Handling the Positive-Definite Constraint in the Bayesian Learning Rule

Wu Lin (UBC)

Joint work with Mark Schmidt (UBC) and Mohammad Emtiyaz Khan (AIP, RIKEN)

Bayesian Learning Rule

Bayes vs Optimization

Unified via Bayesian Learning Rule

A Natural-gradient Variational Inference Algorithm

Challenges: How to deal with constraints
Contributions: Improved Bayesian Learning Rule
efficient updates for the positive-definite constraints
using Riemannian Gradient Descent

Two different worlds in ML

Bayesian:

Given: data **Infer:** posterior

latent variables

model: prior + likelihood

Unconstrained optimization:

Given: data **Min:** objective

decision variables

model: regularizer + loss

Distinct methods in each world:

Inference methods v.s. Optimizers

Scope

Challenges:
- unify methods in both worlds
- use inference techniques to design optimizers
- do optimization with uncertainty

The Bayesian learning rule (Khan & Rue 2019):
- **variational inference** approach unifies methods
- help to design new optimizers
- give an uncertainty estimation for decision variables

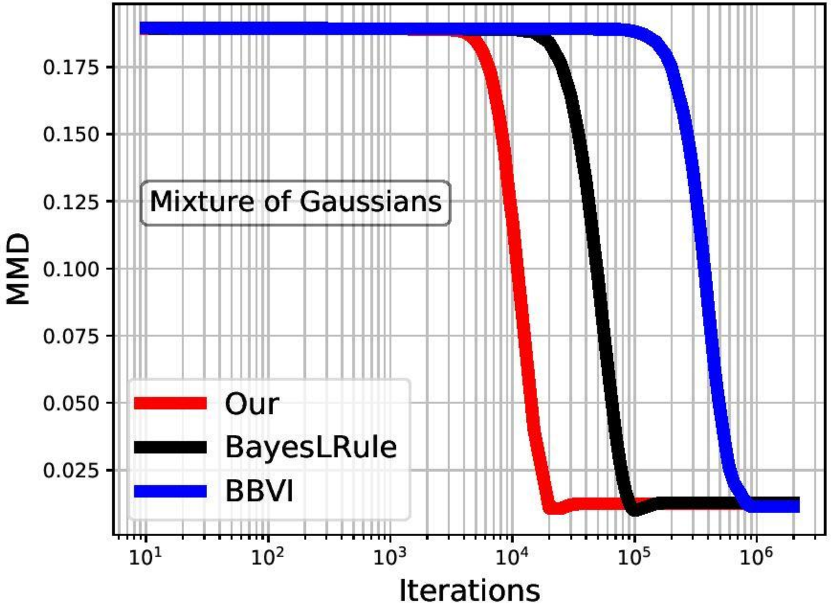
Issue of the rule: expensive to deal with constraints

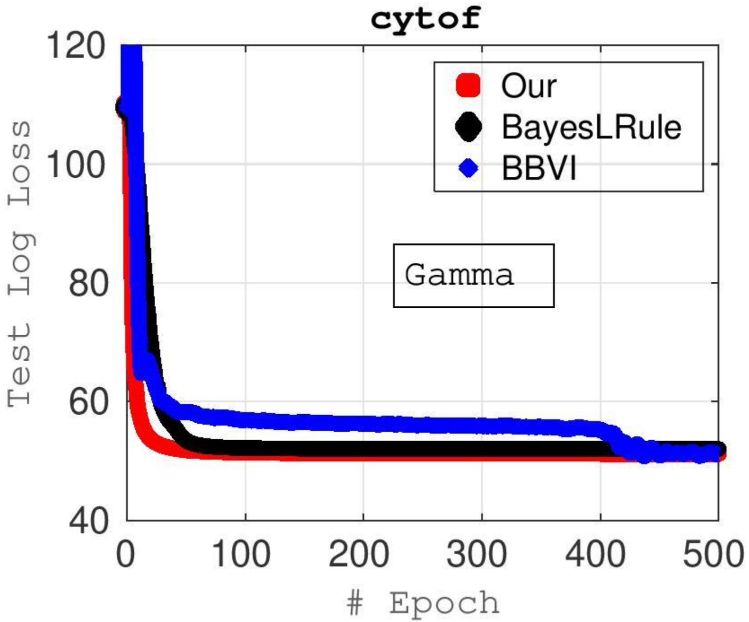
Example:

Gaussian posterior approximation (covariance matrix is **positive-definite**)

Our contributions

- new (constrained) parametrization
- closed-form correction term to address the constraint issue
- Gaussian, mixture of Gaussians, gamma, and more approximations





Outline

- Bayesian learning rule
- Key idea from a geometric viewpoint
- Example of our new rule
- Experimental results

Variational inference

Variational Inference (VI): data $\mathcal{D}$, latent variables $\mathbf{w}$, approx: $p(\mathbf{w}|\mathcal{D})$

$$\min_{\lambda} \mathcal{L}(\lambda) = \mathbb{E}_{q(w|\lambda)}[\ell(\mathcal{D}, \mathbf{w})] + \mathbb{D}_{\mathrm{KL}}[q(\mathbf{w}|\lambda) || \exp(-R(\mathbf{w}))]$$

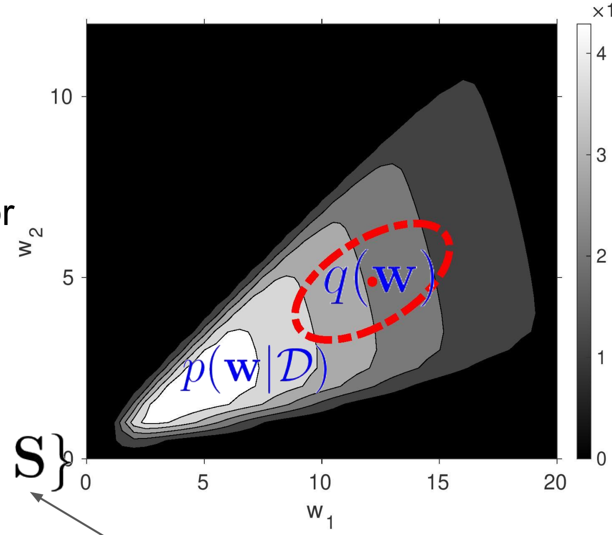
negative ELBO

loss/negative log-likelihood

$$\ell(\mathcal{D}, \mathbf{w}) = -\log p(\mathcal{D}|\mathbf{w})$$

regularizer/negative log-prior

$$R(\mathbf{w}) = -\log p(\mathbf{w})$$



Example:

$q(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mu, \mathbf{S})$, Gaussian approximation $\lambda = \{\mu, \mathbf{S}\}$

the mean

the precision matrix

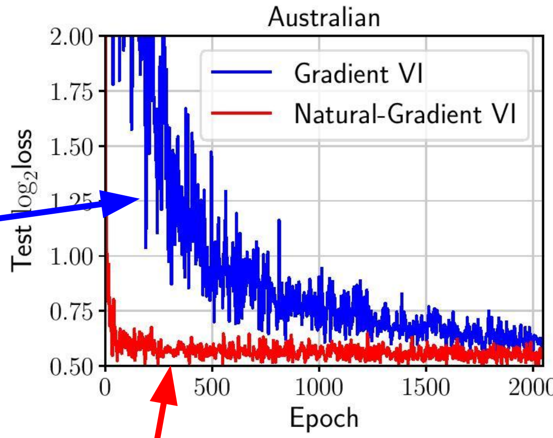
Issue: positive-definite constraint

The VI framework bridges the gap between the Bayesian world and the optimization world.

Handling the constraint

λ' is a unconstrained transformation

BBVI is **slow**



Black-box gradient VI (BBVI): $\lambda' = \lambda' - t\nabla_{\lambda'}\mathcal{L}(\lambda')$

Why NGVI instead of BBVI ?

- **fast** iteration progress

Natural-gradient VI (NGVI): $\lambda' = \lambda' - t\mathbf{F}(\lambda')^{-1}\nabla_{\lambda'}\mathcal{L}(\lambda')$

Fisher matrix of $q(\mathbf{w}|\lambda')$

Issue: **high** iteration cost in NGVI

Notation: ∇ denotes the ordinary/standard derivative in this talk

Bayesian learning rule

Bayesian learning rule is NGVI with low iteration cost

Under a right (**constrained**) parametrization λ

low iteration cost if easy to compute

$\nabla_m \mathcal{L}(\lambda) = \mathbf{F}(\lambda)^{-1} \mathcal{L}(\lambda)$, $\mathbf{m}$ is an auxiliary parameter.

Bayesian learning rule (BayesLRule): $\lambda = \lambda - t \nabla_m \mathcal{L}(\lambda)$

line search for the step-size (expensive)

The rule:

- simple way to implement NGVI but gives rise to **constrained** optimization

Example of the rule

Gaussian posterior approximation:

$$q(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mu, \mathbf{S})$$

the mean **the precision matrix**

Issue: **S** must be **positive-definite**

We define: $\bar{\ell}(\mathbf{w}) := R(\mathbf{w}) + \ell(\mathcal{D}, \mathbf{w})$

regularizer loss

Recall: minimize the negative ELBO $\mathcal{L}(\lambda)$

This work: we address this constraint issue by using Riemannian gradient descent

$\mathcal{L}(\lambda)$ can be re-written as $\mathcal{L}(\lambda) = \mathbb{E}_{q(w|\lambda)}[\bar{\ell}(\mathbf{w})] + \mathbb{E}_{q(w|\lambda)}[\log q(\mathbf{w}|\lambda)]$

Bayesian learning rule for $\min_{\lambda} \mathcal{L}(\lambda)$

$$\mathbf{S} = (1-t)\mathbf{S} + t\mathbb{E}_q[\nabla_w^2 \bar{\ell}(\mathbf{w})]$$
$$\mu = \mu - t\mathbf{S}^{-1}\mathbb{E}_q[\nabla_w \bar{\ell}(\mathbf{w})]$$

Recall: Newton's update for $\min_{w} \bar{\ell}(\mathbf{w})$

$$\mathbf{w} = \mathbf{w} - t[\nabla_w^2 \bar{\ell}(\mathbf{w})]^{-1} \nabla_w \bar{\ell}(\mathbf{w})$$

require line search for step-size t

The Bayesian learning rule
- bring the two worlds closer
- recover many optimizers (Khan & Rue 2019)

Key idea of this work

a **(shortest curve) line** as t varies

(Euclidean) gradient descent/BBVI: $L(t) = \lambda - t\nabla_\lambda \mathcal{L}(\lambda)$

starting point   Euclidean direction

$\mathcal{M}$ is a manifold induced by $q(\mathbf{w}|\mathbf{x})$, x is constrained, $\mathbf{F}(\mathbf{x})$ is the Fisher matrix

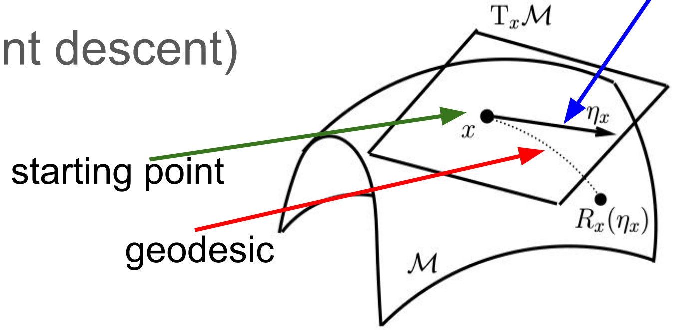
geodesic: generalization of the “shortest” curve in the manifold

natural-gradient descent/BLR

$N(t) = x - t\mathbf{F}(\mathbf{x})^{-1}\nabla_x \mathcal{L}(\mathbf{x})$

manifold direction: $-\mathbf{F}(\mathbf{x})^{-1}\nabla_x \mathcal{L}(\mathbf{x})$, let $\eta_x := -t\mathbf{F}(\mathbf{x})^{-1}\nabla_x \mathcal{L}(\mathbf{x})$

geodesic: $R(t) = R_x(\eta_x)$ (Riemannian gradient descent)



Key idea of this work

Natural-gradient descent/Bayesian learning rule(BLR):

- like a “2nd-order” method in a Euclidean case
- 1st-order method in a (Riemannian) manifold
- 1st-order approximation of the geodesic at the starting point

Challenges:

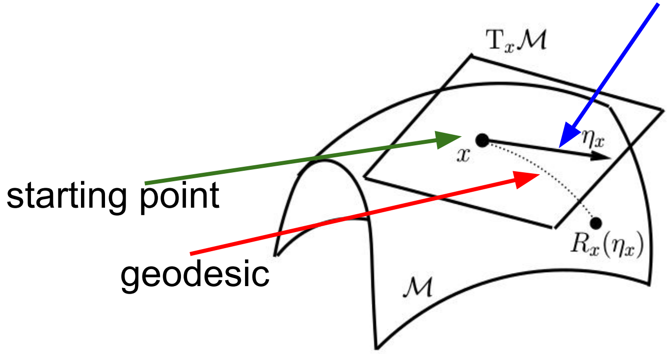
- hard to exactly compute a geodesic (manifold exponential map)
- NGD ignores the **curvature** of a geodesic

natural-gradient descent/BLR

$$N(t) = x - t\mathbf{F}(\mathbf{x})^{-1}\nabla_x \mathcal{L}(\mathbf{x})$$

This work:

- approximate a geodesic
- efficiently capture the curvature of a geodesic



Contributions of this work

capture the curvature of a geodesic

Under a **new** (constrained) parametrization, our rule:

- efficiently approximate a geodesic as: $\lambda = \lambda - t\nabla_m \mathcal{L}(\lambda) + \text{correction}$
  - efficiently approximate **Christoffel symbols/Levi-Civita coefficients**
- efficiently compute natural-gradients $\nabla_m \mathcal{L}(\lambda) = \mathbf{F}(\lambda)^{-1}\mathcal{L}(\lambda)$

Example: Gaussian approximation $q(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mu, \mathbf{S})$

the mean

the precision matrix
(positive-definite constraint)

Bayesian learning rule (BayesLRule):

$$\mathbf{S} = (1-t)\mathbf{S} + t\mathbb{E}_q[\nabla_w^2 \bar{\ell}(\mathbf{w})]$$
$$\mu = \mu - t\mathbf{S}^{-1}\mathbb{E}_q[\nabla_w \bar{\ell}(\mathbf{w})]$$

require line search for step-size t

Our rule (iBayesLRule):

$$\mu = \mu - t\mathbf{S}^{-1}\mathbb{E}_q[\nabla_w \bar{\ell}(\mathbf{w})] + \mathbf{0}$$
$$\mathbf{S} = (1-t)\mathbf{S} + t\mathbb{E}_q[\nabla_w^2 \bar{\ell}(\mathbf{w})] + \frac{t^2}{2}\mathbf{G}\mathbf{S}^{-1}\mathbf{G}$$

no need to do line search

$$\mathbf{G} := \mathbf{S} - \mathbb{E}_q[\nabla_w^2 \bar{\ell}(\mathbf{w})]$$

same time complexity in terms of big O notation

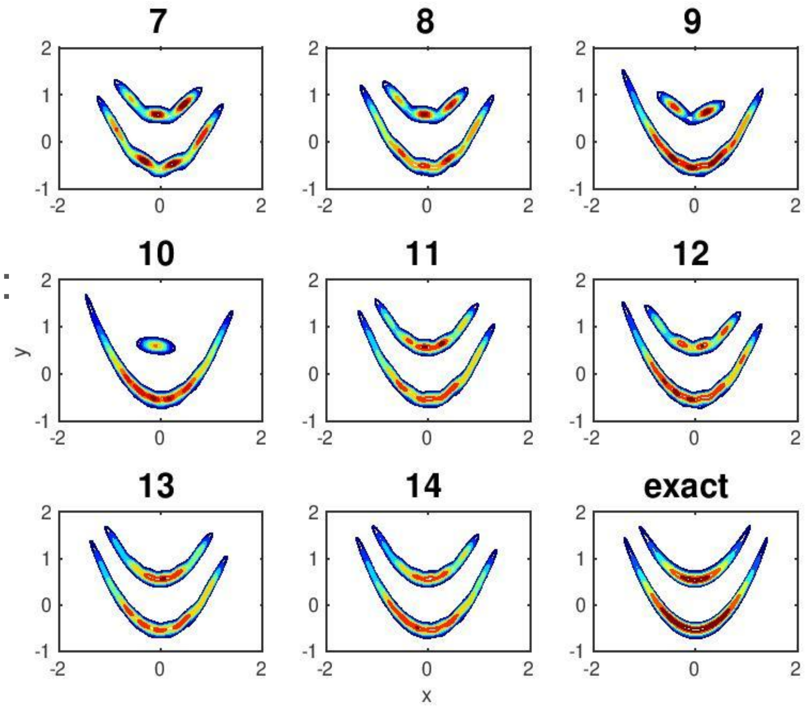
More examples

Our approach can work on useful approximations:

- gamma
- inverse Gaussian
- Wishard

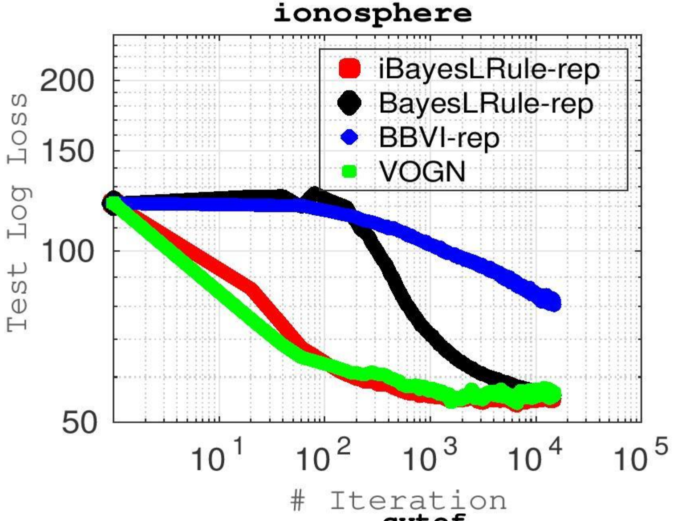
Mixture extensions:

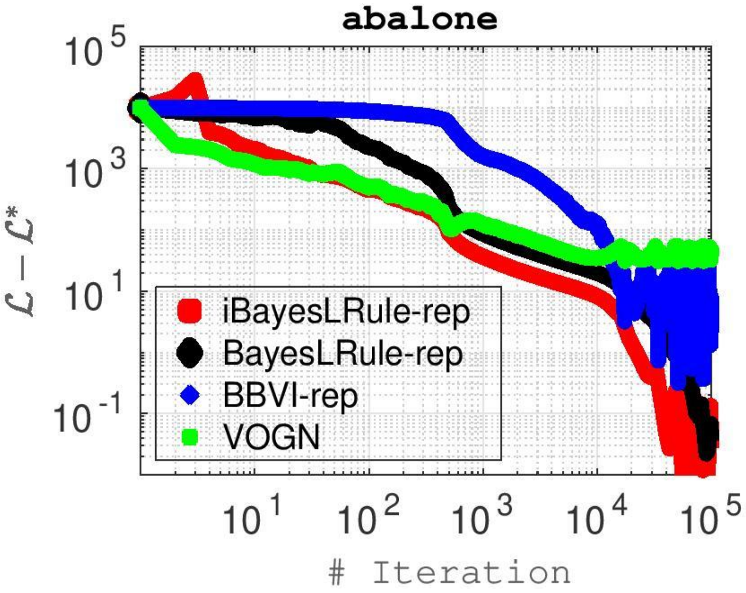
- finite mixture models (e.g., mixture of Gaussians)
- skew Gaussian (continuous Gaussian mixture)
- Student's T (Gaussian scale mixtures)



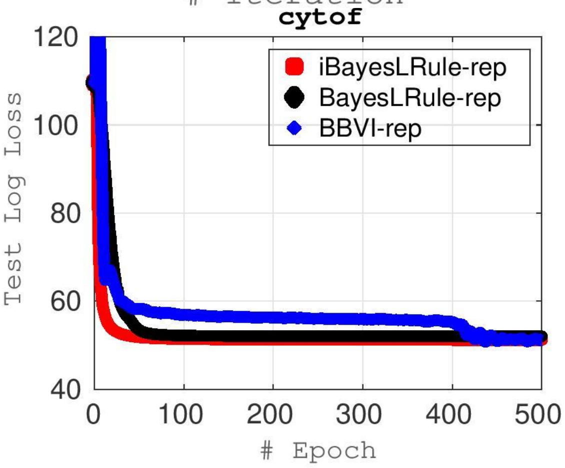
Experimental results

Full Gaussian:





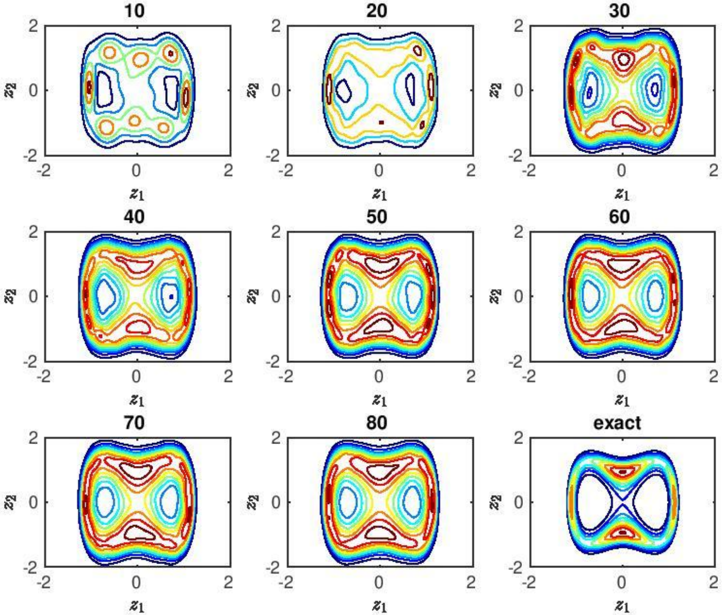
Gamma:

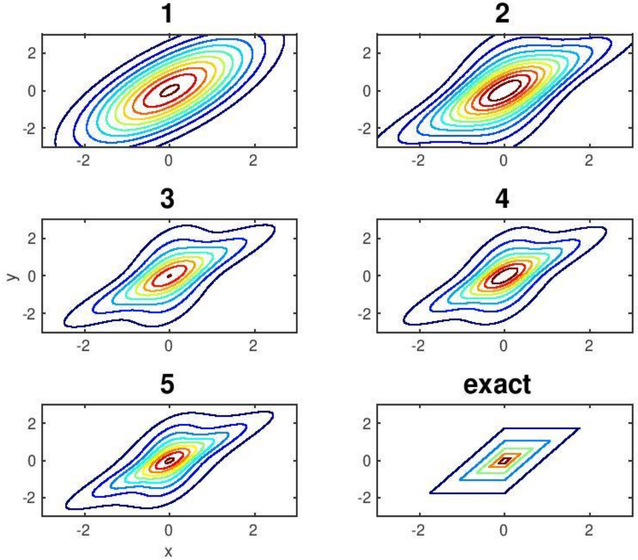


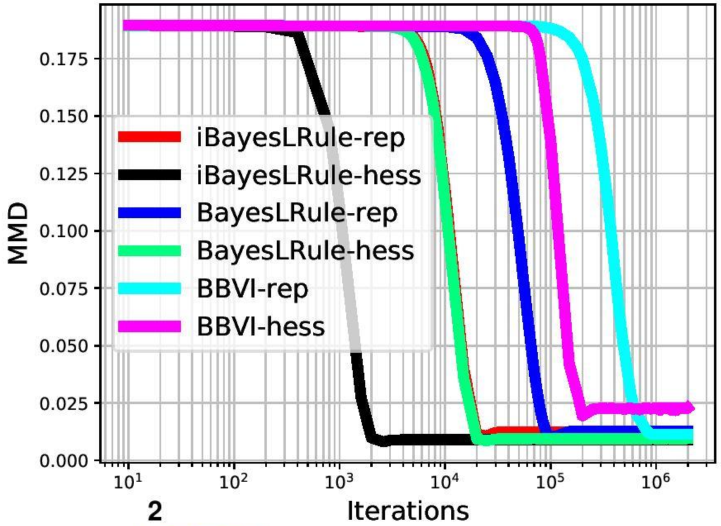
More experimental results

Mixture of full Gaussians: 2-dim , 20-dim, 300-dim

# of mixing components







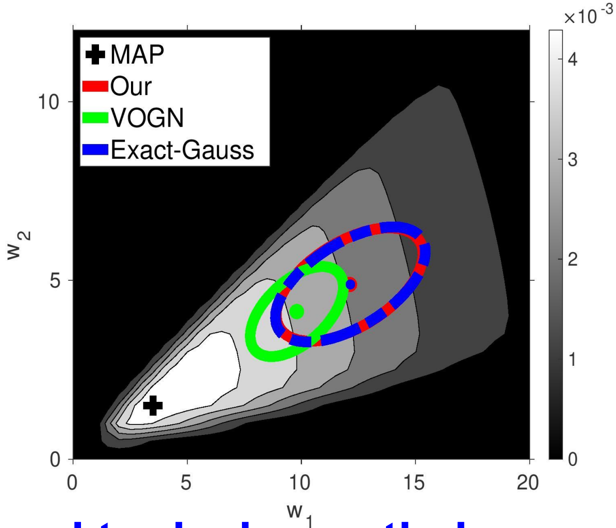
Conclusion

Improved Bayesian learning rule (under a new constrained parametrization):

- handle positive-definite constraints
- use natural-gradients with low iteration cost
- no line search for many useful approximations
- as accurate as the original Bayesian learning rule
- easy to design new optimizers (e.g., ADAM-like optimizer)

Thank You

Approaches to handle constraints



Example: Gaussian posterior approximation

Issue: parameter constraint (eg, covariance matrix)

Existing approaches to handle the constraint:

- block-box gradient VI: unconstrained, slow, accurate, **hard to design optimizers**
- Bayes learning rule: constrained, fast, accurate, **line search (expensive)**
- Gauss-Newton (VOGN): constrained, fast, specific approximation, **inaccurate**

This work:

constrained, fast, no line search, accurate, easy to design new optimizers