[Paper review 13]

Bayesian Uncertainty Estimation for Batch Normalized Deep Networks

(Mattias Teye, et al , 2018)

[Contents]

0. Abstract

show that **"BN(batch normalization) = approximate inference in Bayesian models"**

- allow us to make estimate of "model uncertainty" using conventional architecture, without modifications to the network!

1. Introduction

In this work, focus on estimating "**predictive uncertainties in DNN**"

Previous works

1) Drop out (Gal & Ghahramani, 2015)

- any network trained with dropout is an approximate Bayesian Model
- uncertainty estimates can be obtained by "computing the variance of multiple predictions" with different dropout masks
- technique called "MCDO" (Monte Carlo Dropout)

  (can be applied to any pre-trained networks with dropout layers)

  (uncertainty estimation for free!)

2) Batch normalization

- ability to stabilize learning with improved generalization
- mini-batch statistics depend on randomly selected batch memebers

  $\rightarrow$ using this stochasticity, this paper shows that "using BN can be cast as an approximate Bayesian Inference"

  $\rightarrow$ MCBN (Monte Carlo Batch Normalization)

2. Related Works

Bayesian models for modeling uncertainty

- Gaussian process for infinite parameters (Neal, 1995)
- Bayesian NN (Mackay, 1992)
- Variational Inference (Hinton & Van Camp, 1993) (Kingma & Welling, 2014)
- Probabilistic Backpropagation (PBP) (Graves, 2011)
- Factorized posterior via Expectation Propagation (Hernandez-Lobato & Adams, 2015)
- Deep GP (Bui et al., 2016)
- Bayesian Hypernetworks (Kruger et al., 2017)
- Multiplicative Normalizing Flows (MNF) (Louizos & Welling, 2017)

They all require "MODIFICATION to the architecture"

- Network trained with dropout implicitly performs the VI object

  Thus, any network trained with dropout can be treated as approximate Bayesian Model (Gal & Ghahramani, 2015)

  (By making multiple predictions $\rightarrow$ get mean & variance of them)

3. Method

3.1 Bayesian Modeling

deterministic model : $\hat{\mathbf{y}}=\arg \max_{\mathbf{y}} f_{\omega}(\mathbf{x}, \mathbf{y})$

probabilistic model : $\hat{\mathbf{y}}=\arg \max_{\mathbf{y}} f_{\omega}(\mathbf{x}, \mathbf{y})=\arg \max_{\mathbf{y}} p(\mathrm{y} \mid \mathrm{x}, \omega)$

- posterior distribution : $p(\boldsymbol{\omega} \mid \mathbf{D})$
- probabilistic prediction : $p(\mathbf{y} \mid \mathbf{x}, \mathbf{D})=\int f_{\boldsymbol{\omega}}(\mathbf{x}, \mathbf{y}) p(\boldsymbol{\omega} \mid \mathbf{D}) d \boldsymbol{\omega}$

Variational Approximation (VA)

- learn $q_{\theta}(\omega)$ that minimizes $\mathrm{KL}\left(q_{\theta}(\omega) \| p(\omega \mid \mathrm{D})\right)$
- minimizing $\mathrm{KL}\left(q_{\theta}(\omega) \| p(\omega \mid \mathrm{D})\right)$

  = maximizing ELBO

= minimizing negative ELBO (= $\mathcal{L}_{\mathrm{VA}}(\boldsymbol{\theta}):=-\sum_{i=1}^{N} \int q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) \ln f_{\boldsymbol{\omega}}\left(\mathbf{x}_{i}, \mathbf{y}_{i}\right) \mathrm{d} \boldsymbol{\omega}+\mathrm{KL}\left(q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) \| p(\boldsymbol{\omega})\right)$)

= (by MC approximation) minimizing $\hat{\mathcal{L}}_{\mathrm{VA}}(\theta):=-\frac{N}{M} \sum_{i=1}^{M} \ln f_{\omega_{i}}\left(\mathrm{x}_{i}, \mathrm{y}_{i}\right)+\mathrm{KL}\left(q_{\theta}(\omega) \| p(\omega)\right)$

(where M is the size of mini-batch (ex. 64, 128, 256 ...))

$\hat{\mathcal{L}}_{\mathrm{VA}}(\theta):=-\frac{N}{M} \sum_{i=1}^{M} \ln f_{\hat{\omega}_{i}}\left(\mathrm{x}_{i}, \mathrm{y}_{i}\right)+\mathrm{KL}\left(q_{\theta}(\omega) \| p(\omega)\right)$

- (1) data likelihood : $-\frac{N}{M} \sum_{i=1}^{M} \ln f_{\omega_{i}}\left(\mathrm{x}_{i}, \mathrm{y}_{i}\right)$
- (2) divergence of the prior w.r.t approximated posterior : $\mathrm{KL}\left(q_{\theta}(\omega) \| p(\omega)\right)$

3.2 Batch Normalized Deep Nets as Bayesian Modeling

inference function : $f_{\omega}(\mathrm{x})=\mathrm{W}^{L} a\left(\mathrm{~W}^{L-1} \ldots a\left(\mathrm{~W}^{2} a\left(\mathrm{~W}^{1} \mathrm{x}\right)\right)\right.$

- $a(\cdot)$: element-wise non linearity function
- W^{l} : weight vector at layer l
- x^{l} : input to layer l
- $h^{l}=W^{l} x^{l}$

Batch Normalization (BN)

- def) unit-wise operation as below

  (standard the distribution of each "unit's input")

  $\hat{h}^{u}=\frac{h^{u}-\mathbb{E}\left[h^{u}\right]}{\sqrt{\operatorname{Var}\left[h^{u}\right]}}$
- during...

  1) training : use "mini-batch" (thus, estimated mean & variance on minibatch B is used)

  2) evaluation : use "all training data"

  $\rightarrow$ therefore, inference at training time for a sample x is a stochastic process!

  (depends on the samples of the mini-batch)

Loss Function and Optimization

training NN with "mini-batch optimization"

= minimizing $\mathcal{L}_{\mathrm{RR}}(\omega):=\frac{1}{M} \sum_{i=1}^{M} l\left(\hat{\mathbf{y}}_{i}, \mathbf{y}_{i}\right)+\Omega(\boldsymbol{\omega})$ (regularized risk minimization)

$\mathcal{L}_{\mathrm{RR}}(\omega):=\frac{1}{M} \sum_{i=1}^{M} l\left(\hat{\mathbf{y}}_{i}, \mathbf{y}_{i}\right)+\Omega(\boldsymbol{\omega})$

- (1) empirical loss : $\frac{1}{M} \sum_{i=1}^{M} l\left(\hat{\mathbf{y}}_{i}, \mathbf{y}_{i}\right)$
- (2) regularization : $\Omega(\boldsymbol{\omega})$

if we set loss function as cross-entropy or SSE , we can also express $\mathcal{L}_{\mathrm{RR}}$ as below

(= minimizing the negative log likelihood)

$\mathcal{L}_{\mathrm{RR}}(\omega):=-\frac{1}{M\tau}\sum_{i=1}^{M}\ln f_{\omega}\left(\mathrm{x}_i,\mathrm{y}_i\right)+\Omega(\omega)$ ($\tau=1$ for classification)

with BN, parameters : $\left\{\mathbf{W}^{1:L},\gamma^{1:L},\boldsymbol{\beta}^{1:L},\boldsymbol{\mu}_{\mathrm{B}}^{1:L},\sigma_{\mathrm{B}}^{1:L}\right\}$

- learnable params : $\theta=\left\{\mathbf{W}^{1:L},\gamma^{1:L},\beta^{1:L}\right\}$
- stochastic params : $\omega=\left\{\mu_{\mathrm{B}}^{1:L},\sigma_{\mathrm{B}}^{1:L}\right\}$,

  $$\begin{aligned}\mathcal{L}_{\mathrm{RR}}(\omega)&:=-\frac{1}{M\tau}\sum_{i=1}^{M}\ln f_{\omega}\left(\mathrm{x}_i,\mathrm{y}_i\right)+\Omega(\omega)\\&=-\frac{1}{M\tau}\sum_{i=1}^{M}\ln f_{\{\boldsymbol{\theta},\hat{\boldsymbol{\omega}}_i\}}\left(\mathbf{x}_i,\mathbf{y}_i\right)+\Omega(\boldsymbol{\theta})\end{aligned}$$

  (where $\hat{\boldsymbol{\omega}}_i$: mean & variance for sample i's mini-batch)

  ($\hat{\boldsymbol{\omega}}_i$ needs to be i.i.d for training data, but for large number of epochs, it converges to i.i.d cases)

We can estimate uncertainty of predictions by using **"Inherent stochasticity of BN"**

3.3 Prior $p(\omega)$

for VA & BN to be same.... $\frac{\partial}{\partial\theta}$ of (eq 1) and (eq 2) should be equivalent up to a scaling factor

- (eq 1) $\hat{\mathcal{L}}_{\mathrm{VA}}(\theta)=-\frac{N}{M}\sum_{i=1}^{M}\ln f_{\omega_i}\left(\mathrm{x}_i,\mathrm{y}_i\right)+\mathrm{KL}\left(q_\theta(\omega)\|p(\omega)\right)$
- (eq 2) $\mathcal{L}_{\mathrm{RR}}(\omega)=-\frac{1}{M\tau}\sum_{i=1}^{M}\ln f_{\{\boldsymbol{\theta},\hat{\boldsymbol{\omega}}_i\}}\left(\mathbf{x}_i,\mathbf{y}_i\right)+\Omega(\boldsymbol{\theta})$

That is, $\frac{\partial}{\partial\theta}\mathrm{KL}(q_\theta(\omega)\|p(\omega))=N\tau\frac{\partial}{\partial\theta}\Omega(\boldsymbol{\theta})$

How to satisfy the condition above?

Solution (1)

- let the prior $p(\omega)$ imply the regularization term $\Omega(\theta)$
- in (eq 1), contribution of $\mathrm{KL}\left(q_\theta(\omega)\|p(\omega)\right)$ to $\hat{\mathcal{L}}_{\mathrm{VA}}$ is "inversly scaled with N "

  (that is, as $N\rightarrow\infty$, NO regularization)

Solution (2)

- let the regularization term $\Omega(\theta)$ imply the prior $p(\omega)$
- ex) L-2 regularization : $\Omega(\theta)=\lambda\sum_{l=1:L}\left\|W^l\right\|^2$

3.4 Predictive Uncertainty in Batch Normalized Deep Nets

approximate predictive distribution : $p^*(\mathbf{y} \mid \mathbf{x}, \mathbf{D}) := \int f_\omega(\mathbf{x}, \mathbf{y}) q_\theta(\boldsymbol{\omega}) d\boldsymbol{\omega}$

by Dropout as Bayesian Inference (Gal, 2016)

- mean : $\mathbb{E}_{p^*}[\mathbf{y}] \approx \frac{1}{T} \sum_{i=1}^{T} f_{\hat{\omega}_i}(\mathrm{x})$
- covariance : $\mathrm{Cov}_{p^*}[\mathbf{y}] \approx \tau^{-1}\mathbf{I} + \frac{1}{T} \sum_{i=1}^{T} f_{\hat{\omega}_i}(\mathbf{x})^\top f_{\hat{\omega}_i}(\mathbf{x}) - \mathbb{E}_{p^*}[\mathbf{y}]^\top \mathbb{E}_{p^*}[\mathbf{y}]$

  (where $\hat{\omega}_i$ corresponds to sampling the net's stochastic params $\omega = \left\{\mu_{\mathrm{B}}^{1:L}, \sigma_{\mathrm{B}}^{1:L}\right\}$)

  (Sampling $\hat{\omega}_i$ involves sampling a batch B from training set)

Algorithm summary

network is trained just as a regular BN network!

But, instead of replacing $\omega = \left\{\mu_{\mathrm{B}}^{1:L}, \sigma_{\mathrm{B}}^{1:L}\right\}$ with population values from D,

we update these parameters stochastically, once for each forward pass

Algorithm 1 MCBN Algorithm

Input: sample x, number of inferences T, batchsize b
Output: mean prediction $\hat{y}$, predictive uncertainty σ^2

$\mathbf{y} = \{\}$
loop for T iterations
    $B \sim D$ // mini batch
    $\hat{\boldsymbol{\omega}} = \{\boldsymbol{\mu}_B, \boldsymbol{\sigma}_B\}$ // mini batch mean and variance
    $\mathbf{y} = \mathbf{y} \cup f_{\hat{\boldsymbol{\omega}}}(x)$
end loop
$\hat{y} = \mathbb{E}[\mathbf{y}]$
$\sigma^2 = \mathrm{Cov}[\mathbf{y}] + \tau^{-1}\mathbf{I}$ // for regression