Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference

Yeseul Jeon

March 1, 2021

Contents

About paper:

Under review as a conference paper at ICLR 2016

Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference

Yarin Gal & Zoubin Ghahramani
University of Cambridge
{yg279,zg201}@cam.ac.uk

Abstract

- **Bayesian NN**
  - CNNs require huge amounts of data for regularisation and quickly over-fit on small data
  - In contrasts Bayesian Neural Networks are robust to over-fitting, offer uncertainty estimates and can easily learn from small datasets.
  - Inferring model posterior in a Bayesian NN is a difficult task.
  - Therefore, using a simple variational distribution such as a Gaussian is required.

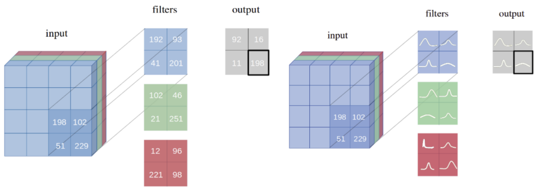
Abstract

- **MC Dropout to CNN**

  - However, the variational approach used to approximate the posterior in Bayesian NNs can be computationally expensive.
  - To prevent this computational issue, dropout as a bayesian approach has been suggested
  - However, it is questionable to apply dropout as a bayesian approximation to CNNS.

Abstract

- **Idea:**
  - Previous study have shown that dropout in NNs can be interpreted as an approximation to a well known Bayesian model.
  - 1) Dropout networks' training can be cast as approximate **Bernoulli variational inference** in Bayesian NNs
  - 2) This allows us to use operations such as convolution and pooling in probabilistic models in a principled way.

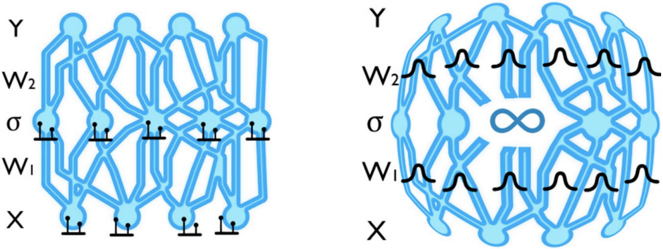


Abstract

- **Benefits:**
  - In existing literature, dropout is often not used after convolution layers.
  - This is because test error suffers, which renders small dataset modelling a difficult task.
  1) This problem can be alleviated by interleaving Bayesian techniques into deep learning.
  2) There is no additional computational cost during training.
  3) This model reduces over-fitting on small datasets compared to standard approach.
  4) It improves model results and estimates a predictive distribution.

Dropout as a Bayesian Approximation

- **Dropout vs Bayesian NN**



- Dropout: Remove random nodes with a probability p
- Bayesian NN: Update the posterior distribution of the weights

Approximation of full Bayesian learning

- **Posterior Distribution**

$$p(\omega\,|\mathrm{X},\mathrm{Y}) = \frac{\overbrace{p(\mathrm{Y}|\omega,\mathrm{X})}^{\text{likelihood}} \cdot \overbrace{p(\omega)}^{\text{prior}}}{\underbrace{p(\mathrm{Y}|\mathrm{X})}_{\text{normalizer=marginal likelihood}}}$$

- We can approximate the posterior distribution for the model parameters via Variation Inference
- replacing the posterior distribution at the observed data $p(w|X,Y)$ with a member $q(w)$ of a simpler distribution family Q that minimizes the Kullback–Leibler divergence to the posterior

Variational Inference Approximated full Bayesian Learning

- **Variational Inference**

$$\mathrm{KL}\left(q_\theta(\omega) \,\|\, \mathrm{p}(\omega \mid \mathbf{X}, \mathbf{Y})\right) = \int q_\theta(\omega) \log \frac{q_\theta(\omega)}{\mathrm{p}(\omega \mid \mathbf{X}, \mathbf{Y})} d\omega = E_q\left[\log\left(q_\theta(\omega)\right) - \log\left(\mathrm{p}(\omega \mid \mathbf{X}, \mathbf{Y})\right)\right]$$

- Approximated $p(w|X, Y)$ with simple distribution $q_\theta(w)$
- Minimize Kullback Leibler divergence of q from the posterior w.r.t to the variational parameters θ:

What kind of q-distribution should we use?

- **The deep Gaussian process**

$$\mathbf{K}(\mathbf{x},\mathbf{y}) = \int p(\mathbf{w})p(b)\sigma(\mathbf{w}^T\mathbf{x}+b)\sigma(\mathbf{w}^T\mathbf{y}+b)\mathbf{d}\mathbf{w}\mathbf{d}b$$

$$\mathbf{w}_k \sim p(\mathbf{w}),\ b_k \sim p(b),$$
$$\mathbf{W}_1 = [\mathbf{w}_k]_{k=1}^K, \mathbf{b} = [b_k]_{k=1}^K$$
$$\widehat{\mathbf{K}}(\mathbf{x},\mathbf{y}) = \frac{1}{K}\sum_{k=1}^K \sigma(\mathbf{w}_k^T\mathbf{x}+b_k)\sigma(\mathbf{w}_k^T\mathbf{y}+b_k)$$
$$\mathbf{F} \mid \mathbf{X}, \mathbf{W}_1, \mathbf{b} \sim \mathcal{N}(\mathbf{0}, \widehat{\mathbf{K}}(\mathbf{X},\mathbf{X}))$$
$$\mathbf{Y} \mid \mathbf{F} \sim \mathcal{N}(\mathbf{F}, \tau^{-1}\mathbf{I}_N),$$

- W_i be a random matrix of dimensions $K_i \times K_{i-1}$ for each layer i.
- A prior let each row of W_i distribute according to the p(w) above.
- Assume vectors m_i of dimensions K_i for each GP layer.

What kind of q-distribution should we use?

- **Prediction Probability of deep GP model**

$$p(\mathbf{y}|\mathbf{x}, \mathbf{X}, \mathbf{Y}) = \int p(\mathbf{y}|\mathbf{x}, \boldsymbol{\omega})p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})\mathrm{d}\boldsymbol{\omega}$$

$$p(\mathbf{y}|\mathbf{x}, \boldsymbol{\omega}) = \mathcal{N}\left(\mathbf{y}; \widehat{\mathbf{y}}(\mathbf{x}, \boldsymbol{\omega}), \tau^{-1}\mathbf{I}_D\right)$$

$$\widehat{\mathbf{y}}\left(\mathbf{x}, \boldsymbol{\omega} = \{\mathbf{W}_1, ..., \mathbf{W}_L\}\right)$$

- The posterior distribution $p(w|X, Y)$ is intractable.
- Use $q(w)$, a distribution over matrices whose columns are randomly set to zero
- To approximated the intractable posterior

Define the structure of the approximate distribution q

- $q_\theta(w)$

$$q_{\mathbf{M}_i}(\mathbf{W}_i) = \mathbf{M}_i \cdot \mathrm{diag}\left(\left\lfloor \mathbf{z}_{i,j} \right\rfloor\right) \qquad \mathbf{M}_i = \mathrm{mean}(\mathbf{W}_i)$$

M_i is as variational parameter of q

$$\mathbf{z}_{i,j} \sim \mathrm{Bernoulli}(\mathrm{p}_i)$$

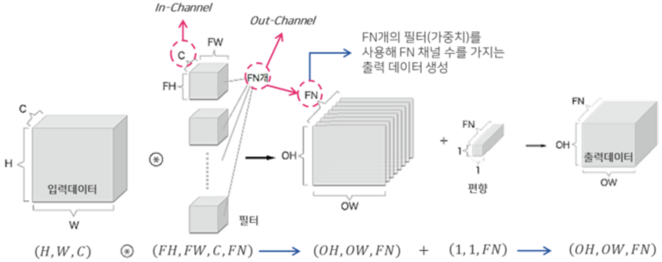
$$\mathbf{W}_i \sim q_{\mathbf{M}_i}(\mathbf{W}_i)$$

Approximate posterior of model parameters

Sampling the diagonal elements **z** from a Bernoulli is identical to randomly setting columns of M to zero which is identical to randomly setting units of the network to zero -> dropout!

- Bernoullis are computationally cheap to get multi-modality

- **Simple Idea**
  - Implementing a Bayesian CNN we apply dropout after all convolution layers as well as inner-product layers.
  - To integrate over the kernels, we reformulate the convolutiona as a linear operation



Bayesian Convolutional Neural Netoworks

- **Simple Idea**

  - Convolving the filters with the input with a given stride is equivalent to extracting patches from the input and performing a inner-product.
  - Extract n patches with $FH \times FW \times C$ dimensional patches from the input with stride s and vectorise these.
  - Collecting the vectors in the rows of a matrix we obtain a new representaiton for our input $x^* = R^{n \times FHFWC}$.
  - The vextorised filter form the columns of the weight matrix $W_i = R^{FHFWC_{i-1} \times C_i}$

- **Simple Idea**

  - The convolution operation is then equivalent to the matrix product $x^* \times W$

  - The columns of the output can be re-arranged to a 3 dimensional tensor $y = R^{H_i \times W_i \times C_i}$

  - Since $n = H_i \times W_i$

Bayesian Convolutional Neural Netoworks

- **Simple Idea**

  - We place a prior distribution over each filter and approximately integrate each filters-patch pair with Bernoulli variational distributions.

  - Sample Bernoulli random variables $Z_{i,j,n}$ and multiply patch n by the weight matrix $W_i \times diag(z_{i,j,n})$ - This distribution randomly sets filters to zero for different patches.

  - This is also equivalent to applying dropout for each element in the tensor y before pooling.

Conclusion

- **Estimate Model uncertainty with MC-dropout**
  - Can represents model uncertainty in deep learning, better model regularisation, computationally efficient Bayesian convolutional neural networks.
  - A neural network with arbitrary depth and non-linearities and with dropout applied before every weight layer is mathematically equivalent to an approximation to the deep Gaussian process (marginalised over its covariance function parameters).