Bayesian Deep Learning (contd), (Shallow and Deep) Generative Models

CS772A: Probabilistic Machine Learning
Piyush Rai
(Deep) Neural Networks

- These are nonlinear function approximators
- Consists of an input layer, one or more hidden layers, and an output layer

Hidden layers act as feature extractors

Can think of the last hidden layer’s node values being used as features in a GLM (linear/logistic/softmax, etc) modeled by the output layer

Network weights typically learned by backpropagation (basically, gradient descent + chain rule)
Bayesian Neural Networks

- Backprop for neural nets only gives us point estimates for the weights
- Another alternative is to be Bayesian and learn the posterior distribution over weights

Standard neural net: Each weight has a fixed value, learned by backprop

Bayesian neural net: Each weight has a posterior distribution inferred by some Bayesian inference algo (VI/MCMC/Laplace approx., etc)

Note: Just having a likelihood and prior will still give us a standard neural net if we choose to do MLE/MAP only

Also, test time will require computing PPD, not just a plug-in prediction

Using reparametrization trick (known as "Bayes by Backprop" in this context), BBVI etc

Pic from: *Weight Uncertainty in Neural Networks (Blundell et al, 2015)
A Hybrid Bayesian Neural Net

- Learning the posterior for all weights can be expensive
- PPD computation is also slow if using Monte Carlo approximation for PPD
- A cheaper practical alternative is
  - Do point estimation for hidden layer weights ($\mathbf{W}$)
  - Infer the full posterior for output layer weights ($\mathbf{V}$)
  - The PPD will then be
    $$ p(y_*|x_*, \mathcal{D}) \approx \frac{1}{S} \sum_{s=1}^{S} p(y_*|x_*, \mathbf{V}^{(s)}, \mathbf{W}) $$
    where $\mathbf{V}^{(s)} \sim p(\mathbf{V}|\mathcal{D})$
  - Faster because the posterior of $\mathbf{V}$ is much lower dimensional
  - A rough approximation of the above is the following
    - Use a pretrained neural net to extract feature
    - Train Bayesian linear model (e.g., Bayesian linear/logistic/softmax/GLM reg.) on these features

A rough approximation of the above is the following

$$ p(y_*|x_*, \mathcal{D}) \approx \frac{1}{S} \sum_{s=1}^{S} p(y_*|x_*, \theta^{(s)}) $$
where $\theta^{(s)} \sim p(\theta|\mathcal{D})$
Bayesian Neural Networks: The Priors

- Zero-mean isotropic Gaussian priors are common and convenient
  - Corresponds to weight-decay or $\ell_2$ regularizer

- Another alternative is to use sparsity-inducing priors, e.g.,

$$p(w) = \prod_j \pi N(w_j | 0, \sigma_1^2) + (1 - \pi) N(w_j | 0, \sigma_2^2) \quad \sigma_1 > \sigma_2 \text{ and } \sigma_2 \ll 1$$

- Gaussian priors have been found somewhat problematic in recent work
  - Cold-posterior effect

$T$ is like temperature

$T = 1$ is the standard Bayesian inference

Recent work has shown that BNNs with standard Gaussian priors work poorly for $T = 1$ but $T \ll 1$ improves performance

Maybe Gaussian priors aren’t really ideal??

Pic from: *How Good is the Bayes Posterior in Deep Neural Networks Really? (Wenzel et al, 2020)*
Other Inference Methods for Bayesian Neural Nets

- Laplace approximation is very common: \( p(W|D) \approx \mathcal{N}(W_{\text{MAP}}, H^{-1}) \)
  - However, can be slow since the number of parameters is very large
  - One option is to use a simpler covariance matrix (e.g., diagonal or block-diag)
  - Another option is to use the hybrid Bayesian neural net
    - Use MAP estimates for the hidden layer weights
    - Use Laplace approximation only for the output layer weights

- Using SGD iterates obtained from backprop

\[ p(w|D) \approx q(w|D) = \mathcal{N}(\tilde{w}, K) \]

\[ \tilde{w} = \frac{1}{T} \sum_t w_t, \quad K = \frac{1}{2} \left( \frac{1}{T-1} \sum_t (w_t - \tilde{w})(w_t - \tilde{w})^T + \frac{1}{T-1} \sum_t \text{diag}(w_t - \tilde{w})^2 \right) \]

Extension: A mixture of Gaussian approximation: Multi-SWAG – Run SGD \( M \) times and use a mixture of \( M \) such Gaussians

SWA based Gaussian approximation: SWAG

Pic from: *A Simple Baseline for Bayesian Uncertainty in Deep Learning (Maddox et al, 2019)*
Other Inference Methods for Bayesian Neural Nets

- **Monte Carlo Dropout** is another popular and efficient way
- **Standard Dropout**
  - Drop some weights randomly (with some “drop” probability) during training
  - At test time, multiply each weight by the “keep” probability
  - Note: Dropout applied only at training time

- **Monte Carlo Dropout**

\[
p(y_\star | x_\star, \mathcal{D}) \approx \frac{1}{S} \sum_{s=1}^{S} p(y_\star | x_\star, \theta^{(s)})
\]

where \( \theta^{(s)} \sim p(\theta | \mathcal{D}) \)

\[
p(y_\star | x_\star, \mathcal{D}) \approx \frac{1}{S} \sum_{s=1}^{S} p(y_\star | x_\star, \theta^{(s)})
\]

where \( \theta^{(s)} = \epsilon^{(s)} \odot \hat{\theta} \)

Can be seen as learning a variational approximation of the weights (see paper for details, if interested)

*Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning (Gal and Ghahramani, 2016)*
Other Inference Methods for Bayesian Neural Nets

- SGMCMC methods like SGLD and SGHMC are also used nowadays (very efficient)

\[
\theta(t) = \theta(t-1) + \frac{\eta t}{2} \nabla_{\theta} \left[ \log p(D|\theta) + \log p(\theta) \right] \bigg|_{\theta(t-1)} + \epsilon_t
\]

- Recently, SGMCMC with cyclic step sizes (cSGLD) was proposed (Zhang et al, 2020)
  - Use big steps to explore different modes
  - Use small steps later to sample once a mode is localized

Pic from: *Cyclical Stochastic Gradient MCMC for Bayesian Deep Learning (Zhang et al, 2020)*

Step size in iteration \(k\)

\(K\) is the total number of iterations and \(M\) is the number of cycles

A complex mixture of Gaussian distributions

<table>
<thead>
<tr>
<th>Method</th>
<th>CIFAR-10</th>
<th>CIFAR-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGD</td>
<td>5.29±0.15</td>
<td>23.61±0.09</td>
</tr>
<tr>
<td>SGDM</td>
<td>5.17±0.09</td>
<td>22.98±0.27</td>
</tr>
<tr>
<td>Snapshot-SGD</td>
<td>4.46±0.04</td>
<td>20.83±0.01</td>
</tr>
<tr>
<td>Snapshot-SGDM</td>
<td>4.39±0.01</td>
<td>20.81±0.10</td>
</tr>
<tr>
<td>SGLD</td>
<td>5.20±0.06</td>
<td>23.23±0.01</td>
</tr>
<tr>
<td>cSGLD</td>
<td>4.29±0.06</td>
<td>20.55±0.06</td>
</tr>
<tr>
<td>SGHMC</td>
<td>4.93±0.10</td>
<td>22.60±0.17</td>
</tr>
<tr>
<td>cSGHMC</td>
<td><strong>4.27±0.03</strong></td>
<td><strong>20.50±0.11</strong></td>
</tr>
</tbody>
</table>

Pic from: *Cyclical Stochastic Gradient MCMC for Bayesian Deep Learning (Zhang et al, 2020)*
Deep Ensembles

- Most inference methods tend to produce local approximations only
  - VI methods typically learn an approximation around one of the modes
  - Sampling methods may give most samples near one of the modes (though in principle they may explore other modes as well)
  - Thus the uncertainties may be underestimated in general

- Deep Ensembles* is a method that tries to address this issue
  - Train the network $M$ times with different seeds and permutations of training data
  - Denote the learned weights by $\theta_1, \theta_2, \ldots, \theta_M$ (assuming these are $M$ modes)
  - Approximate the posterior by the following

  $$ p(\theta | \mathcal{D}) = \frac{1}{M} \sum_{m=1}^{M} \delta_{\theta_m}(\theta) $$

  - This approach is considered non-Bayesian but often performs better (in terms of more diversity in the set of parameters learned) than other inference methods

*Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles (Lakshminarayanan, 2017)
Deep Generative Models
(for unsupervised learning)
Many generative models for unsupervised learning have this form

\[ p(z) \rightarrow p(x | f(z)) \]

Depending on the prior, likelihood, and \( f \), various latent factor models arise, e.g.,

- Factor Analysis and Probabilistic PCA: \( p(x | f(z)) = N(x | Wz, \Sigma) \)
- Gaussian Process Latent Variable Models (GPLVM) – \( f \) is nonlinear modeled by a GP
- Deep generative models (constructed using deep neural nets)
  - Variational Autoencoders (VAE) - \( f \) is nonlinear modeled by a neural net
  - Generative Adversarial Network (GAN) – \( f \) is nonlinear modeled by a neural net and the likelihood is only implicitly defined
  - Denoising Diffusion Models
  - .. and several others..
Some Classical Models
Factor Analysis and Probabilistic PCA

- Assumption: Latent variables $\mathbf{z}_n \in \mathbb{R}^K$ typically assumed to have a Gaussian prior
  - If we want sparse latent variable, can use Laplace or spike-and-slab prior on $\mathbf{z}_n$
  - More complex extensions of FA/PPCA use a mixture of Gaussians prior on $\mathbf{z}_n$

- Assumption: Observations $\mathbf{x}_n \in \mathbb{R}^D$ typically assumed to have a Gaussian likelihood
  - Other likelihood models (e.g., exp-family) can also be used if data not real-valued

- Relationship between $\mathbf{z}_n$ and $\mathbf{x}_n$ modeled by a noisy linear mapping

\[
\mathbf{x}_n = \mathbf{W} \mathbf{z}_n + \mathbf{\epsilon}_n = \sum_{k=1}^{K} \mathbf{w}_k \mathbf{z}_{nk} + \mathbf{\epsilon}_n
\]

- Unknowns $\mathbf{W}$, $\mathbf{z}_n$’s, and $\Psi$ can be learned
  - EM, VI, MCMC

- Gaussian prior $p(\mathbf{z}_n) = \mathcal{N}(\mathbf{z}_n | 0, \mathbf{I})$
- Gaussian likelihood $p(\mathbf{x}_n | \mathbf{z}_n) = \mathcal{N}(\mathbf{x}_n | \mathbf{W} \mathbf{z}_n, \Psi)$

Zero-mean and diagonal or spherical Gaussian noise
Linear combination of the columns of $\mathbf{W}$
Diagonal for FA, spherical for PPCA
Some Other Classical Models

- **Gamma-Poisson latent factor model**
  - Assumes $K$-dim non-negative latent variable $\mathbf{z}_n$ and $D$-dim count-valued observations $\mathbf{x}_n$
  - An example: Each $\mathbf{x}_n$ is the word-count vector representing a document
    
    \[
    p(\mathbf{z}_n) = \prod_{k=1}^{K} \text{Gamma}(z_{nk}|a_k, b_k)) \\
    p(\mathbf{x}_n|\mathbf{z}_n) = \prod_{d=1}^{D} \text{Poisson}(x_{nd}|f(\mathbf{w}_d, \mathbf{z}_n))
    \]
  - This can be thought of as a probabilistic non-negative matrix factorization model

- **Dirichlet-Multinomial/Multinoulli PCA**
  - Assumes $K$-dim non-negative latent variable $\mathbf{z}_n$ and $D$ categorical obs $\mathbf{x}_n = \{x_{nd}\}_{d=1}^{D}$
  - An example: Each $\mathbf{x}_n$ is a document with $D$ words in it (each word is a categorical value)
    
    \[
    p(\mathbf{z}_n) = \text{Dirichlet}(\mathbf{z}_n|\boldsymbol{\alpha}) \\
    p(\mathbf{x}_n|\mathbf{z}_n) = \prod_{d=1}^{D} \text{Multinoulli}(x_{nd}|f(\mathbf{w}_d, \mathbf{z}_n))
    \]
  - Also sums to 1

Non-negative priors often give a nice interpretability to such latent variable models (will see some more examples of such models shortly)
Assume $D$ documents, and document $d$ has $N_d$ words in it.

We can represent doc $d$ by a word count vector $w_d$.

Assuming a vocab of $V$ unique words, $w_d$ is a $V \times 1$ vector of counts.

- $w_{dv} = \text{no of times word } v \text{ appears in doc } d$

Let's model the docs by a mixture of $K$ multinomial distributions, each $V$-dim.

- The $k^{th}$ multinomial modeled by a $V$-dim prob vector $\phi_k$ (sums to 1)
- $\phi_k$ can be thought of as a “topic vector” (or just “topic”), $\phi_{kv}$: prob of word $v$ in topic $k$

Generative model and plate diagram below.

$$ z_d \sim \text{multinoulli}(\pi) $$

$$ w_d \sim \text{multinomial}(\phi_{z_d}, N_d) $$

Counts will sum to $N_d$.

Each topic is a prob. distribution over word tokens.

Limitation: Each doc $d$ belongs to a single cluster $z_d$ and all words in a document assumed to be from the same topic. This is unrealistic/restrictive.
Documents can be about multiple topics

Seeking Life’s Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—

How many genes does an organism need to

survive? Last week at the genome meeting

here, two genome researchers with radically
different approaches presented complementar-

ary views of the basic genes needed for life.

One research team, using computer analy-

ses to compare known genomes, concluded

that today’s organisms can be sustained with

just 250 genes, and that the earliest life forms

required a mere 128 genes. The

other researcher mapped genes in a simple parasite and estimated that for this organism,

800 genes are plenty to do the job—but that anything short of 100 wouldn’t be enough.

Although the numbers don’t match precisely, those predictions

“are not all that far apart,” especially in

comparison to the 75,000 genes in the hu-

man genome, notes Siv Andersson of Uppsala

University in Sweden, who arrived at the 800

number. But coming up with a consen-

sus answer may be more than just a genetic

numbers game, particularly as more and

more genomes are completely mapped and

sequenced. “It may be a way of organizing

any newly sequenced genome,” explains

Arcady Mushegian, a computational mol-

ecular biologist at the National Center

for Biotechnology Information (NCBI) in

Bethesda, Maryland. Comparing an


Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

256 genes

Reducant and parasite-specific genes removed ~4 genes

Minimal gene set 250 genes

Related and modern genes removed ~122 genes

Ancient gene set 128 genes

Genes in common 230 genes

Mycoplasma genome 469 genes

Haemophilus genome 1708 genes

ADAPTED FROM NCBI

How do we find the word-topic associations in each document?

How do we use them to learn topics in the given text collection?

How do we learn low-dim document representations in terms of the topics they represent?
Assume a corpus-level topic mixing proportions $\alpha$ ($K \times 1$ prob vector)

Also assume doc-level topic mixing props $\theta_d$ ($K \times 1$ prob vector)

Instead of assuming a single cluster $z_d$ for doc $d$, cluster each word in it

- $z_{d,n} \in \{1,2,\ldots,K\}$ denotes the cluster/topic of word $w_{d,n} \in \{1,2,\ldots,V\}$

Can obtain the “average” clustering for doc $d$ using $\theta_d$ or $\bar{z}_d = \frac{1}{N_d} \sum_{n=1}^{N_d} z_{d,n}$

The generative model is as follows

\[
\phi_k \sim \text{Dir}(\eta) \quad k = 1,2,\ldots,K \\
\theta_d \sim \text{Dir}(\alpha) \quad d = 1,2,\ldots,D
\]

- $\phi_k \sim \text{Dir}(\eta)$ ($V$-dim Dirichlet)
- $\theta_d \sim \text{Dir}(\alpha)$ ($K$-dim Dirichlet)

- $z_{d,n} \sim \text{multinoulli}(\theta_d)$
- $w_{d,n} \sim \text{multinoulli}(\phi_{z_{d,n}})$

Locally-conjugate. Easy Gibbs sampling, VI, etc

Somewhat similar to Dir-Mult PCA model

Each assumed a one-hot $K \times 1$ vector

Latent Dirichlet Allocation* (LDA) Topic Model
Latent Dirichlet Allocation (LDA)

- A very widely used probabilistic model for text data
- Nice and easy insights into the text collection
  - Each $\phi_k = [\phi_{k1}, \ldots, \phi_{kV}]$ can be interpreted as topic ($\phi_{kv} = \text{prob. of word } v \text{ in topic } k$)
  - $\theta_d = [\theta_{d1}, \ldots, \theta_{dK}]$: how much each topic is present in document $d$ (topic distribution)
  - $\bar{z}_d = \frac{1}{N_d} \sum_{n=1}^{N_d} z_{d,n}$ also has a similar interpretation as $\theta_d$

A topic is a set of words that tend to co-occur together

15 most frequent (most probable) words from four most prominent topics in this doc

Topic distribution for the document on left
LDA: Inference and Evaluation

- LDA is locally conjugate. Many inference methods (VI, variational EM, Gibbs samp, etc)
  
  \[ p(Z, \Theta, \Phi|W, \alpha, \eta) = \frac{p(W|\Phi, Z)p(Z|\Theta)p(\Phi|\eta)p(\Theta|\alpha)}{p(W|\alpha, \eta)} \]  
  (assuming hyperparams \( \alpha, \eta \) are fixed)

- Can even collapse some variables and do collapsed Gibbs or collapsed VB
  - E.g., collapse \( \theta_d \) and \( \phi_k \) (if needed, these can be approximated using \( Z \))

- Many ways to evaluate how well LDA performs on some data
  - Extrinsic measures: Perform LDA and use its output for another task (e.g., classification)
  - Perplexity is another intrinsic measure to evaluate LDA-style models

\[ \text{perplexity}(D_{test}) = \exp \left\{ -\frac{1}{M} \sum_{d=1}^{M} \log p(w_d) \right\} \]  
  Marginal likelihood of all words in the \( d^{th} \) test doc
  Test set with \( M \) docs
  Lower is better
LDA: Limitations and Extensions

- LDA assumes topics remain static over time (improvement: Dynamic Topic Model)

  Assume a first-order Markov evolution for each topic w.r.t. time

  \[ w^t_k \sim \mathcal{N}(w^{t-1}_k, \sigma^2 I) \quad \phi^t_k = S(w^t_k) \]

  Simplex transformation (convert \( w^t_k \) into a probability vector)

- LDA assumes topics are uncorrelated (improvement: Corr-LDA)
  - Use a logistic normal distribution on \( \theta_d \) (cov matrix of log-normal makes component correlated)

- LDA ignores the sequential structure in the text (improvement: HMM-LDA)

Evolution of topic “Neuroscience” (learned from the journal Science)

Fig courtesy: Dynamic Topic Models (Blei and Lafferty, 2006)
LDA Extensions (Contd)

- LDA for non-text data, e.g., images
  - Each image can be represented as a bag of "visual words" and LDA can be applied
- Supervised/Labeled LDA (when we have a label for each document)
- LDA for paired/multimodality data (e.g., images and text caption)
- LDA for graph-structured data instead of documents

Plate diagrams for some LDA extensions

LDA is also equivalent to doing a non-negative matrix fact. of the $V \times D$ word-document matrix $X$ using a Poisson likelihood model*

$$X \sim \text{Poisson}(\Phi \Theta)$$

$\Phi$ ($V \times K$) and $\Theta$ ($K \times D$) can be given any non-negative priors (Dirichlet/gamma)

This can be extended to “deep” matrix factorization** (modeling $\Theta$ using many layers)

*Sec 4 and 5 of “Beta-Negative Binomial Process and Poisson Factor Analysis” (Zhou et al, 2012)

** Poisson-gamma belief networks* (Zhou et al, 2015)
Next Class

- Generative models using deep neural networks
  - Variational Autoencoders
  - Generative Adversarial Networks
  - Denoising Diffusion Models