SLANG: Fast Structured Covariance Approximations for Bayesian Deep Learning with Natural Gradient

Aaron Mishkin[1,2], Frederik Kunstner[1,3], Didrik Nielsen[1], Mark Schmidt[2], Mohammad Emtiyaz Khan[1]
[1]Center for Advanced Intelligence Project, RIKEN, Tokyo, Japan
[2]University of British Columbia, Vancouver, Canada, [3]École Polytechnique Fédérale de Lausanne (EPFL), Switzerland

Introduction

Motivation:

- Uncertainty estimation is essential to make reliable decisions based on the predictions of deep models, but is computationally challenging.
- It is difficult to form even a Gaussian approximation to the posterior for large deep models.
- Mean-field methods reduce the computational complexity, but yield poor estimates of the uncertainty.

Contributions:

- We propose a new **stochastic, low-rank, approximate natural-gradient (SLANG)** method for Gaussian variational inference.
- Our method estimates a “low-rank plus diagonal” covariance matrix based solely on back-propagated gradients.
- SLANG is faster and more accurate than mean-field methods, and performs comparably to state-of-the-art methods.

Natural Gradient Variational Inference

Given a deep model $p(\mathcal{D}|\boldsymbol{\theta})$ with weights $\boldsymbol{\theta}$, **Gaussian Variational Inference** computes a Gaussian approximation $q(\boldsymbol{\theta}) := \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ to the posterior by maximizing the ELBO:

$$\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathbb{E}_q \left[\underbrace{\log p(\mathcal{D}|\boldsymbol{\theta})}_{\text{Likelihood}} + \underbrace{\log \mathcal{N}(\boldsymbol{\theta} \mid 0, \mathbf{I}/\lambda)}_{\text{Prior}} - \underbrace{\log q(\boldsymbol{\theta})}_{\text{Approximation}} \right],$$

Gradient-based methods optimize the ELBO using the stochastic gradient updates (t is the iteration, γ_t is the learning rate)

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \gamma_t \hat{\nabla}_{\mu} \mathcal{L}_t, \qquad \boldsymbol{\Sigma}_{t+1} = \boldsymbol{\Sigma}_t - \gamma_t \hat{\nabla}_{\boldsymbol{\Sigma}} \mathcal{L}_t.$$

Gradient descent uses Euclidean geometry and may converge slowly.

Natural Gradient methods do steepest descent in the space of realizable approximations $q(\theta)$ by optimizing on the Riemannian manifold. This is expected to converge faster and gives the update [3]

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \beta_t \boldsymbol{\Sigma}_{t+1} \hat{\nabla}_{\mu} \mathcal{L}_t \qquad \boldsymbol{\Sigma}_{t+1}^{-1} = (1-\beta_t)\boldsymbol{\Sigma}_t^{-1} + \beta_t \hat{\nabla}_{\boldsymbol{\Sigma}} \mathcal{L}_t.$$

Both methods require storing the covariance $\boldsymbol{\Sigma}_t$, which is infeasible for large models. We build upon Variational Online Gauss-Newton [4], which can be modified to learn a low-rank approximation.

Variational Online Gauss-Newton approximates the Hessian with the empirical Fisher Information matrix $\hat{\mathbf{G}}(\boldsymbol{\theta}_t)$. This gives

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \beta_t \boldsymbol{\Sigma}_{t+1} \left[\hat{g}(\boldsymbol{\theta}_t) + \lambda \boldsymbol{\mu}_t\right]$$
$$\boldsymbol{\Sigma}_{t+1}^{-1} = (1-\beta_t)\boldsymbol{\Sigma}_t^{-1} + \beta_t \left[\hat{\mathbf{G}}(\boldsymbol{\theta}_t) + \lambda \mathbf{I}\right],$$

where $\hat{g}(\boldsymbol{\theta}_t)$ is the gradient and

$$\hat{\mathbf{G}}(\boldsymbol{\theta}_t) = \frac{1}{M} \sum_{i=1}^{M} g_i(\boldsymbol{\theta}_t) g_i(\boldsymbol{\theta}_t)^\top$$

is the empirical Fisher Information matrix for $p(\mathcal{D} \mid \boldsymbol{\theta}_t)$ computed with a minibatch of size M and individual gradients $g_i(\boldsymbol{\theta}_t)$.

SLANG

We approximate the covariance with a "low-rank plus diagonal" matrix

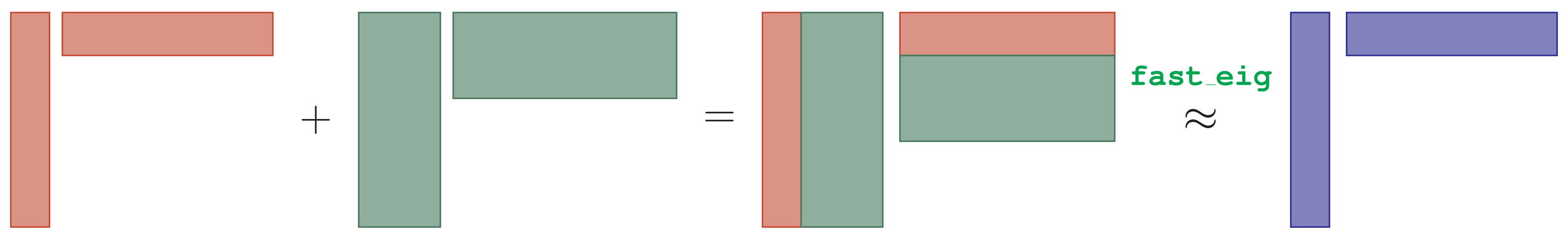
$$\boldsymbol{\Sigma}_t^{-1} \approx \hat{\boldsymbol{\Sigma}}_t^{-1} := \mathbf{U}_t \mathbf{U}_t^\top + \mathbf{D}_t,$$

where $\mathbf{U}_t$ is a $D \times L$ matrix and $\mathbf{D}_t$ is diagonal. The cost of storing and inverting this matrix is linear in D which is reasonable when $L \ll D$.
The approximate natural gradient update for $\hat{\boldsymbol{\Sigma}}_t^{-1}$ is

$$\hat{\boldsymbol{\Sigma}}_{t+1}^{-1} := \mathbf{U}_{t+1}\mathbf{U}_{t+1}^\top + \mathbf{D}_{t+1} \approx (1-\beta_t)\hat{\boldsymbol{\Sigma}}_t^{-1} + \beta_t \left[\hat{\mathbf{G}}(\boldsymbol{\theta}_t) + \lambda \mathbf{I}\right]$$

This update may increase the rank of $\mathbf{U}_{t+1}$, so we project the matrix onto a L-dimensional subspace using an eigenvalue decomposition:

$$(1-\beta_t)\hat{\boldsymbol{\Sigma}}_t^{-1} + \beta_t \left[\hat{\mathbf{G}}(\boldsymbol{\theta}_t) + \lambda \mathbf{I}\right] = \underbrace{(1-\beta_t)\mathbf{U}_t\mathbf{U}_t^\top + \beta_t \hat{\mathbf{G}}(\boldsymbol{\theta}_t)}_{\text{Rank at most } L+M} + \underbrace{(1-\beta_t)\mathbf{D}_t + \beta_t \lambda \mathbf{I}}_{\text{Diagonal component}},$$
$$\approx \underbrace{\mathbf{Q}_{1:L}\boldsymbol{\Lambda}_{1:L}\mathbf{Q}_{1:L}^\top}_{\text{Rank } L \text{ eigendecomposition}} + \underbrace{(1-\beta_t)\mathbf{D}_t + \beta_t \lambda \mathbf{I}}_{\text{Diagonal component}}.$$



$(1-\beta)\mathbf{U}_t\mathbf{U}_t^\top$ $\quad$ $\beta\mathbf{G}(\boldsymbol{\theta}_t)$ $\quad$ $\mathbf{U}_{t+1}\mathbf{U}_{t+1}^\top$

The diagonal information lost in this projection is equal to

$$\Delta_D = \text{diag}\left[(1-\beta)\mathbf{U}_t\mathbf{U}_t^\top + \beta_t \hat{\mathbf{G}}(\boldsymbol{\theta}_t) - \mathbf{U}_{t+1}\mathbf{U}_{t+1}^\top\right].$$

We add this to $\mathbf{D}_t$ as a diagonal correction. The final SLANG update is

$$\textbf{SLANG:} \quad \mathbf{U}_{t+1} = \mathbf{Q}_{1:L}\boldsymbol{\Lambda}_{1:L}^{1/2}$$
$$\mathbf{D}_{t+1} = (1-\beta)\mathbf{D}_t + \beta_t \lambda \mathbf{I} + \Delta_D.$$
$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \alpha_t \left[\mathbf{U}_{t+1}\mathbf{U}_{t+1}^\top + \mathbf{D}_{t+1}\right]^{-1} \left[\hat{\mathbf{g}}(\boldsymbol{\theta}_t) + \lambda \boldsymbol{\mu}_t\right].$$

The Algorithm

Pseudo-code for SLANG is shown in Algorithm 1. α, β are learning rates, D is denoted with a vector $\mathbf{d}$ and $\mathbf{u}_j$ and $\mathbf{v}_j$ are the columns of $\mathbf{U}$ and $\mathbf{V}$, respectively.

Algorithm 1: `SLANG`

Require: Data $\mathcal{D}$, hyperparameters $M, L, \lambda, \alpha, \beta$
Initialize $\boldsymbol{\mu}, \mathbf{U}, \mathbf{d}$
$\delta \leftarrow (1-\beta)$
while not converged **do**
  $\boldsymbol{\theta} \leftarrow$ `fast_sample`$(\boldsymbol{\mu}, \mathbf{U}, \mathbf{d})$
  $\mathcal{M} \leftarrow$ sample a minibatch
  $[\mathbf{g}_1, .., \mathbf{g}_M] \leftarrow$ `backprop`$(\mathcal{D}_{\mathcal{M}}, \boldsymbol{\theta})$
  $\mathbf{V} \leftarrow$ `fast_eig`$(\delta\mathbf{u}_1, .., \delta\mathbf{u}_L, \beta\mathbf{g}_1, .., \beta\mathbf{g}_M, L)$
  $\Delta_d \leftarrow \sum_{j=1}^{L} \delta \mathbf{u}_j^2 + \sum_{i=1}^{M} \beta \mathbf{g}_i^2 - \sum_{j=1}^{L} \mathbf{v}_j^2$
  $\mathbf{U} \leftarrow \mathbf{V}$
  $\mathbf{d} \leftarrow \delta \mathbf{d} + \Delta_d + \lambda \mathbf{1}$
  $\hat{\mathbf{g}} \leftarrow \sum_i \mathbf{g}_i + \lambda \boldsymbol{\mu}$
  $\Delta_\mu \leftarrow$ `fast_inverse`$(\hat{\mathbf{g}}, \mathbf{U}, \mathbf{d})$
  $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \alpha \Delta_\mu$
end while
return $\boldsymbol{\mu}, \mathbf{U}, \mathbf{d}$

Algorithm 2: `fast_inverse`$(\mathbf{g}, \mathbf{U}, \mathbf{d})$

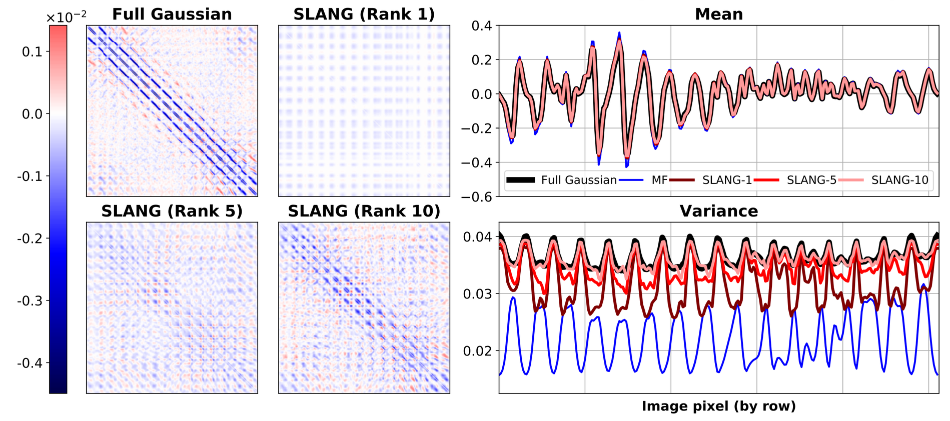
$\mathbf{A} \leftarrow (\mathbf{I}_L + \mathbf{U}^\top \mathbf{d}^{-1} \mathbf{U})^{-1}$
$\mathbf{y} \leftarrow \mathbf{d}^{-1}\mathbf{g} - \mathbf{d}^{-1}\mathbf{U}\mathbf{A}\mathbf{U}^\top\mathbf{d}^{-1}\mathbf{g}$
return $\mathbf{y}$

Algorithm 3: `fast_sample`$(\boldsymbol{\mu}, \mathbf{U}, \mathbf{d})$

$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}_D)$
$\mathbf{V} \leftarrow \mathbf{d}^{-1/2} \odot \mathbf{U}$
$\mathbf{A} \leftarrow \text{Cholesky}(\mathbf{V}^\top\mathbf{V})$
$\mathbf{B} \leftarrow \text{Cholesky}(\mathbf{I}_L + \mathbf{V}^\top\mathbf{V})$
$\mathbf{C} \leftarrow \mathbf{A}^{-\top}(\mathbf{B} - \mathbf{I}_L)\mathbf{A}^{-1}$
$\mathbf{K} \leftarrow (\mathbf{C} + \mathbf{V}^\top\mathbf{V})^{-1}$
$\mathbf{y} \leftarrow \mathbf{d}^{-1/2}\boldsymbol{\epsilon} - \mathbf{V}\mathbf{K}\mathbf{V}^\top\boldsymbol{\epsilon}$
return $\boldsymbol{\mu} + \mathbf{y}$

Results

Covariance Structure for Logistic Regression on USPS



- SLANG doesn’t underestimate variance like mean-field methods.

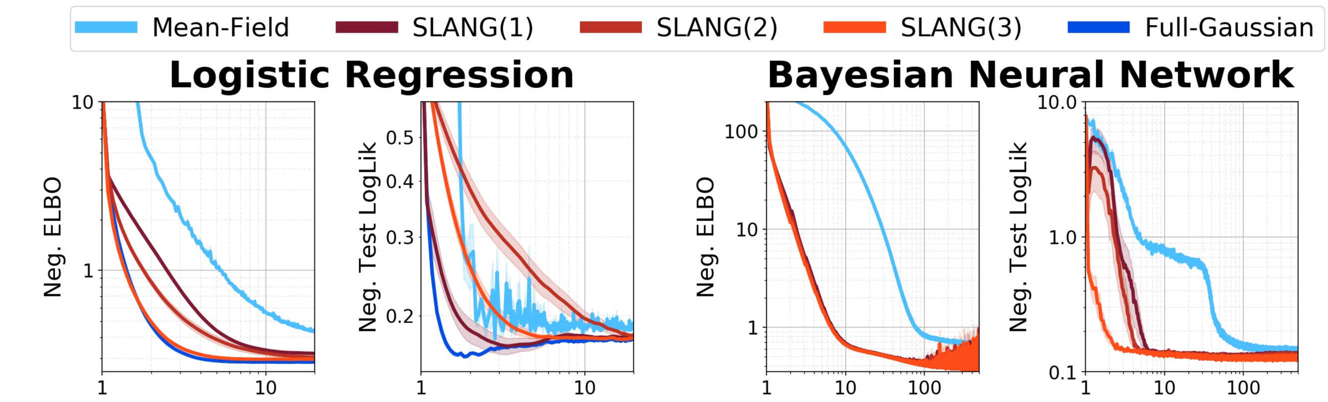
Logistic Regression Results

- SLANG performs similarly to full-Gaussian methods at test time.

Dataset	Metrics	Mean-Field EF	Mean-Field Hess.	Mean-Field Exact	SLANG L = 1	SLANG L = 5	SLANG L = 10	Full Gaussian EF	Full Gaussian Hess.	Full Gaussian Exact
Australian	NLL	0.348	0.347	0.341	0.342	0.339	**0.338**	0.340	0.339	0.338
	KL ($\times 10^4$)	2.240	2.030	0.195	0.033	0.008	**0.002**	0.000	0.000	0.000
a1a	NLL	0.339	0.339	0.339	0.339	0.339	**0.339**	0.339	0.339	0.339
	KL ($\times 10^2$)	2.590	2.208	1.295	0.305	0.173	**0.118**	0.014	0.000	0.000
USPS	NLL	0.139	0.139	0.138	0.132	0.132	**0.131**	0.131	0.130	0.130
3vs5	KL ($\times 10^1$)	7.684	7.188	7.083	1.492	0.755	**0.448**	0.180	0.001	0.000

Convergence Experiments

- SLANG converges faster than mean-field methods for logistic regression and BNNs.



UCI Regression with Bayesian Neural Networks:

- Performance on BNNs is comparable to Bayesian Dropout [2] and Bayes-by-Backprop [1].

Dataset	Test RMSE BBB	Test RMSE Dropout	Test RMSE SLANG	Test log-likelihood BBB	Test log-likelihood Dropout	Test log-likelihood SLANG
Boston	3.43 $\pm$ 0.20	**2.97** $\pm$ **0.19**	3.21 $\pm$ 0.19	-2.66 $\pm$ 0.06	**-2.46** $\pm$ **0.06**	-2.58 $\pm$ 0.05
Concrete	6.16 $\pm$ 0.13	**5.23** $\pm$ **0.12**	5.58 $\pm$ 0.19	-3.25 $\pm$ 0.02	**-3.04** $\pm$ **0.02**	-3.13 $\pm$ 0.03
Energy	0.97 $\pm$ 0.09	1.66 $\pm$ 0.04	**0.64** $\pm$ **0.03**	-1.45 $\pm$ 0.10	-1.99 $\pm$ 0.02	**-1.12** $\pm$ **0.01**
Kin8nm	0.08 $\pm$ 0.00	0.10 $\pm$ 0.00	**0.08** $\pm$ **0.00**	1.07 $\pm$ 0.00	0.95 $\pm$ 0.01	1.06 $\pm$ 0.00
Naval	0.00 $\pm$ 0.00	0.01 $\pm$ 0.00	**0.00** $\pm$ **0.00**	4.61 $\pm$ 0.01	3.80 $\pm$ 0.01	**4.76** $\pm$ **0.00**
Power	4.21 $\pm$ 0.03	**4.02** $\pm$ **0.04**	4.16 $\pm$ 0.04	-2.86 $\pm$ 0.01	**-2.80** $\pm$ **0.01**	-2.84 $\pm$ 0.01

MNIST Classification with Bayesian Neural Networks:

	BBB	SLANG L = 1	SLANG L = 2	SLANG L = 4	SLANG L = 8	SLANG L = 16	SLANG L = 32
Test Error	1.82%	2.00%	1.95%	1.81%	1.92%	1.77%	**1.73%**

- Larger L leads to better test accuracy.

References

[1] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural networks. *CoRR*, abs/1505.05424, 2015.
[2] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of 33rd ICML*, pages 1050–1059, 2016.
[3] M. E. Khan and D. Nielsen. Fast yet simple natural-gradient descent for variational inference in complex models. *CoRR*, abs/1807.04489, 2018.
[4] M. E. Khan, D. Nielsen, V. Tangkaratt, W. Lin, Y. Gal, and A. Srivastava. Fast and scalable Bayesian deep learning by weight-perturbation in Adam. In *Proceedings of 35 ICML*, pages 2611–2620, 2018.