Task-agnostic Continual Learning with Hybrid Probabilistic Models

Polina Kirichenko [1] Mehrdad Farajtabar [2] Dushyant Rao [2]
Balaji Lakshminarayanan [3] Nir Levine [2] Ang Li [2]
Huiyi Hu [2] Andrew Gordon Wilson [1] Razvan Pascanu [2]

[1]New York University

[2]DeepMind

[3]Google Brain

August 15, 2021

Outline

Existing Approaches

- re-sample the data or design specific loss functions that better facilitate learning with imbalanced data
- enhance recognition performance of the tail classes by transferring knowledge from the head classes

Title of

hi

Current Situation

- From a probability theory perspective, it is unjustifiable to use single point-estimates as weights to base any classification on.
- Modelling a distribution over the kernels (also known as filters) of a Cnns has never been attempted successfully before.
- The variational approach used to approximate the posterior in Bayesian NNs can be fairly computationally expensive – the use of Gaussian approximating distributions increases the number of model parameters considerably, without increasing model capacity by much.

Our Hypothesis

we approximate the intractable true posterior probability distributions $p(w|\mathcal{D})$ with variational probability distributions $q_\theta(w|\mathcal{D})$, which comprise the properties of Gaussian distributions $\mu \in \Re^d$ and $\sigma \in \Re^d$, denoted by $\mathcal{N}(\theta|\mu, \sigma^2)$, where d is the total number of parameters defining a probability distribution.

Our Contribution (1)

- Present how *Bayes by Backprop* can be efficiently applied to Cnns.
- Show how the model learns richer representations and predictions from cheap model averaging.
- Show that the proposed variational inference method for Bayesian Cnns can be applied to various Cnn architectures without any limitations on their performances.

Our Contribution (2)

- Examine how to estimate the aleatoric and epistemic uncertainties and empirically show how the uncertainty can decrease, allowing the decisions made by the network to become more deterministic as the training accuracy increases.
- Show how the method typically only doubles the number of parameters yet trains an infinite ensemble using unbiased Monte Carlo estimates of the gradients.
- Apply L1 norm to the trained model parameters and prune the number of non zero values.
- Apply the concept of Bayesian CNN to tasks like Image Super-Resolution and Generative Adversarial.

Neural Networks

- Brain Analogies
- Neural Network
- Convolutional Neural Network

Probabilistic Machine Learning (1)

$$p(y^*|x^*, X, Y) = \int p(y^*|f^*)p(f^*|x^*, X, Y)\,df^* \tag{1}$$

The equation is intractable due to the integration sign. We can approximate it by taking a finite set of random variables w and conditioning the model on it.

$$p(y^*|x^*, X, Y) = \int p(y^*|f^*)p(f^*|x^*, w)p(w|X, Y)\,df^*\,dw$$

Probabilistic Machine Learning (2)

we approximate $p(w|X, Y)$ with a variational distribution $q(w)$. Then

$$q(y^*|x^*) = \int p(y^*|f^*)p(f^*|x^*, w)\,q(w)\,df^*\,dw \tag{2}$$

Minimising the Kullback–Leibler divergence is equivalent to maximising the *log evidence lower bound*,

$$KL_{VI} := \int q(w)p(F|X, w)\log p(Y|F)\,dF dw - KL(q(w)\|p(w)) \tag{3}$$

Uncertainties in Bayesian Learning

- *Aleatoric* uncertainty: measure the noise inherent in the observations. It is present in the data collection method. It cannot be reduced if more data is collected.
- *Epistemic* uncertainty: represent the uncertainty caused by the model. It can be reduced given more data and is often referred to as *model uncertainty*.

Epistemic uncertainty is modelled by placing a prior distribution over a model's weights and then trying to capture how much these weights vary given some data. Aleatoric uncertainty, on the other hand, is modelled by placing a distribution over the output of the model.

Bayes by Backprop (1)

Bayes by Backprop is a variational inference method to learn the posterior distribution on the weights $w \sim q_\theta(w|\mathcal{D})$ of a neural network from which weights w can be sampled in backpropagation.

Bayes by Backprop (2)

An approximate distribution $q_\theta(w|\mathcal{D})$ is defined that is aimed to be as similar as possible to the true posterior $p(w|\mathcal{D})$, measured by the KL divergence. Hence, we define the optimal parameters θ^{opt} as

$$\begin{aligned}\theta^{opt} &= arg\ min_\theta KL[q_\theta(w|\mathcal{D})\|p(w|\mathcal{D})] \\ &= arg\ min_\theta KL[q_\theta(w|\mathcal{D})\|p(w)] \\ &- \mathbb{E}_{q(w|\theta)}[\log p(\mathcal{D}|w)] + \log p(\mathcal{D})\end{aligned} \tag{4}$$

where

$$KL[q_\theta(w|\mathcal{D})\|p(w)] = \int q_\theta(w|\mathcal{D}) \log \frac{q_\theta(w|\mathcal{D})}{p(w)}\, dw. \tag{5}$$

Activation Function

We apply the *Softplus* function because we want to ensure that the variance $\alpha\mu^2$ never becomes zero.

$$\text{Softplus}(x) = \frac{1}{\beta} \cdot \log\left(1 + \exp(\beta \cdot x)\right) \tag{6}$$

where β is by default set to 1.

Network Architecture

For all conducted experiments, we implement the foregoing description of Bayesian Cnns with variational inference in LeNet-5 and AlexNet.

GitHub repositor:

`https://github.com/kumar-shridhar/PyTorch-BayesianCN`

Objective Function

$$\mathcal{F}(\mathcal{D}, \theta) \approx \sum_{i=1}^{n} \log q_\theta(w^{(i)}|\mathcal{D}) - \log p(w^{(i)}) - \log p(\mathcal{D}|w^{(i)}) \quad (7)$$

where n is the number of draws.

Variational Posterior

$$\mathcal{F}(\mathcal{D}, \theta) \approx \sum_{i=1}^{n} \log q_\theta(w^{(i)}|\mathcal{D}) - \log p(w^{(i)}) - \log p(\mathcal{D}|w^{(i)})$$

The variational posterior is taken as Gaussian distribution centred around mean μ and variance as σ^2

$$q_\theta(w^{(i)}|\mathcal{D}) = \prod_i \mathcal{N}(w_i|\mu, \sigma^2) \tag{8}$$

Then,

$$\log(q_\theta(w^{(i)}|\mathcal{D})) = \sum_i \log \mathcal{N}(w_i|\mu, \sigma^2) \tag{9}$$

Prior

$$\mathcal{F}(\mathcal{D}, \theta) \approx \sum_{i=1}^{n} \log q_\theta(w^{(i)}|\mathcal{D}) - \log p(w^{(i)}) - \log p(\mathcal{D}|w^{(i)})$$

Define the prior over the weights as a product of individual Gaussians:

$$p(w^{(i)}) = \prod_i \mathcal{N}(w_i|0, \sigma_p^2) \tag{10}$$

Then,

$$\log(p(w^{(i)})) = \sum_i \log \mathcal{N}(w_i|0, \sigma_p^2) \tag{11}$$

Likelihood

$$\mathcal{F}(\mathcal{D}, \theta) \approx \sum_{i=1}^{n} \log q_\theta(w^{(i)}|\mathcal{D}) - \log p(w^{(i)}) - \log p(\mathcal{D}|w^{(i)})$$

$\log p(\mathcal{D}|w^{(i)})$ is the likelihood term and is computed using the softmax function.

Parameter Initialization

- We express variance σ as $\sigma_i = \mathit{softplus}(\rho_i)$ where ρ is an unconstrained parameter.

Parameter Initialization

- We express variance σ as $\sigma_i = \mathit{softplus}(\rho_i)$ where ρ is an unconstrained parameter.
- We take the Gaussian distribution and initialize mean μ as 0 and variance σ (and hence ρ) randomly.

Parameter Initialization

- We express variance σ as $\sigma_i = \mathit{softplus}(\rho_i)$ where ρ is an unconstrained parameter.
- We take the Gaussian distribution and initialize mean μ as 0 and variance σ (and hence ρ) randomly.
- We observed mean centred around 0 and a variance starting with a big number and gradually decreasing over time.

Parameter Initialization

- We express variance σ as $\sigma_i = \textit{softplus}(\rho_i)$ where ρ is an unconstrained parameter.
- We take the Gaussian distribution and initialize mean μ as 0 and variance σ (and hence ρ) randomly.
- We observed mean centred around 0 and a variance starting with a big number and gradually decreasing over time.
- We perform gradient descent over $\theta = (\mu, \rho)$, and individual weight $w_i \sim \mathcal{N}(w_i|\mu_i, \sigma_i)$.

Optimizer

For all our tasks, we take Adam optimizer to optimize the parameters. We also perform the local reparameterization trick as mentioned in the previous section and take the gradient of the combined loss function with respect to the variational parameters (μ, ρ).

Model Pruning

- Apply an L1 norm over the network and for all the weights value as zero or below a defined threshold are removed and the model is pruned.

Model Pruning

- Apply an L1 norm over the network and for all the weights value as zero or below a defined threshold are removed and the model is pruned.
- Reduce the size of the network to half (AlexNet and LeNet- 5) by reducing the number of filters to half.