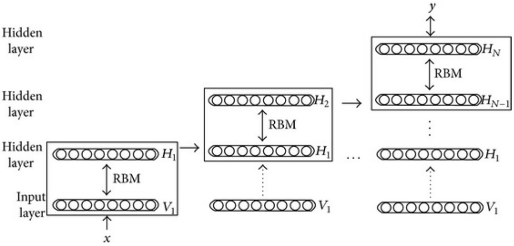
$$\boxed{\frac{\partial \log p(\boldsymbol{v})}{\partial \theta} = -\mathbb{E}_{p(\boldsymbol{h}|\boldsymbol{v})}\left[\frac{\partial (E(\boldsymbol{v},\boldsymbol{h}))}{\partial \theta}\right] + \mathbb{E}_{p(\boldsymbol{v},\boldsymbol{h})}\left[\frac{\partial (E(\boldsymbol{v},\boldsymbol{h}))}{\partial \theta}\right]} \quad (14)$$

- The first term of equation 14
  - Known as positive phase
  - Depends on training data
  - Can be computed exactly
- The second term of equation 14
  - Known as negative phase
  - independent of training data, completely model dependent
  - Must be estimated through Gibb's sampling and a procedure known as Contrastive Divergence

Applications of RBMs

- Deep belief networks
- Collaborative filtering

Deep Belief Networks



Method for initializing a multilayer network

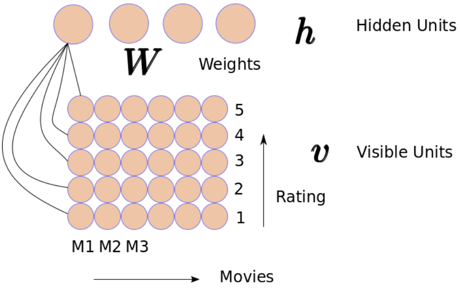
1. Train an RBM with training data
2. Initialize the current layer with the trained parameters
3. Present training data to the RBM and sample the hidden layer values
4. Use the hidden layer values as training data and repeat from step 1.

Collaborative Filtering

	M1	M2	M3	M4	M5	M6
U1				3		
U2	5		1			
U3		3	5			
U4	4		?			5
U5			4			
U6					2	

- Application in recommendation systems. Eg: Movie rating/recommendation
- Different users rate different items (eg: movies) using a rating scale such as 1 to 5
- Problem is to estimate the rating for an unrated item by a given user

Collaborative filtering with RBM



- Train a different RBM for each user. But share weights across users
- Visible units correspond to the ratings given to each movie
- In training movies with missing ratings are omitted
- For prediction of a missing rating, find $p(\boldsymbol{h}|\boldsymbol{v})$ and back to $p(v|\boldsymbol{h})$

The End