Uncertainty Estimation with Noisy Optimizers

Master Seminar Beyond Deep Learning: Selected Topics on Novel Challenges

Orhun Güley

09.12.2020

Technical University of Munich

Table of contents

1. Introduction: Bayesian Inference
2. Variational Inference(VI)
3. Natural Gradient
4. Variational Inference using Noisy Natural Gradients[1]
   - Noisy Adam
   - Noisy K-FAC
5. Variational Online Gauss-Newton(VOGN)[2]
   - Vadam
6. Comparison
7. Evaluation

[1] Noisy Natural Gradient as Variational Inference (Zhang et al. [2018])
[2] Fast and Scalable Bayesian Deep Learning by Weight-Perturbation in Adam(Khan et al. [2018])

Introduction: Bayesian Inference

Why Bayesian?

- Important tasks that have high risk or more interpretability require more than just "high accuracies".
- Autonomous driving, medical diagnosis, financial predictions for algorithmic trading...
- Bayesian Inference provides a framework where you can measure your confidence in your prediction.

Bayesian Inference

Given a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)_{i=1}^n\}$,

$$p(w|\mathcal{D}) = \frac{p(\mathcal{D}|w)p(w)}{p(\mathcal{D})} = \frac{p(\mathcal{D}|w)p(w)}{\int p(\mathcal{D}|w)p(w)} = \frac{p(\mathcal{D}|w)p(w)}{\sum_{i=1}^n p(\mathcal{D}|w)p(w)} \tag{1}$$

- **Frequentist** way: Directly maximizing the likelihood of the data given weights.
- In **Bayesian inference**, we put a prior distribution on weights and we try to find the posterior distribution
- **Likelihood**: $p(\mathcal{D}|w)$
- **Prior**: $p(w)$
- **Evidence**: $p(\mathcal{D}) = \int p(\mathcal{D}|w)p(w) \rightarrow$ Not tractable in complex models like deep neural networks!

Bayesian Inference

- Due to the **intractability of the evidence term**, we do not have an exact solution. But we can approximate to posterior!
- **Several methods** $\rightarrow$ variational inference(VI), Markov Chain Monte Carlo(MCMC), and ensemble methods like MC-Dropout...
- In the recent years, it has been shown that with slight modifications on well-known optimizers, they perform variational inference!
- This presentation's topic is on variational inference.

Variational Inference(VI)

Variational Inference: KL-Divergence

- Takes roots from relative entropy in information theory.
- A **non-negative, non-symmetric** measure between two probability distribution, defined as

$$\mathbb{KL}\left(q(x) \| p(x)\right) = \int q(x) \log \frac{q(x)}{p(x)} dx = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}$$

- The integral version represents the continuous case and the summation version represents the discrete case

Variational Inference

- We try to approximate the true posterior distribution with a variational distribution $q_\theta(\mathbf{w})$, which is parametrized by θ - the parameters of the posterior distribution.
- **Example:** If we model our approximate posterior q to be a Gaussian distribution

$$q_\theta(\mathbf{w}) := \mathcal{N}(w|\mu, \, diag(\sigma^2)) \quad \text{or} \quad q_\theta(\mathbf{w}) := \mathcal{N}(w|\mu, \, \Sigma)$$

, the parameters would be

$$\theta = (\mu, \sigma) \quad \text{or} \quad \theta = (\mu, \Sigma)$$

, where μ, $\sigma \in \mathbb{R}^D$, and $\Sigma \in \mathbb{R}^{DxD}$.

Variational Inference

We try to maximize variational parameters θ:

$$\begin{aligned}\theta^\star &= \arg\min_\theta \mathbb{KL}[q_\theta(\mathrm{w}) \| p(w|\mathcal{D})] \\ &= \arg\min_\theta \underbrace{\mathbb{KL}[q_\theta(\mathrm{w} \mid \theta) \| p(\mathrm{w})] - \mathbb{E}_{q(\mathrm{w}|\theta)}[\log p(\mathcal{D} \mid \mathrm{w})]}_{\mathcal{F}(\mathcal{D},\theta) = -\mathcal{L}(\boldsymbol{\theta}, q) \to \text{negative ELBO}}\end{aligned} \tag{2}$$

It turns out that minimizing the KL-divergence is equal to maximizing the variational objective ELBO!

Reparametrization Trick:

In order to free our gradients of μ and Σ from the Gaussian distribution by using a parameter-free noise ϵ.

1. Draw samples from noise $\epsilon \sim \mathcal{N}(0, I)$
2. Set $\mathrm{w} = t(\epsilon, \theta = \{\mu, \Sigma\}) = \mu + \Sigma \circ \epsilon$

Natural Gradient

Natural Gradient

- Natural gradient method takes its roots from information geometry.
- In optimization of the deep learning cost functions, learning takes place in the space of parameters, which is actually a space of probability distributions.
- Aims to change the gradients direction from the steepest direction in the Euclidean space to the steepest direction the **natural parameter space**.
- **Good news:** This space can be a probability distribution space!

Natural Gradient

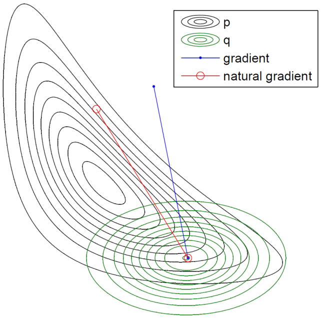


Figure 1: Gradient and vs natural gradient directions for mean of variational distribution q. VI with a diagonal covariance is applied to the posterior $p(x, y) \propto \exp\left[-9(xy-1)^2 - x^2 - y^2\right]$.[1]

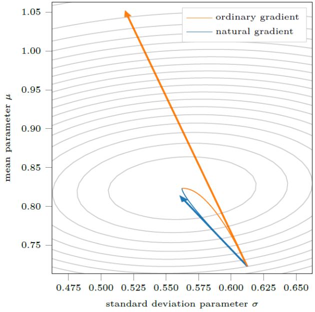


Figure 2: Arrows correspond the gradients and the curves correspond the path of gradients.[2]

[1] A gradient-based algorithm competitive with variational bayesian em for mixture of gaussians, 2009 Kuusela et al. [2009]

[2] Natural gradients inpractice: Non-conjugate variational inference in gaussian process models Salimbeni et al.

Natural Gradient: Fisher Information Matrix

- Natural gradient method scales the gradient with a inverse of **Fisher Information Matrix(FIM)** of a **score function**.
- In **probabilistic setting**, this score function is a **probability distribution**.
- The likelihood function can be a proper choice in this task, but in a Bayesian setting that we are trying to approximate the true posterior, our variational posterior $q_\theta(\mathrm{w})$ would be the right choice for score function
- FIM is given by:

$$\mathrm{F} = \mathop{\mathbb{E}}_{q_\theta(\mathrm{w})}\left[\nabla_\theta q_\theta(\mathrm{w}) \nabla_\theta q_\theta(\mathrm{w})^{\mathrm{T}}\right] \tag{3}$$

Natural Gradient

- Let $\tilde{\nabla}_\theta$ be the term for natural gradient. Then, the mathematical equation for the natural gradient of our variational objective is defined as

$$
\begin{aligned}
\tilde{\nabla}_\theta \mathcal{L}(\boldsymbol{\theta}, q) &\stackrel{\text{def}}{=} \nabla_\theta \mathcal{L}(\theta) \mathbb{E}_{\mathbf{z}} \left[(\nabla \log p_\theta(\mathbf{z}))^T (\nabla \log p_\theta(\mathbf{z})) \right]^{-1} \\
&\stackrel{\text{def}}{=} \nabla_\theta \mathcal{L}(\boldsymbol{\theta}, q) \mathbf{F}^{-1}
\end{aligned}
\tag{4}
$$

Natural Gradient

2 major variants of natural gradient that are frequently in use. The first one in named **natural gradient for point estimation(NGPE)** and the second one is **natural gradient for variational inference(NGVI)**:

1. **NGPE:** NGPE tries to estimate just the likelihood $p(\mathcal{D}|w)$ by optimizing over a loss function. For this task, the FIM is constructed by calculating the $\mathrm{F} = \mathrm{Cov}_{\mathcal{D}\sim p(\mathcal{D})}[\nabla_{\mathrm{w}} \log p(\mathcal{D}|w)]$.

2. **NGVI:** We try to fit the parameters of a variational posterior $q_\theta(\mathrm{w})$ to maximize our variational objective ELBO. We need to compute the FIM of our variational posterior q instead of the predictive distribution, which is $\mathrm{F} = \mathrm{Cov}_{\mathrm{w}\sim q_\theta(\mathrm{w})}[\nabla_{\mathrm{w}} \log q_\theta(\mathrm{w})]$.

Variational Inference using Noisy Natural Gradients[a]

[a] Noisy Natural Gradient as Variational Inference (Zhang et al. [2018])

Variational Inference Using Noisy Gradient

The following updates for μ and Λ:

$$\begin{aligned} \mu &\leftarrow \mu + \alpha\Lambda^{-1}\left[\nabla_{\mathbf{w}}\log p(\mathcal{D}|w) - \tfrac{\eta}{N}\mathbf{w}\right] \\ \Lambda &\leftarrow \left(1 - \tfrac{\beta}{N}\right)\Lambda - \beta\left[\nabla^2_{\mathbf{w}}\log p(\mathcal{D}|w), \mathbf{w}) - \tfrac{\eta}{N}\mathbf{I}\right] \end{aligned} \tag{5}$$

where $q_\theta(\mathbf{w}) = \mathcal{N}(w|\mu, \Sigma)$, the precision matrix $\Lambda = \Sigma^{-1}$, a spherical Gaussian prior $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \mathbf{I}/\eta)$, and then $\nabla^2_{\mathbf{w}}\log p(\mathbf{w}) = \eta\mathbf{I}$.

Problems:

1. Hessian is hard to compute and not differentiable in every point(Ex: ReLU).
2. Hessian might have negative eigen-values if the negative log-likelihood is not convex, which means that our update for Λ might not be positive semi-definite.

Variational Inference Using Noisy Gradient

Idea: Approximate Hessian with NGPE FIM $\mathrm{F} = \mathrm{Cov}_{\mathcal{D}\sim p(\mathcal{D})}[\nabla_{\mathrm{w}} \log p(\mathcal{D}|w)]$, which prevents both of the problems stated:

$$\Lambda \leftarrow \left(1 - \frac{\beta}{N}\right)\Lambda + \beta\left[\underbrace{\nabla_{\mathrm{w}} \log p(\mathcal{D}|w)\nabla_{\mathrm{w}} \log p(\mathcal{D}|w)^{\mathrm{T}}}_{\mathrm{F}} + \frac{\eta}{N}\mathbf{I}\right] \tag{6}$$

Observation: With a fixed prior variance η, Λ will become a damped version of the moving average of the FIM

Rewriting the equations with the FIM approximation of Hessian:

$$
\begin{aligned}
\Lambda &= N\bar{\mathbf{F}} + \eta \mathbf{I} \\
\bar{\mathbf{F}} &\leftarrow (1-\tilde{\beta})\bar{\mathbf{F}} + \tilde{\beta}[\nabla_{\mathbf{w}} \log p(\mathcal{D}|w) \nabla_{\mathbf{w}} \log p(\mathcal{D}|w)^{\mathrm{T}}] \\
\mu &\leftarrow \mu + \tilde{\alpha}\left(\bar{\mathbf{F}} + \tfrac{\eta}{N}\mathbf{I}\right)^{-1}\left[\nabla_{\mathbf{w}} \log p(\mathcal{D}|w) - \tfrac{\eta}{N}\mathbf{w}\right]
\end{aligned}
\tag{7}
$$

Noisy Adam

- Estimating Λ^{-1} as a full covariance Gaussian is unrealistic since the number of parameters needed for a full covariance matrix is $(\mathbf{dim(w)})^2$.
- Zhang et al. (2018) proposed a solution by approximating the FIM with a **diagonal matrix f**. For their noisy natural gradient version of Adam, the updates for μ and **f** are:

$$\begin{aligned} \mu &\leftarrow \mu + \tilde{\alpha}\left[\nabla_{\mathrm{w}} \log p(\mathcal{D}|w) - \tfrac{\eta}{N}\mathbf{w}\right] / \left(\bar{\mathbf{f}} + \tfrac{\eta}{N}\right) \\ \bar{\mathbf{f}} &\leftarrow (1-\tilde{\beta})\bar{\mathbf{f}} + \tilde{\beta}(\nabla_{\mathrm{w}} \log p(\mathcal{D}|w))^2 \end{aligned} \tag{8}$$

Noisy K-FAC

They use a Kronecker-factored approximation[1] to the FIM to perform efficient approximate natural gradient updates, given by

$$
\begin{aligned}
\mathrm{F}_l = \mathbb{E}\left[\mathrm{vec}\{\mathcal{D}\mathrm{W}_l\}\,\mathrm{vec}\{\mathcal{D}\mathrm{W}_l\}^\top\right] = \mathbb{E}\left[\mathbf{g}_l\mathbf{g}_l^\top \otimes \mathbf{a}_l\mathbf{a}_l^\top\right] \\
\approx \mathbb{E}\left[\mathbf{g}_l\mathbf{g}_l^\top\right] \otimes \mathbb{E}\left[\mathbf{a}_l\mathbf{a}_l^\top\right] = \mathrm{S}_l \otimes \mathrm{A}_l = \tilde{\mathrm{F}}_l
\end{aligned}
\tag{9}
$$

where, $\mathbf{a}_l$ and $\mathbf{s}_l$ correspond to input activations and outputs at layer l, $\mathcal{D}v = \nabla_v \log p(\mathcal{D}|w)$, $\mathbf{g}_l = \mathcal{D}\mathbf{s}_l$ and the gradient of weights at layer l is $\mathcal{D}\mathrm{W}_l = \mathbf{a}_l\mathbf{g}_l^T$.

Then we have the inverse FIM as

$$
\begin{aligned}
\tilde{\mathrm{F}}_l^{-1}\,\mathrm{vec}\{\nabla_{\mathrm{W}_l} h\} = \mathrm{S}_l^{-1} \otimes \mathrm{A}_l^{-1}\,\mathrm{vec}\{\nabla_{\mathrm{W}_l} h\} \\
= \mathrm{vec}\left[\mathrm{A}_l^{-1}\nabla_{\mathrm{W}_l} h \mathrm{S}_l^{-1}\right]
\end{aligned}
\tag{10}
$$

[1] Optimizing Neural Networks with Kronecker-factored Approximate Curvature

Noisy K-FAC

- By plugging in this inverse FIM approximation to eqn. (7), we will have the MVG posterior. The update rule for $\bar{\mathsf{A}}_l$ and $\bar{\mathsf{S}}_l$ is as follows:

$$\begin{aligned} \bar{\mathsf{A}}_l &\leftarrow (1-\tilde{\beta})\bar{\mathsf{A}}_l + \tilde{\beta}\mathbf{a}_l\mathbf{a}_l^\top \\ \bar{\mathsf{S}}_l &\leftarrow (1-\tilde{\beta})\bar{\mathsf{S}}_l + \tilde{\beta}\mathcal{D}\mathbf{s}_l\mathcal{D}\mathbf{s}_l^\top \end{aligned} \tag{11}$$

- And the Σ_l can be decomposed as the Kronecker product of two terms:

$$\begin{aligned} \Sigma_l &= \frac{1}{N}\left[\mathsf{S}_l^\gamma\right]^{-1} \otimes \left[\mathsf{A}_l^\gamma\right]^{-1} \\ &\triangleq \frac{1}{N}\left(\bar{\mathsf{S}}_l + \frac{1}{\pi_l}\sqrt{\frac{\eta}{N}}\mathbf{I}\right)^{-1} \otimes \left(\bar{\mathsf{A}}_l + \pi_l\sqrt{\frac{\eta}{N}}\mathbf{I}\right)^{-1} \end{aligned} \tag{12}$$

Noisy K-FAC

Algorithm 1: Noisy K-FAC. Subscript l denotes layers, . We assume zero momentum for simplicity. Differences from standard K-FAC are shown in red.

input : α: Stepsize
input : β: Exponential moving average parameter
input : $\lambda, \eta, \gamma_{\text{ex}},$: KL weighting, prior variance, extrinsic damping term
input : Stats and inverse update intervals T_{stats} and T_{inv}

$k \leftarrow 0$ and initialize $\{\boldsymbol{\mu}_l\}_{l=1}^{L}, \{S_l\}_{l=1}^{L}, \{A_l\}_{l=1}^{L}$ Calculate the intrinsic damping term $\gamma_{\text{in}} = \frac{\eta}{N}$, total damping term $\gamma = \gamma_{\text{in}} + \gamma_{\text{ex}}$.
while *not converged* **do**
  $k \leftarrow k + 1$
  $W_l \sim \mathcal{MN}\left(M_l, \frac{1}{N}\left[A_l^{\gamma_{\text{in}}}\right]^{-1}, \left[S_l^{\gamma_{\text{in}}}\right]^{-1}\right)$
  if $k \equiv 0 \pmod{T_{stats}}$ **then**
    Update the factors $\{S_l\}_{l=1}^{L}, \{A_l\}_{l=0}^{L-1}$ using eq.(11)
  if $k \equiv 0 \pmod{T_{inv}}$ **then**
    Calculate the inverses $\left\{\left[S_l^{\gamma}\right]^{-1}\right\}_{l=1}^{L}, \left\{\left[A_l^{\gamma}\right]^{-1}\right\}_{l=0}^{L-1}$ using eq. (13).
  $V_l = \nabla_{W_l} \log p(\mathcal{D}|w) - \gamma_{\text{in}} \cdot W_l$
  $M_l \leftarrow M_l + \alpha \left[A_l^{\gamma}\right]^{-1} V_l \left[S_l^{\gamma}\right]^{-1}$

Variational Online Gauss-Newton(VOGN)[a]

[a] Fast and Scalable Bayesian Deep Learning by Weight-Perturbation in Adam(Khan et al. [2018])

Variational Online Gauss-Newton (VOGN)

Shown[1] that we can express the NGVI update in terms of the MLE objective.

- Let's denote the MLE objective $f_i(\boldsymbol{\theta}) := -\log p\left(\mathcal{D}_i \mid \boldsymbol{\theta}\right)$ and and minibatch stochastic-gradient estimates as

$$f(\boldsymbol{\theta}) := \frac{1}{N} \sum_{i=1}^{N} f_i(\boldsymbol{\theta}), \quad \hat{\mathrm{g}}(\boldsymbol{\theta}) := \frac{1}{M} \sum_{i \in \mathcal{M}} \nabla_\theta f_i(\boldsymbol{\theta}), \tag{13}$$

- show the the NGVI update can be written in terms of stochastically approximated Hessian of f as

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \beta_t \left(\hat{\mathrm{g}}\left(\boldsymbol{\theta}_t\right) + \tilde{\eta}\boldsymbol{\mu}_t\right) / \left(\mathrm{s}_{t+1} + \tilde{\eta}\right), \quad \mathrm{s}_{t+1} = \left(1 - \beta_t\right) \mathrm{s}_t + \beta_t \operatorname{diag}\left[\widehat{\nabla}^2_{\theta\theta} f\left(\boldsymbol{\theta}_t\right)\right] \tag{14}$$

where $\boldsymbol{\theta}_t \sim \mathcal{N}\left(\boldsymbol{\theta} \mid \boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2\right)$ with $\boldsymbol{\sigma}_t^2 := 1/\left[N\left(\mathrm{s}_t + \tilde{\eta}\right)\right]$ and $\tilde{\eta} := \eta/N$.

[1] Fast and Scalable Bayesian Deep Learning by Weight-Perturbation in Adam(Khan et al. [2018]

Variational Online Gauss-Newton (VOGN)

Resembles the online-Newton method, since s_t contains an online estimate of the diagonal of the Hessian.
Problem: Hessian can be negative again!

Solution

Approximate Hessian with

$$\nabla^2_{\theta_j \theta_j} f(\boldsymbol{\theta}) \approx \frac{1}{M} \sum_{i \in \mathcal{M}} \left[\nabla_{\theta_j} f_i(\theta) \right]^2 := \hat{h}_j(\boldsymbol{\theta}) \tag{15}$$

which is guaranteed to be positive if initial σ^2 is positive!

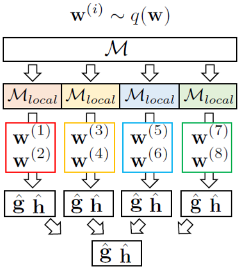
As a result, the new update rule for s_t becomes

$$\mathbf{s}_{t+1} = (1 - \beta_t)\, \mathbf{s}_t + \beta_t \hat{h}_j(\boldsymbol{\theta}) \tag{16}$$

Variational Online Gauss-Newton (VOGN)

Algorithm 1: Variational Online Gauss Newton (VOGN)

Initialise $\boldsymbol{\mu}_0$, $\mathbf{s}_0$, $\mathbf{m}_0$.
$N \leftarrow \rho N$, $\tilde{\delta} \leftarrow \tau\delta/N$.
repeat
  Sample a minibatch $\mathcal{M}$ of size M.
  Split $\mathcal{M}$ into each GPU (local minibatch $\mathcal{M}_{local}$).
  for each GPU in parallel **do**
    for $k = 1, 2, \ldots, K$ **do**
      Sample $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
      $\mathbf{w}^{(k)} \leftarrow \boldsymbol{\mu} + \epsilon\boldsymbol{\sigma}$ with $\boldsymbol{\sigma} \leftarrow (1/(N(\mathbf{s} + \tilde{\delta} + \gamma)))^{1/2}$.
      Compute $\mathbf{g}_i^{(k)} \leftarrow \nabla_w \ell(\mathbf{y}_i, \mathbf{f}_{w^{(k)}}(\mathbf{x}_i)), \forall i \in \mathcal{M}_{local}$ using the method described in Appendix B.
      $\hat{\mathbf{g}}_k \leftarrow \frac{1}{M}\sum_{i\in\mathcal{M}_{local}} \mathbf{g}_i^{(k)}$.
      $\hat{\mathbf{h}}_k \leftarrow \frac{1}{M}\sum_{i\in\mathcal{M}_{local}} (\mathbf{g}_i^{(k)})^2$.
    end for
    $\hat{\mathbf{g}} \leftarrow \frac{1}{K}\sum_{k=1}^{K} \hat{\mathbf{g}}_k$ and $\hat{\mathbf{h}} \leftarrow \frac{1}{K}\sum_{k=1}^{K} \hat{\mathbf{h}}_k$.
  end for
  AllReduce $\hat{\mathbf{g}}$, $\hat{\mathbf{h}}$.
  $\mathbf{m} \leftarrow \beta_1\mathbf{m} + (\hat{\mathbf{g}} + \tilde{\delta}\boldsymbol{\mu})$.
  $\mathbf{s} \leftarrow (1 - \tau\beta_2)\mathbf{s} + \beta_2\hat{\mathbf{h}}$.
  $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \alpha\mathbf{m}/(\mathbf{s} + \tilde{\delta} + \gamma)$.
until stopping criterion is met



Learning rate	α
Momentum rate	β_1
Exp. moving average rate	β_2
Prior precision	δ
External damping factor	γ
Tempering parameter	τ
# MC samples for training	K
Data augmentation factor	ρ

Figure 3: Table is prepared from the experimental results of [Khan et al., 2018, Zhang et al., 2018]

Variational Adam(Vadam)

- Assuming that variational posterior q is from an exponential-family distribution with natural parameter η, Khan et al. (2018) proposed the following natural-momentum method:

$$\eta_{t+1} = \eta_t + \bar{\alpha}_t \tilde{\nabla}_\eta \mathcal{L}_t + \bar{\gamma}_t \left(\eta_t - \eta_{t-1}\right) \tag{17}$$

where $\tilde{\nabla}$ it the natural-gradients with respect to natural parameter space and the gradient scaled by the Fisher information matrix of $q(\theta)$.

Variational Adam(Vadam)

- They show that the eqn.(17) can be expressed as VON update with momentum:

$$\begin{aligned}\mu_{t+1} &= \mu_t - \bar{\alpha}_t \left[\frac{1}{\mathrm{s}_{t+1} + \tilde{\eta}}\right] \circ \left(\nabla_\theta f(\boldsymbol{\theta}_t) + \tilde{\eta}\boldsymbol{\mu}_t\right) + \bar{\gamma}_t \left[\frac{\mathrm{s}_t + \tilde{\eta}}{\mathrm{s}_{t+1} + \tilde{\eta}}\right] \circ \left(\boldsymbol{\mu}_t - \boldsymbol{\mu}_{t-1}\right) \\ \mathrm{s}_{t+1} &= (1 - \bar{\alpha}_t)\,\mathrm{s}_t + \bar{\alpha}_t \nabla^2_{\theta\theta} f(\boldsymbol{\theta}_t)\end{aligned} \tag{18}$$

where $\mathbf{w}_t \sim \mathcal{N}\left(\theta \mid \mu_t, \sigma_t^2\right)$ with $\sigma_t^2 := 1/[N(\mathrm{s}_t + \tilde{\lambda})]$.

Noisy Adam vs Vadam

Algorithm 2: Noisy Adam. Differences from standard Adam are shown in red.

input : α: Stepsize
input : β_1, β_2: Exponential decay rates for updating μ and **f**
input : $, \eta, \gamma_{\text{ex}},$: prior variance, extrinsic damping term

$\mathrm{m} \leftarrow 0$;
Calculate the intrinsic damping term $\gamma_{\text{in}} = \frac{\eta}{N}$, total damping term $\gamma = \gamma_{\text{in}} + \gamma_{\text{ex}}$.
while *not converged* **do**
$\quad \mathrm{w} \sim \mathcal{N}\left(\boldsymbol{\mu}, \frac{1}{N} \operatorname{diag}\left(\mathrm{f} + \gamma_{\text{in}}\right)^{-1}\right)$
$\quad \mathrm{g} \leftarrow \nabla_{\mathrm{w}} \log p(\mathcal{D}|w)$
$\quad \mathrm{m} \leftarrow \beta_1 \cdot \mathrm{m} + (1 - \beta_1) \cdot (\mathrm{g} + \gamma_{\text{in}} \cdot \mathrm{w})$
$\quad \mathrm{f} \leftarrow \beta_2 \cdot \mathrm{f} + (1 - \beta_2) \cdot (\mathrm{g} \circ \mathrm{g})$. (Update momentum)
$\quad \tilde{\mathrm{m}} \leftarrow \mathrm{m} / \left(1 - \beta_1^k\right)$
$\quad \hat{\mathrm{m}} \leftarrow \tilde{\mathrm{m}} / (\mathrm{f} + \gamma)$
$\quad \mu \leftarrow \mu + \alpha \cdot \hat{\mathrm{m}}$ (Update parameters)

Algorithm 3: Vadam. Differences from standard Adam are shown in red

input : α: Stepsize
input : β_1, β_2: Exponential decay rates for updating μ and **f**
input : η:prior variance

$\mathrm{m} \leftarrow 0$;
while *not converged* **do**
$\quad w \leftarrow \boldsymbol{\mu} + \boldsymbol{\sigma} \circ \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathrm{I}), \boldsymbol{\sigma} \leftarrow 1/\sqrt{N\mathrm{s} + \lambda}$
$\quad \mathrm{g} \leftarrow -\nabla \log p\left(\mathcal{D}_i \mid \boldsymbol{\theta}\right)$
$\quad \mathrm{m} \leftarrow \beta_1 \mathrm{m} + (1 - \beta_1)\left(\mathrm{g} + \mu \frac{\eta}{N}\right)$
$\quad \mathrm{s} \leftarrow \beta_2 \mathrm{s} + (1 - \beta_2)(\mathrm{g} \circ \mathrm{g})$
$\quad \hat{\mathrm{m}} \leftarrow \mathrm{m} / \left(1 - \beta_1^t\right), \quad \hat{\mathrm{s}} \leftarrow \mathrm{s} / \left(1 - \beta_2^t\right)$
$\quad \mu \leftarrow \mu - \alpha \hat{\mathrm{m}} / \left(\sqrt{\hat{\mathrm{s}}} + \frac{\eta}{N}\right)$

Vadam and VOGN Convergence

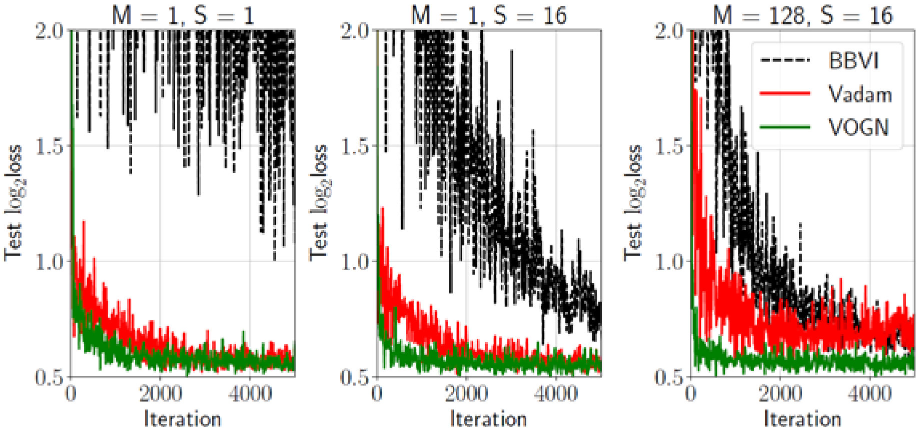


Figure 4: Results on the Australian-Scale dataset using a neural network with a hidden layer of 64 units for different minibatch sizes M and number of MC samples S. Figure taken from Khan et al. [2018]

Comparison

Comparison of Optimizers

	Noisy Adam	Vadam	Noisy K-FAC	VOGN
Prior	Spherical Gaussian	Spherical Gaussian	Spherical Gaussian	Spherical Gaussian
Posterior	Gaussian with diag. cov.	Gaussian with diag. cov.	Matrix-Variate Gaussian	Gaussian with diag. cov.
FIM Approximation	GM	GM	GM	GGN
Gradient Estimators of Gaussian	Opper Estimator	Reparametrization Trick	Opper Estimator	Reparametrization Trick

Table 1: Difference and similarities between the noisy optimizers. GM corresponds to Gradient Magnitude, and GGN corresponds to Generalized Gauss-Newton approximation.

- **Noisy Adam and Vadam** $\leftarrow$ algorithmically very similar and both easy to implement.
- Derivation of **Vadam** uses **natural-momentum** term from **Polyak's heavy ball method**, which noisy Adam doesn't provide.
- **Noisy K-FAC** $\rightarrow$ more complex weight distribution due to covariances.

Evaluation

Evaluation

	Test log-likelihood				
Dataset	**MC-Dropout**	**BBVI**	**Vadam**	**Noisy Adam**	**Noisy K-FAC**
Boston	-2.46 +/- 0.06	-2.73 +/- 0.05	-2.85 +/- 0.07	-2.558 +/- 0.032	**-2.417 +/- 0.029**
Concrete	-3.04 +/- 0.02	-3.24 +/- 0.02	-3.39 +/- 0.02	-3.145 +/- 0.023	**-3.039 +/- 0.025**
Energy	-1.99 +/- 0.02	-2.47 +/- 0.02	-2.15 +/- 0.07	-1.629 +/- 0.020	**-1.421 +/- 0.005**
Kin8nmm	0.95 +/- 0.01	0.95 +/- 0.01	0.76 +/- 0.00	1.112 +/- 0.008	**1.148 +/- 0.007**
Naval	3.80 +/- 0.01	4.46 +/- 0.03	4.72 +/- 0.22	6.231 +/- 0.041	**7.079 +/- 0.034**
Power	-2.80 +/- 0.01	-2.88 +/- 0.01	-2.88 +/- 0.01	-2.803 +/- 0.010	**-2.776 +/- 0.011**
Wine	**-0.93 +/- 0.01**	1.00 +/- 0.001	-1.00 +/- 0.01	-0.976 +/- 0.016	-0.969 +/- 0.014
Yacht	**-1.55 +/- 0.03**	-2.41 +/- 0.02	-1.70 +/- 0.03	-2.412 +/- 0.006	-2.316 +/- 0.006

Table 2: Comparison of Noisy Adam, Noisy K-FAC and Vadam with other popular methods. Table is prepared from the experimental results of [Khan et al., 2018, Zhang et al., 2018]

Evaluation

Dataset/ Architecture	Optimiser	Train/Validation Accuracy (%)	Validation NLL	Epochs	Time/ epoch (s)	ECE	AUROC
CIFAR-10/ LeNet-5 (no DA)	Adam	71.98 / **67.67**	**0.937**	210	6.96	**0.021**	0.794
	BBB	66.84 / 64.61	1.018	800	11.43†	0.045	0.784
	MC-dropout	68.41 / **67.65**	0.99	210	6.95	0.087	**0.797**
	VOGN	70.79 / **67.32**	**0.938**	210	18.33	0.046	**0.8**
CIFAR-10/ AlexNet (no DA)	Adam	100.0 / 67.94	2.83	161	3.12	0.262	0.793
	MC-dropout	97.56 / **72.20**	1.077	160	3.25	0.140	**0.818**
	VOGN	79.07 / 69.03	**0.93**	160	9.98	**0.024**	0.796
CIFAR-10/ AlexNet	Adam	97.92 / 73.59	1.480	161	3.08	0.262	0.793
	MC-dropout	80.65 / **77.04**	**0.667**	160	3.20	0.114	0.828
	VOGN	81.15 / 75.48	0.703	160	10.02	**0.016**	**0.832**
CIFAR-10/ ResNet-18	Adam	97.74 / **86.00**	0.55	160	11.97	0.082	**0.877**
	MC-dropout	88.23 / 82.85	0.51	161	12.51	0.166	0.768
	VOGN	91.62 / 84.27	**0.477**	161	53.14	**0.040**	**0.876**
ImageNet/ ResNet-18	SGD	82.63 / **67.79**	**1.38**	90	44.13	0.067	0.856
	Adam	80.96 / 66.39	1.44	90	44.40	0.064	0.855
	MC-dropout	72.96 / 65.64	1.43	90	45.86	**0.012**	0.856
	OGN	85.33 / 65.76	1.60	90	63.13	0.128	0.854
	VOGN	73.87 / **67.38**	**1.37**	90	76.04	0.029	0.854
	K-FAC	83.73 / 66.58	1.493	60	133.69	0.158	0.842
	Noisy K-FAC	72.28 / 66.44	1.44	60	179.27	0.080	0.852

Figure 5: Performance comparisons on different dataset/architecture combinations. Figure taken from Osawa et al. [2019]

- The distributed version of momentum employed VOGN method proposed by Osawa et al. (2019) has very promising results as well.
- VOGN and Noisy K-FAC has a comparable performance with standart Adam and MC-Dropout if data augmentation(DA) is applied.

Questions?

References

Mohammad Emtiyaz Khan, Didrik Nielsen, Voot Tangkaratt, Wu Lin, Yarin Gal, and Akash Srivastava. Fast and scalable bayesian deep learning by weight-perturbation in adam, 2018.

Mikael Kuusela, T. Raiko, Antti Honkela, and J. Karhunen. A gradient-based algorithm competitive with variational bayesian em for mixture of gaussians. *2009 International Joint Conference on Neural Networks*, pages 1688–1695, 2009.

Kazuki Osawa, Siddharth Swaroop, Anirudh Jain, Runa Eschenhagen, Richard E. Turner, Rio Yokota, and Mohammad Emtiyaz Khan. Practical deep learning with bayesian principles, 2019.

Hugh Salimbeni, Stefanos Eleftheriadis, and James Hensman. Natural gradients in practice: Non-conjugate variational inference in gaussian process models, 2018.

References ii

Guodong Zhang, Shengyang Sun, David Duvenaud, and Roger Grosse. Noisy natural gradient as variational inference, 2018.