Bayesian Inference for Deep Learning

Inference and modern trends for Bayesian Neural Networks: Variational Inference

Simone Rossi and Maurizio Filippone

Data Science Department, EURECOM (France)

Roadmap of the tutorial

- **Introduction to Bayesian Inference**
- **Bayesian inference as Optimization with Variational Inference**
  - Introduction to variational inference (objective and gradients)
  - Challenges and solutions for variational inference on Bayesian neural networks
- **Sampling with MCMC methods**
- **Alternatives for Approximate Bayesian Deep Learning**
- **Gaussian processes and Bayesian neural networks**
- **Priors and Model Selection**
- **Uncertainty Quantification with Bayesian Neural Networks**

An Introduction to Bayesian Neural Networks with Monte-Carlo Dropout

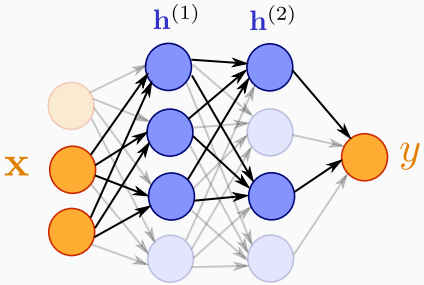
Let's reuse what we know

Dropout is a simple and powerful method to avoid overfitting in deep neural networks.

Srivastava et al. (2014). *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*, JMLR

Let's reuse what we know

Dropout is a simple and powerful method to avoid overfitting in deep neural networks.

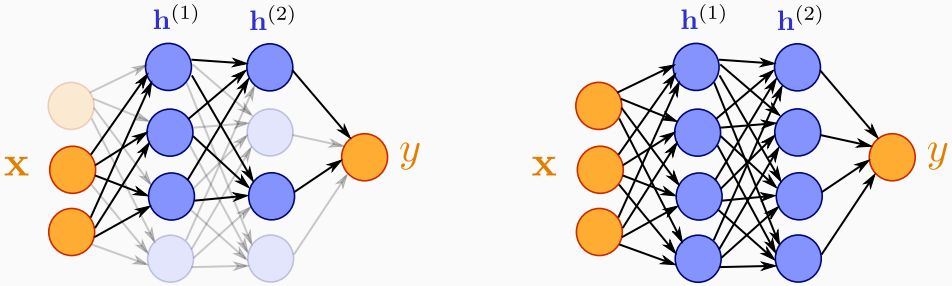


At each iteration of **train time**, some units are dropped (with probability p).

Srivastava et al. (2014). *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*, JMLR

Let's reuse what we know

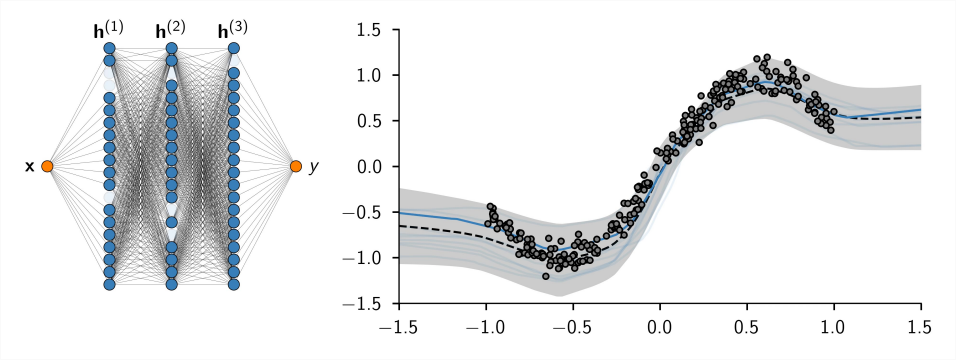
Dropout is a simple and powerful method to avoid overfitting in deep neural networks.



At each iteration of **train time**, some units are dropped (with probability p).
At **test time**, all units are considered.

Srivastava et al. (2014). *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*, JMLR

What happens if we use multiple dropout masks also at test time?



Compute the mean prediction as $\widehat{\mathbf{y}}_\star \approx \frac{1}{T}\sum_t^T f(\mathbf{x}_\star; [\{\widetilde{\mathbf{W}}^{(1)}, \ldots, \widetilde{\mathbf{W}}^{(L)}\}]_t)$ and evaluate the variance at test point $\mathrm{Var}[\widehat{\mathbf{y}}_\star]$

Gal and Ghahramani (2016). *Dropout as a Bayesian Approximation*, ICML

A bit more formal

- Training with **regularized loss** ...

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} e(\mathbf{y}_i, f(\mathbf{x}_i)) + \lambda \sum_{l=1}^{L} \|\mathbf{W}^{(l)}\|^2$$

  ... equivalent to do approximate Bayesian inference.

A bit more formal

- Training with **regularized loss** ...

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} e(\mathbf{y}_i, f(\mathbf{x}_i)) + \lambda \sum_{l=1}^{L} \|\mathbf{W}^{(l)}\|^2$$

  ... equivalent to do approximate Bayesian inference.

- If we define $\mathbf{w} = \{\mathbf{W}_1, \ldots, \mathbf{W}_L\}$ and a "posterior" distribution $q(\mathbf{w})$ such that,

$$\mathbf{W}_i = \mathbf{M}_i \cdot \text{diag}(\mathbf{z}_i), \text{ with } \quad z_{ij} \sim \text{Bern}(p_i)$$

  we can use the following objective

$$\mathcal{L}_{\text{ELBO}} = -\sum_{i=1}^{N} \int \log p(\mathbf{y}_i \mid \mathbf{x}_i, \mathbf{w}) q(\mathbf{w}) \, \mathrm{d}\mathbf{w} + \text{KL}\left[q(\mathbf{w}) \parallel p(\mathbf{w})\right]$$

Why? What is this?

An Introduction to Variational Inference

Our problem setup

- Model: $f(\mathbf{x}; \mathbf{w})$ is a L-layer neural network with weights $\mathbf{w} = \{\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(L)}\}$
- Likelihood:

$$p(\mathbf{Y} \mid \mathbf{X}, \mathbf{w}) = \prod_{n=1}^{N} p(\mathbf{y}_n | \mathbf{x}_n, \mathbf{w}) = \prod_{n=1}^{N} p(\mathbf{y}_n | f(\mathbf{x}_n; \mathbf{w}))$$

- Prior:

$$p(\mathbf{w}) = \prod_{l=1}^{L} p(\mathbf{W}^{(i)}) = \prod_{i=1}^{L} \prod_{jk} \mathcal{N}(W_{jk}^{(l)} \mid 0, 1)$$

- Posterior (using Bayes's theorem):

$$p(\mathbf{w} | \mathbf{Y}, \mathbf{X}) = \frac{p(\mathbf{Y} | \mathbf{X}, \mathbf{w}) p(\mathbf{w})}{p(\mathbf{Y} | \mathbf{X})}$$

Our problem setup

- Model: $f(\mathbf{x}; \mathbf{w})$ is a L-layer neural network with weights $\mathbf{w} = \{\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(L)}\}$
- Likelihood:

$$p(\mathbf{Y} \mid \mathbf{X}, \mathbf{w}) = \prod_{n=1}^{N} p(\mathbf{y}_n | \mathbf{x}_n, \mathbf{w}) = \prod_{n=1}^{N} p(\mathbf{y}_n | f(\mathbf{x}_n; \mathbf{w}))$$

- Prior:

$$p(\mathbf{w}) = \prod_{l=1}^{L} p(\mathbf{W}^{(i)}) = \prod_{i=1}^{L} \prod_{jk} \mathcal{N}(W_{jk}^{(l)} \mid 0, 1)$$

- Posterior (using Bayes's theorem):

$$p(\mathbf{w}|\mathbf{Y}, \mathbf{X}) = \frac{p(\mathbf{Y}|\mathbf{X}, \mathbf{w}) p(\mathbf{w})}{p(\mathbf{Y}|\mathbf{X})}$$

We can't compute the marginal likelihood ...

$$p(\mathbf{Y}|\mathbf{X}) = \int p(\mathbf{Y}|\mathbf{X}, \mathbf{w}) p(\mathbf{w}) \, \mathrm{d}\mathbf{w}$$

... and we can't compute the predictive distribution:

$$p(\mathbf{y}_*|\mathbf{x}_*, \mathbf{y}, \mathbf{X}) = \int p(\mathbf{y}_*|\mathbf{x}_*, \mathbf{w}) p(\mathbf{w}|\mathbf{Y}, \mathbf{X}) \, \mathrm{d}\mathbf{w}$$

Variational inference

Solutions:

- Collapse the posterior on the most likely value (Maximum-a-Posteriori or MAP)
- Approximate the intractable posterior:
  - Use Variational Inference
- Local approximation:
  - Use Laplace approximation (local approximation at the MAP solution)
- Sample from the intractable posterior:
  - Markov-Chain Monte-Carlo (Hamiltonian Monte-Carlo)

Variational inference

Solutions:

- Collapse the posterior on the most likely value (Maximum-a-Posteriori or MAP)
- Approximate the intractable posterior:
  - Use Variational Inference
- Local approximation:
  - Use Laplace approximation (local approximation at the MAP solution)
- Sample from the intractable posterior:
  - Markov-Chain Monte-Carlo (Hamiltonian Monte-Carlo)

Variational Inference

Main idea

Instead of trying to solve intractable integrals, let's solve an optimization problem.

Variational Inference

Main idea

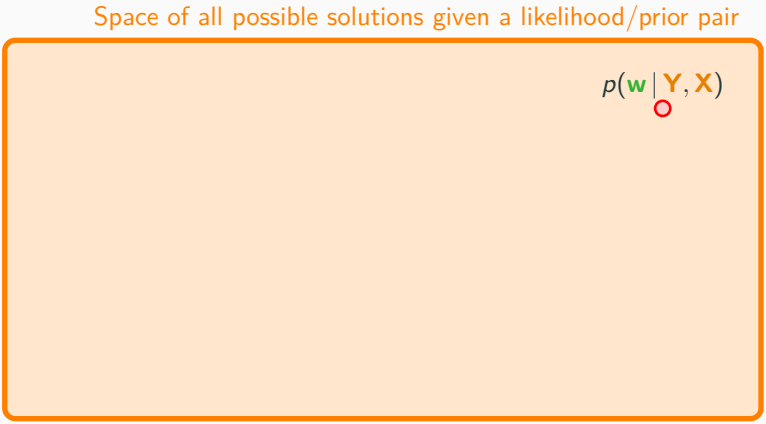
Instead of trying to solve intractable integrals, let's solve an optimization problem.

A very general recipe:

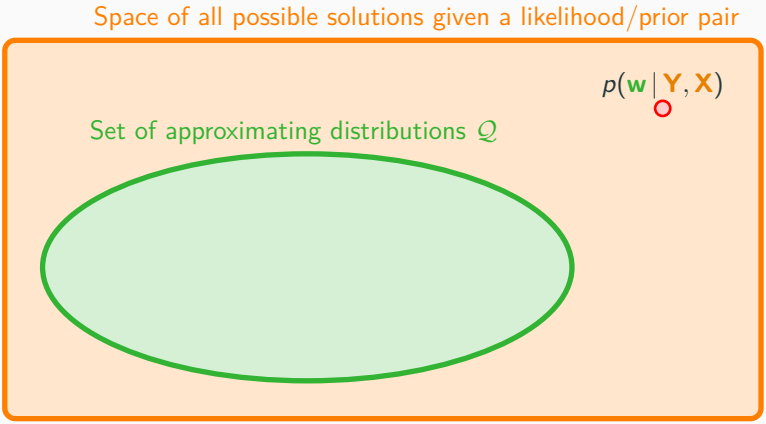
- Introduce a set $\mathcal{Q}$ of distributions $q(\mathbf{w})$
- Define an objective which measures the "distance" between an arbitrary distribution $q(\mathbf{w}) \in \mathcal{Q}$ and $p(\mathbf{w}|\mathbf{Y}, \mathbf{X})$
- In the set of possible solutions $\mathcal{Q}$, find the best $q(\mathbf{w})$ that minimizes the "distance" to $p(\mathbf{w}\,|\,\mathbf{Y}, \mathbf{X})$

  Interpret $q(\mathbf{w})$ as a distribution that approximates the intractable $p(\mathbf{w}|\mathbf{Y}, \mathbf{X})$

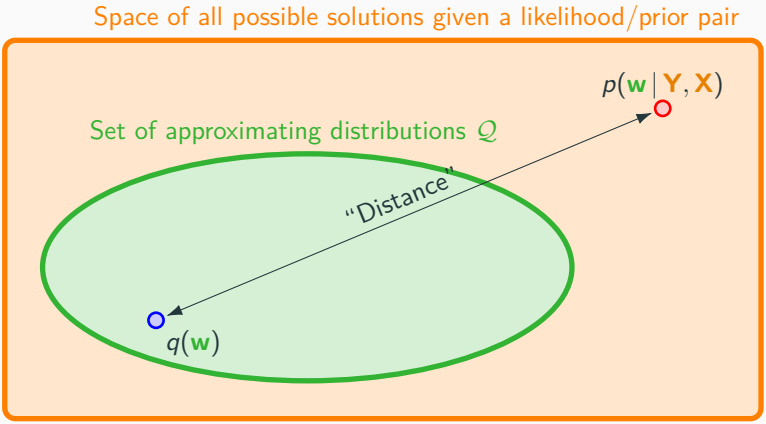
Visual illustration of Variational Inference



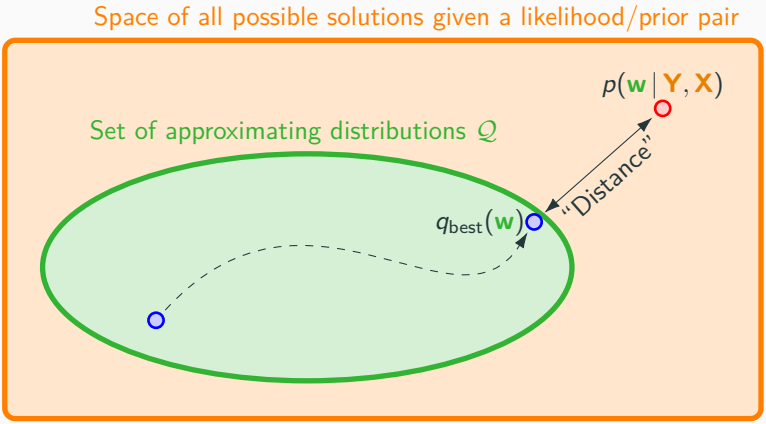
Visual illustration of Variational Inference



Visual illustration of Variational Inference



Visual illustration of Variational Inference



Variational Inference - Form of the approximation

- The *mean-field approach* imposes independent distributions for each component of $\mathbf{w}$

$$q(\mathbf{w}) = \prod_{l=1}^{L} \prod_{i=1}^{D_{in}^{(l)}} \prod_{j=1}^{D_{out}^{(l)}} q(W_{ij}^{(l)})$$

For notation purposes, all weights are concatenated in $\mathbf{w} \in \mathbb{R}^P$ (P is the number of weights in the network)

Variational Inference - Form of the approximation

- The *mean-field approach* imposes independent distributions for each component of $\mathbf{w}$

$$q(\mathbf{w}) = \prod_{l=1}^{L} \prod_{i=1}^{D_{in}^{(l)}} \prod_{j=1}^{D_{out}^{(l)}} q(W_{ij}^{(l)})$$

  For notation purposes, all weights are concatenated in $\mathbf{w} \in \mathbb{R}^P$ (P is the number of weights in the network)

- We start with Gaussian distributions

$$q(\mathbf{w}) = \prod_{i=1}^{P} q(w_i) = \prod_{i=1}^{P} \mathcal{N}(w_i | \mu_i, \sigma_i^2)$$

  μ_i, σ_i^2 are called *variational parameters* (for notation, they are collected into $\boldsymbol{\nu}$)

Variational Inference - Objective

- We will use the KL divergence to measure the "distance" between the two distributions,

$$\mathrm{KL}\left[q(\mathbf{w};\boldsymbol{\nu}) \parallel p(\mathbf{w}|\mathbf{Y},\mathbf{X})\right] = \int q(\mathbf{w};\boldsymbol{\nu}) \log \frac{q(\mathbf{w};\boldsymbol{\nu})}{p(\mathbf{w}|\mathbf{Y},\mathbf{X})} \, \mathrm{d}\mathbf{w} = \mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})}\left[\log \frac{q(\mathbf{w};\boldsymbol{\nu})}{p(\mathbf{w}|\mathbf{Y},\mathbf{X})}\right]$$

- A tractable objective to optimize $q(\mathbf{w};\boldsymbol{\nu})$ is obtained by manipulating the KL divergence

$$\mathrm{KL}\left[q(\mathbf{w};\boldsymbol{\nu}) \parallel p(\mathbf{w}|\mathbf{Y},\mathbf{X})\right] = -\mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})}\left[\log p(\mathbf{Y}|\mathbf{X},\mathbf{w})\right] + \mathrm{KL}\left[q(\mathbf{w};\boldsymbol{\nu}) \parallel p(\mathbf{w})\right] + \log p(\mathbf{Y}|\mathbf{X})$$

Variational Inference - Objective

- We will use the KL divergence to measure the "distance" between the two distributions,

$$\mathrm{KL}\left[q(\mathbf{w};\boldsymbol{\nu}) \parallel p(\mathbf{w}|\mathbf{Y},\mathbf{X})\right] = \int q(\mathbf{w};\boldsymbol{\nu}) \log \frac{q(\mathbf{w};\boldsymbol{\nu})}{p(\mathbf{w}|\mathbf{Y},\mathbf{X})} \, \mathrm{d}\mathbf{w} = \mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})}\left[\log \frac{q(\mathbf{w};\boldsymbol{\nu})}{p(\mathbf{w}|\mathbf{Y},\mathbf{X})}\right]$$

- A tractable objective to optimize $q(\mathbf{w};\boldsymbol{\nu})$ is obtained by manipulating the KL divergence

$$\mathrm{KL}\left[q(\mathbf{w};\boldsymbol{\nu}) \parallel p(\mathbf{w}|\mathbf{Y},\mathbf{X})\right] = -\mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})}\left[\log p(\mathbf{Y}|\mathbf{X},\mathbf{w})\right] + \mathrm{KL}\left[q(\mathbf{w};\boldsymbol{\nu}) \parallel p(\mathbf{w})\right] + \log p(\mathbf{Y}|\mathbf{X})$$

- The last term is the problematic one . . .

Variational Inference - Objective

- ... but manipulating the previous expression

$$\log p(\mathbf{Y}|\mathbf{X}) - \mathrm{KL}\left[q(\mathbf{w};\boldsymbol{\nu}) \parallel p(\mathbf{w}|\mathbf{Y},\mathbf{X})\right] = \underbrace{\mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})}\left[\log p(\mathbf{Y}|\mathbf{X},\mathbf{w})\right] - \mathrm{KL}\left[q(\mathbf{w};\boldsymbol{\nu}) \parallel p(\mathbf{w})\right]}_{\mathcal{L}_{\mathrm{ELBO}}}$$

  we have a computable objective

- Minimizing the original KL is equivalent to maximize $\mathcal{L}_{\mathrm{ELBO}}$

Variational Inference - Objective

- ... but manipulating the previous expression

$$\log p(\mathbf{Y}|\mathbf{X}) - \mathrm{KL}\left[q(\mathbf{w};\boldsymbol{\nu}) \parallel p(\mathbf{w}|\mathbf{Y},\mathbf{X})\right] = \underbrace{\mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})}\left[\log p(\mathbf{Y}|\mathbf{X},\mathbf{w})\right] - \mathrm{KL}\left[q(\mathbf{w};\boldsymbol{\nu}) \parallel p(\mathbf{w})\right]}_{\mathcal{L}_{\mathrm{ELBO}}}$$

  we have a computable objective
- Minimizing the original KL is equivalent to maximize $\mathcal{L}_{\mathrm{ELBO}}$

Note that $\mathrm{KL}\left[q(\mathbf{w};\boldsymbol{\nu}) \parallel p(\mathbf{w}|\mathbf{Y},\mathbf{X})\right] \geq 0$:

- The right hand side is a lower bound to the marginal likelihood (hence the name)
- If we can make $q(\mathbf{w};\boldsymbol{\nu})$ equal to the posterior, our objective will be equal to the marginal likelihood!

Variational Inference - Objective

Variational objective (a.k.a. evidence lower bound) to be maximized wrt $q(\mathbf{w}; \boldsymbol{\nu})$

$$\mathcal{L}_{\text{ELBO}} = \mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})} \left[\log p(\mathbf{Y}|\mathbf{X}, \mathbf{w})\right] - \mathrm{KL}\left[q(\mathbf{w}; \boldsymbol{\nu}) \| p(\mathbf{w})\right]$$

Variational Inference - Objective

Variational objective (a.k.a. evidence lower bound) to be maximized wrt $q(\mathbf{w}; \boldsymbol{\nu})$

$$\mathcal{L}_{\text{ELBO}} = \mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})} \left[\log p(\mathbf{Y}|\mathbf{X}, \mathbf{w})\right] - \mathrm{KL}\left[q(\mathbf{w}; \boldsymbol{\nu}) \| p(\mathbf{w})\right]$$

- First term is a model fitting term:

$$\mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})} \left[\log p(\mathbf{Y}|\mathbf{X}, \mathbf{w})\right]$$

  the higher the better the parameters drawn from $q(\mathbf{w}; \boldsymbol{\nu})$ are at modeling the labels

Variational Inference - Objective

Variational objective (a.k.a. evidence lower bound) to be maximized wrt $q(\mathbf{w}; \boldsymbol{\nu})$

$$\mathcal{L}_{\text{ELBO}} = \mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})} \left[\log p(\mathbf{Y}|\mathbf{X}, \mathbf{w})\right] - \mathrm{KL}\left[q(\mathbf{w}; \boldsymbol{\nu}) \| p(\mathbf{w})\right]$$

- First term is a model fitting term:

  $$\mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})} \left[\log p(\mathbf{Y}|\mathbf{X}, \mathbf{w})\right]$$

  the higher the better the parameters drawn from $q(\mathbf{w}; \boldsymbol{\nu})$ are at modeling the labels
- Second term is a regularization term:

  $$-\mathrm{KL}\left[q(\mathbf{w}; \boldsymbol{\nu}) \| p(\mathbf{w})\right]$$

  which penalizes $q(\mathbf{w}; \boldsymbol{\nu})$ deviating too much from the prior

Variational Inference - Computation

$$\mathcal{L}_{\text{ELBO}} = \mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})}\left[\log p(\mathbf{Y}|\mathbf{X},\mathbf{w})\right] - \mathrm{KL}\left[q(\mathbf{w};\boldsymbol{\nu})\|p(\mathbf{w})\right]$$

- The second term can be expressed analytically by using the expression of the KL divergence and it does **not** depend on the neural network architecture

Hoffman et al. (2013). *Stochastic Variational Inference*. JMLR

Variational Inference - Computation

$$\mathcal{L}_{\text{ELBO}} = \mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})}\left[\log p(\mathbf{Y}|\mathbf{X},\mathbf{w})\right] - \mathrm{KL}\left[q(\mathbf{w};\boldsymbol{\nu})\|p(\mathbf{w})\right]$$

- The second term can be expressed analytically by using the expression of the KL divergence and it does **not** depend on the neural network architecture
- The first expectation is never analytically available for Bayesian neural networks but it can be approximated using Monte Carlo integration:

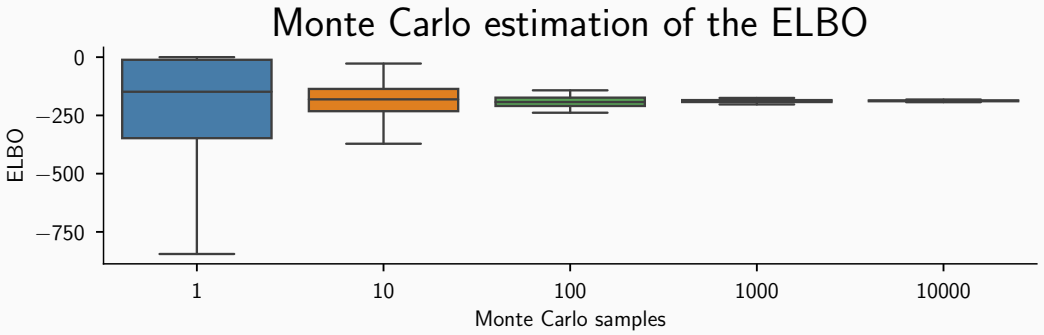
$$\mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})}\left[\log p(\mathbf{Y}|\mathbf{X},\mathbf{w})\right] \approx \frac{1}{N_{\text{MC}}}\sum_{i=1}^{N_{\text{MC}}} \log p(\mathbf{Y}|\mathbf{X},\tilde{\mathbf{w}}^{(h)}), \quad \tilde{\mathbf{w}}^{(h)} \sim q(\mathbf{w};\boldsymbol{\nu})$$

Hoffman et al. (2013). *Stochastic Variational Inference*. JMLR

Variational Inference - Computation

$$\mathcal{L}_{\text{ELBO}} = \frac{1}{N_{\text{MC}}} \sum_{i=1}^{N_{\text{MC}}} \log p(\mathbf{Y}|\mathbf{X}, \tilde{\mathbf{w}}^{(h)}) - \text{KL}\left[q(\mathbf{w}; \boldsymbol{\nu}) \| p(\mathbf{w})\right]$$

Remember: this estimator is unbiased and its variance shrinks $\propto 1/N_{\text{MC}}$, independently of the dimensionality.



Variational Inference - Optimization

We need to compute the gradients of the objective w.r.t the *variational parameters* $\boldsymbol{\nu}$

$$\nabla_{\boldsymbol{\nu}} \mathcal{L}_{\text{ELBO}} = \nabla_{\boldsymbol{\nu}} \mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})} \left[\log p(\mathbf{Y}|\mathbf{X}, \mathbf{w})\right] - \nabla_{\boldsymbol{\nu}} \mathrm{KL}\left[q(\mathbf{w};\boldsymbol{\nu}) \| p(\mathbf{w})\right]$$

- The second term is easy (everything is deterministic)

Variational Inference - Optimization

We need to compute the gradients of the objective w.r.t the *variational parameters* $\boldsymbol{\nu}$

$$\nabla_{\boldsymbol{\nu}} \mathcal{L}_{\text{ELBO}} = \nabla_{\boldsymbol{\nu}} \mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})} \left[\log p(\mathbf{Y}|\mathbf{X}, \mathbf{w})\right] - \nabla_{\boldsymbol{\nu}} \mathrm{KL}\left[q(\mathbf{w};\boldsymbol{\nu}) \| p(\mathbf{w})\right]$$

- The second term is easy (everything is deterministic)
- The first term requires a bit of work

Variational Inference - Optimization

- The Monte Carlo approximation

$$\mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})}\left[\log p(\mathbf{Y}|\mathbf{X},\mathbf{w})\right] \approx \frac{1}{N_{\mathrm{MC}}}\sum_{h=1}^{N_{\mathrm{MC}}} \log p(\mathbf{Y}|\mathbf{X},\tilde{\mathbf{w}}^{(h)})$$

  makes it tricky to optimize the objective
- With $q(\mathbf{w};\boldsymbol{\nu})$ fixed, when we resample $\mathbf{w}$ from $q(\mathbf{w};\boldsymbol{\nu})$ we obtain a different value!
- How can we make gradient updates to the μ_i, σ_i^2 parameters of $q(\mathbf{w};\boldsymbol{\nu})$?

Variational Inference - Optimization

- The Monte Carlo approximation

$$\mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})}\left[\log p(\mathbf{Y}|\mathbf{X},\mathbf{w})\right] \approx \frac{1}{N_{\mathrm{MC}}}\sum_{h=1}^{N_{\mathrm{MC}}} \log p(\mathbf{Y}|\mathbf{X},\tilde{\mathbf{w}}^{(h)})$$

  makes it tricky to optimize the objective
- With $q(\mathbf{w};\boldsymbol{\nu})$ fixed, when we resample $\mathbf{w}$ from $q(\mathbf{w};\boldsymbol{\nu})$ we obtain a different value!
- How can we make gradient updates to the μ_i, σ_i^2 parameters of $q(\mathbf{w};\boldsymbol{\nu})$?
- Answer: freeze the randomness within Monte Carlo!

Variational Inference - Reparameterization trick

$$\nabla_{\nu} \mathrm{E}_{q(\mathbf{w};\nu)} \log p(\mathbf{Y}|\mathbf{X}, \mathbf{w})$$

Idea:

- Samples of $\mathbf{w}$ can be obtained by a *deterministic transformation* ϕ of a random variable $\varepsilon \sim p(\varepsilon)$, such that $p(\varepsilon)$ does not depend on the variational parameters
- The variational parameters ν are parameters of the function ϕ
- Use the chain rule of differentiation to push the gradient though this function ϕ

Variational Inference - Reparameterization trick (Derivation)

Key observation

$$\nabla_{\boldsymbol{\nu}} \mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w}) = \mathrm{E}_{p(\boldsymbol{\varepsilon})} \nabla_{\boldsymbol{\nu}} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w})|_{\mathbf{w}=\phi(\boldsymbol{\varepsilon};\boldsymbol{\nu})}$$

Now, we can turn the *gradient of an expectation* into an *expectation of a gradient*.

$$\nabla_{\boldsymbol{\nu}} \mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w}) = \nabla_{\boldsymbol{\nu}} \mathrm{E}_{p(\boldsymbol{\varepsilon})} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w})|_{\mathbf{w}=\phi(\boldsymbol{\varepsilon};\boldsymbol{\nu})}$$

Variational Inference - Reparameterization trick (Derivation)

Key observation

$$\nabla_{\boldsymbol{\nu}} \mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w}) = \mathrm{E}_{p(\boldsymbol{\varepsilon})} \nabla_{\boldsymbol{\nu}} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w})|_{\mathbf{w}=\phi(\boldsymbol{\varepsilon};\boldsymbol{\nu})}$$

Now, we can turn the *gradient of an expectation* into an *expectation of a gradient*.

$$\begin{aligned}\nabla_{\boldsymbol{\nu}} \mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w}) &= \nabla_{\boldsymbol{\nu}} \mathrm{E}_{p(\boldsymbol{\varepsilon})} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w})|_{\mathbf{w}=\phi(\boldsymbol{\varepsilon};\boldsymbol{\nu})} \\ &= \mathrm{E}_{p(\boldsymbol{\varepsilon})} \left[\nabla_{\boldsymbol{\nu}} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w})|_{\mathbf{w}=\phi(\boldsymbol{\varepsilon};\boldsymbol{\nu})}\right]\end{aligned}$$

Variational Inference - Reparameterization trick (Derivation)

Key observation

$$\nabla_{\boldsymbol{\nu}} \mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w}) = \mathrm{E}_{p(\boldsymbol{\varepsilon})} \nabla_{\boldsymbol{\nu}} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w})|_{\mathbf{w}=\phi(\boldsymbol{\varepsilon};\boldsymbol{\nu})}$$

Now, we can turn the *gradient of an expectation* into an *expectation of a gradient*.

$$\begin{aligned}
\nabla_{\boldsymbol{\nu}} \mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w}) &= \nabla_{\boldsymbol{\nu}} \mathrm{E}_{p(\boldsymbol{\varepsilon})} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w})|_{\mathbf{w}=\phi(\boldsymbol{\varepsilon};\boldsymbol{\nu})} \\
&= \mathrm{E}_{p(\boldsymbol{\varepsilon})} \left[\nabla_{\boldsymbol{\nu}} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w})|_{\mathbf{w}=\phi(\boldsymbol{\varepsilon};\boldsymbol{\nu})}\right] \\
&= \mathrm{E}_{p(\boldsymbol{\varepsilon})} \left[\nabla_{\mathbf{w}} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w})|_{\mathbf{w}=\phi(\varepsilon;\boldsymbol{\nu})} \nabla_{\boldsymbol{\nu}} \mathbf{w}\right]
\end{aligned}$$

Variational Inference - Reparameterization trick (Derivation)

Key observation

$$\nabla_{\boldsymbol{\nu}} \mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w}) = \mathrm{E}_{p(\boldsymbol{\varepsilon})} \nabla_{\boldsymbol{\nu}} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w})|_{\mathbf{w}=\phi(\boldsymbol{\varepsilon};\boldsymbol{\nu})}$$

Now, we can turn the *gradient of an expectation* into an *expectation of a gradient*.

$$\begin{aligned}
\nabla_{\boldsymbol{\nu}} \mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w}) &= \nabla_{\boldsymbol{\nu}} \mathrm{E}_{p(\boldsymbol{\varepsilon})} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w})|_{\mathbf{w}=\phi(\boldsymbol{\varepsilon};\boldsymbol{\nu})} \\
&= \mathrm{E}_{p(\boldsymbol{\varepsilon})} \left[\nabla_{\boldsymbol{\nu}} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w})|_{\mathbf{w}=\phi(\boldsymbol{\varepsilon};\boldsymbol{\nu})}\right] \\
&= \mathrm{E}_{p(\boldsymbol{\varepsilon})} \left[\nabla_{\mathbf{w}} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w})|_{\mathbf{w}=\phi(\varepsilon;\boldsymbol{\nu})} \nabla_{\boldsymbol{\nu}} \phi(\boldsymbol{\varepsilon};\boldsymbol{\nu})\right]
\end{aligned}$$

Variational Inference - Reparameterization trick (Derivation)

Key observation

$$\nabla_{\boldsymbol{\nu}} \mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w}) = \mathrm{E}_{p(\boldsymbol{\varepsilon})} \nabla_{\boldsymbol{\nu}} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w})|_{\mathbf{w}=\phi(\boldsymbol{\varepsilon};\boldsymbol{\nu})}$$

Now, we can turn the *gradient of an expectation* into an *expectation of a gradient*.

$$\begin{aligned}
\nabla_{\boldsymbol{\nu}} \mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w}) &= \nabla_{\boldsymbol{\nu}} \mathrm{E}_{p(\boldsymbol{\varepsilon})} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w})|_{\mathbf{w}=\phi(\boldsymbol{\varepsilon};\boldsymbol{\nu})} \\
&= \mathrm{E}_{p(\boldsymbol{\varepsilon})} \left[\nabla_{\boldsymbol{\nu}} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w})|_{\mathbf{w}=\phi(\boldsymbol{\varepsilon};\boldsymbol{\nu})} \right] \\
&= \mathrm{E}_{p(\boldsymbol{\varepsilon})} \left[\nabla_{\mathbf{w}} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w})|_{\mathbf{w}=\phi(\varepsilon;\boldsymbol{\nu})} \nabla_{\boldsymbol{\nu}} \phi(\boldsymbol{\varepsilon};\boldsymbol{\nu}) \right] \\
&\approx \frac{1}{N_{\mathrm{MC}}} \sum_{h=1}^{N_{\mathrm{MC}}} \nabla_{\mathbf{w}} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w})|_{\mathbf{w}=\phi(\tilde{\boldsymbol{\varepsilon}}^{(h)};\boldsymbol{\nu})} \nabla_{\boldsymbol{\nu}} \phi(\tilde{\boldsymbol{\varepsilon}}^{(h)};\boldsymbol{\nu}), \quad \tilde{\boldsymbol{\varepsilon}}^{(h)} \sim p(\boldsymbol{\varepsilon})
\end{aligned}$$

Variational Inference - Reparameterization trick (Derivation)

Key observation

$$\nabla_{\boldsymbol{\nu}} \mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w}) = \mathrm{E}_{p(\boldsymbol{\varepsilon})} \nabla_{\boldsymbol{\nu}} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w})|_{\mathbf{w}=\phi(\boldsymbol{\varepsilon};\boldsymbol{\nu})}$$

Now, we can turn the *gradient of an expectation* into an *expectation of a gradient*.

$$\begin{aligned}
\nabla_{\boldsymbol{\nu}} \mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w}) &= \nabla_{\boldsymbol{\nu}} \mathrm{E}_{p(\boldsymbol{\varepsilon})} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w})|_{\mathbf{w}=\phi(\boldsymbol{\varepsilon};\boldsymbol{\nu})} \\
&= \mathrm{E}_{p(\boldsymbol{\varepsilon})} \left[\nabla_{\boldsymbol{\nu}} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w})|_{\mathbf{w}=\phi(\boldsymbol{\varepsilon};\boldsymbol{\nu})}\right] \\
&= \mathrm{E}_{p(\boldsymbol{\varepsilon})} \left[\nabla_{\mathbf{w}} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w})|_{\mathbf{w}=\phi(\varepsilon;\boldsymbol{\nu})} \nabla_{\boldsymbol{\nu}} \phi(\boldsymbol{\varepsilon};\boldsymbol{\nu})\right] \\
&\approx \frac{1}{N_{\mathrm{MC}}} \sum_{h=1}^{N_{\mathrm{MC}}} \nabla_{\mathbf{w}} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w})|_{\mathbf{w}=\phi(\tilde{\boldsymbol{\varepsilon}}^{(h)};\boldsymbol{\nu})} \nabla_{\boldsymbol{\nu}} \phi(\tilde{\boldsymbol{\varepsilon}}^{(h)};\boldsymbol{\nu}), \quad \tilde{\boldsymbol{\varepsilon}}^{(h)} \sim p(\boldsymbol{\varepsilon})
\end{aligned}$$

Good news: if you use any auto-diff tool (PyTorch, Tensorflow, JAX, NumPyro, etc.), you will never compute this gradients manually.

Variational Inference - Reparameterization trick (Properties)

$$\nabla_{\boldsymbol{\nu}} \mathrm{E}_{q(\mathbf{w};\boldsymbol{\nu})} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w}) \approx \frac{1}{N_{\mathrm{MC}}} \sum_{h=1}^{N_{\mathrm{MC}}} \nabla_{\mathbf{w}} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w})|_{\mathbf{w}=\phi(\tilde{\boldsymbol{\varepsilon}}^{(h)};\boldsymbol{\nu})} \nabla_{\boldsymbol{\nu}} \phi(\tilde{\boldsymbol{\varepsilon}}^{(h)};\boldsymbol{\nu}), \quad \tilde{\boldsymbol{\varepsilon}}^{(h)} \sim p(\boldsymbol{\varepsilon})$$

- Estimation of the gradients is unbiased
- Need to be able to sample from $p(\boldsymbol{\varepsilon})$, but not from $q(\mathbf{w};\boldsymbol{\nu})$
- The likelihood $p(\mathbf{Y}|\mathbf{X},\mathbf{w})$ must be differentiable $\rightarrow$
  The neural network $f(\mathbf{x};\mathbf{w})$ must be differentiable

Variational Inference with Stochastic Optimization

$$\widetilde{\mathcal{L}_{\text{ELBO}}} = \frac{1}{N_{\text{MC}}} \sum_{h=1}^{N_{\text{MC}}} \log p(\mathbf{Y}|\mathbf{X}, \tilde{\mathbf{w}}^{(h)}) - \text{KL}\left[q(\mathbf{w}; \boldsymbol{\nu}) \| p(\mathbf{w})\right]$$

- We can get unbiased estimate by selecting M out of N data

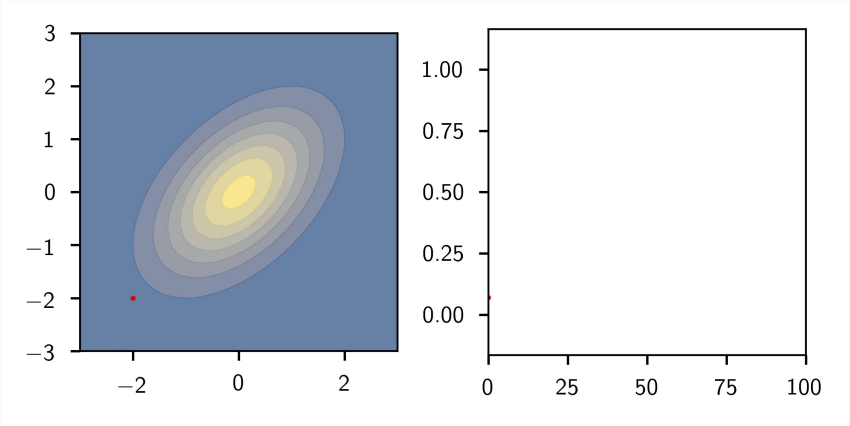
$$\log p(\mathbf{Y}|\mathbf{X}, \tilde{\mathbf{w}}^{(h)}) \approx \frac{N}{M} \sum_{i \in \text{minibatch}} \log p(\mathbf{y}_i|\mathbf{x}_i, \tilde{\mathbf{w}}^{(h)})$$

- We can use stochastic gradient optimization of our approximate variational objective with $\tilde{\mathbf{w}}^{(h)} = f(\tilde{\boldsymbol{\varepsilon}}^{(h)}; \boldsymbol{\nu})$ and $\tilde{\boldsymbol{\varepsilon}}^{(h)} \sim p(\boldsymbol{\varepsilon})$

Hoffman et al. (2013). *Stochastic Variational Inference*. JMLR

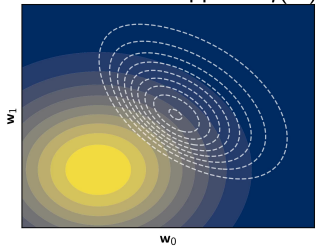
Variational Inference with Stochastic Optimization

$$\boldsymbol{\nu}' = \boldsymbol{\nu} + \frac{\alpha_t}{2} \nabla_{\boldsymbol{\nu}} (\widetilde{\mathcal{L}_{\text{ELBO}}}) \qquad \alpha_t \to 0$$

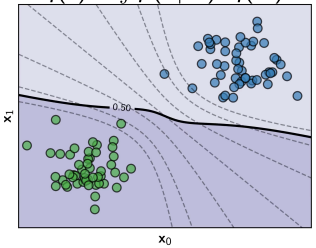


A simple animation of stochastic variational inference in practice

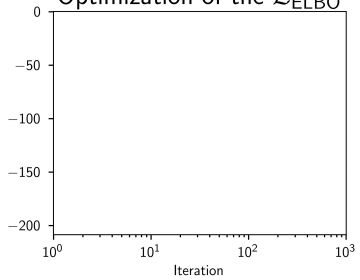
True posterior and the variational approx. $q(\mathbf{w})$



Predictive posterior $q(\mathbf{f}) = \int p(\mathbf{f} \mid \mathbf{w}) \mathrm{d}q(\mathbf{w})$



Optimization of the $\mathcal{L}_{\text{ELBO}}$



Advances in Variational Inference for Bayesian Deep Learning

Variance analysis of Stochastic Variational Inference

Let $L_i = \log p(y_i \mid \mathbf{x}_i, \mathbf{w})$ the likelihood contribution from $\{y_i, \mathbf{x}_i\}$ in minibatch of size M

$$\text{Var}\left[\log p(\mathbf{Y}|\mathbf{X}, \mathbf{w})\right] = N^2 \left(\frac{1}{M} \text{Var}[L_i] + \frac{M-1}{M} \text{Cov}[L_i, L_j] \right)$$

Kingma et al. (2015). *Variational Dropout and the Local Reparameterization Trick*. NeurIPS

Variance analysis of Stochastic Variational Inference

Let $L_i = \log p(y_i \mid \mathbf{x}_i, \mathbf{w})$ the likelihood contribution from $\{y_i, \mathbf{x}_i\}$ in minibatch of size M

$$\text{Var}\left[\log p(\mathbf{Y}|\mathbf{X}, \mathbf{w})\right] = N^2 \left(\frac{1}{M} \text{Var}[L_i] + \frac{M-1}{M} \text{Cov}[L_i, L_j] \right)$$

Kingma et al. (2015). *Variational Dropout and the Local Reparameterization Trick*. NeurIPS

Variance analysis of Stochastic Variational Inference

Let $L_i = \log p(y_i \,|\, \mathbf{x}_i, \mathbf{w})$ the likelihood contribution from $\{y_i, \mathbf{x}_i\}$ in minibatch of size M

$$\text{Var}\left[\log p(\mathbf{Y}|\mathbf{X}, \mathbf{w})\right] = N^2 \left(\frac{1}{M} \text{Var}[L_i] + \frac{M-1}{M} \text{Cov}[L_i, L_j] \right)$$

Solutions:

- Make $\text{Cov}[L_i, L_j] = 0$ by resampling $p(\epsilon)$ for every data-point (for each layer, $N_{MC} \times M \times D_{in} \times D_{out}$ times)

  ▶ **Computational intractable**

Kingma et al. (2015). *Variational Dropout and the Local Reparameterization Trick*. NeurIPS

Variance analysis of Stochastic Variational Inference

Let $L_i = \log p(y_i \,|\, \mathbf{x}_i, \mathbf{w})$ the likelihood contribution from $\{y_i, \mathbf{x}_i\}$ in minibatch of size M

$$\text{Var}\left[\log p(\mathbf{Y}|\mathbf{X}, \mathbf{w})\right] = N^2 \left(\frac{1}{M} \text{Var}[L_i] + \frac{M-1}{M} \text{Cov}[L_i, L_j] \right)$$

Solutions:

- Make $\text{Cov}[L_i, L_j] = 0$ by resampling $p(\epsilon)$ for every data-point (for each layer, $N_{MC} \times M \times D_{in} \times D_{out}$ times)

  ▶ **Computational intractable**

- Move the stochasticity from the weights to the activations

  ▶ **The local reparameterization trick**

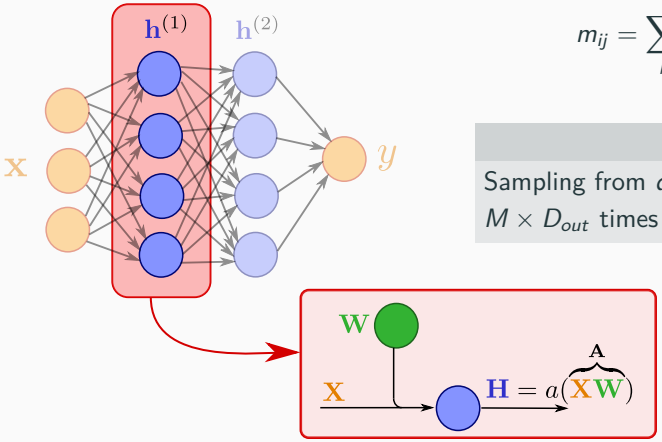
Kingma et al. (2015). *Variational Dropout and the Local Reparameterization Trick*. NeurIPS

The local reparameterization trick

If $q(W_{ij}) = \mathcal{N}(W_{ij} \mid \mu_{ij}, \sigma_{ij}^2)$,
then $q(A_{ij}) = \mathcal{N}(A_{ij} \mid m_{ij}, s_{ij}^2)$, with

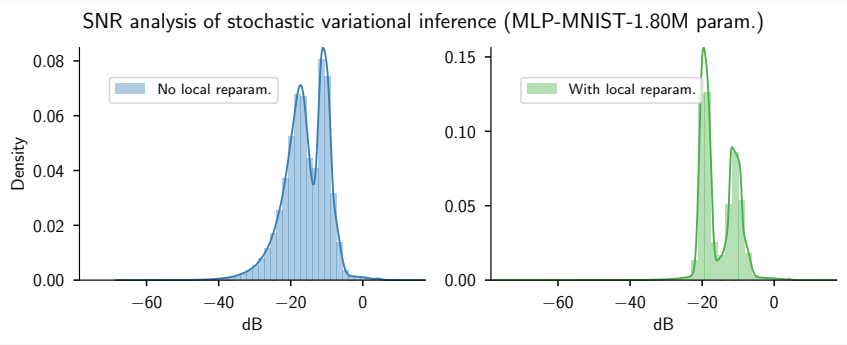
$$m_{ij} = \sum_k X_{ik}\mu_{kj}, \qquad s_{ij}^2 = \sum_k X_{ik}^2 \sigma_{kj}^2$$

Sampling from $q(\mathbf{A})$ requires only to sample only $M \times D_{out}$ times from $q(\epsilon)$.



Kingma et al. (2015). *Variational Dropout and the Local Reparameterization Trick*. NeurIPS

The local reparameterization trick (summary)



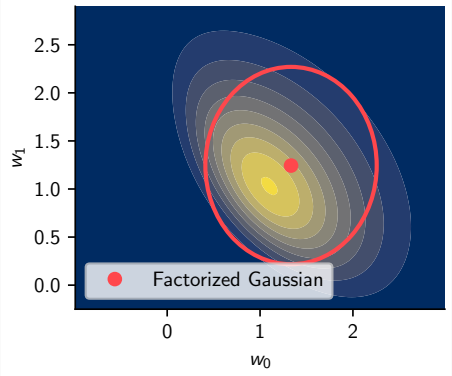
Practical improvements for training due to lower stochastic variance (speed-up convergence of $\mathcal{L}_{\text{ELBO}}$ or smaller batch-size)

Adapted from github.com/JavierAntoran/Bayesian-Neural-Networks

Richer family of parameterizations

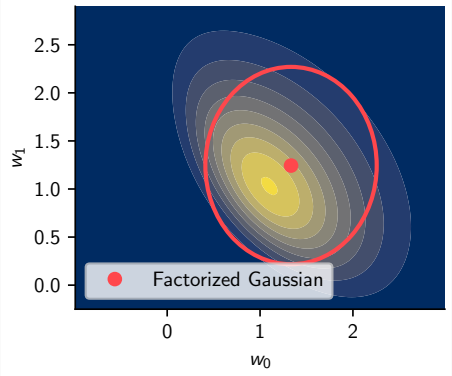
Problem:

Mean-field Gaussians, albeit very simple to implement, is generally a rough approximation.

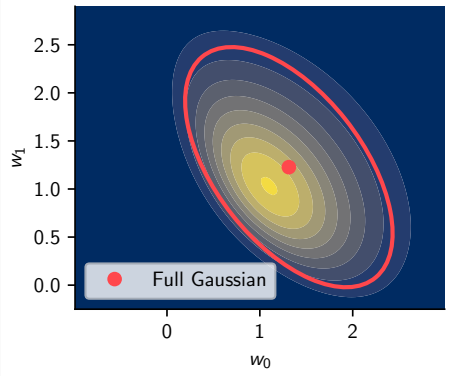


Richer family of parameterizations

Problem:
Mean-field Gaussians, albeit very simple to implement, is generally a rough approximation.



Solution:
Introduce correlations among weights (i.e. Gaussian with non-diagonal covariance)



Here is where we deviate from standard VI methods

Working with full covariance is completely intractable

Assume we have one hidden layer with weights matrix $\mathbf{W}$ of dimension 256×256. From reshaping $\mathbf{W}$ in vector form $\mathbf{w} \in \mathbb{R}^{65536}$, the covariance $\mathbf{\Sigma}$ is a matrix 65536×65536.

Memory required: $\sim 16\,\text{GB}$ (only to store $\mathbf{\Sigma}$, in single-point precision)

Here is where we deviate from standard VI methods

Working with full covariance is completely intractable

Assume we have one hidden layer with weights matrix $\mathbf{W}$ of dimension 256×256. From reshaping $\mathbf{W}$ in vector form $\mathbf{w} \in \mathbb{R}^{65536}$, the covariance $\boldsymbol{\Sigma}$ is a matrix 65536×65536.

Memory required: $\sim 16\,\text{GB}$ (only to store $\boldsymbol{\Sigma}$, in single-point precision)

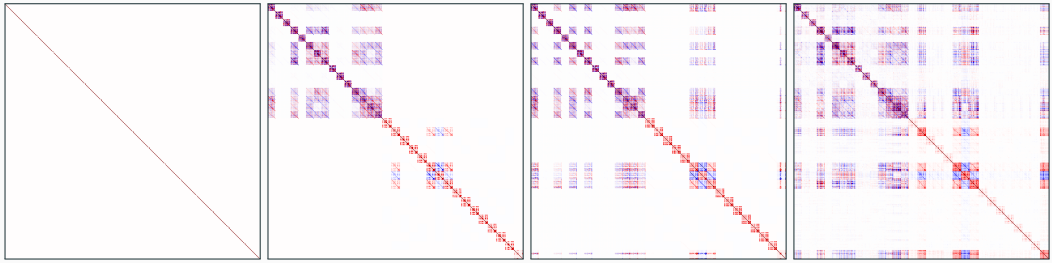
Simple solution:

Use mean-field for columns and full covariance for the rows of $\mathbf{W}_{:,i}$.

$$q(\mathbf{W}) = \prod_{i=1}^{256} q(\mathbf{W}_{:,i}) = \prod_{i=1}^{256} \mathcal{N}(\mathbf{W}_{:,i} \,|\, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$$

where now $\boldsymbol{\Sigma}_i$ is a 256×256 covariance matrix.

Putting some structure in the covariance matrix



(a) Factorized **(b)** MVG **(c)** Block Triangular **(d)** Full

Zhang et al. (2018). *Noisy Natural Gradient as Variational Inference*. ICML

Matrix Variate Gaussian distribution

$$q(\mathbf{W}) = \mathcal{MN}(\mathbf{W} \mid \mathbf{M}, \mathbf{U}, \mathbf{V}) = \frac{\exp(\frac{1}{2}\mathrm{Tr}[\mathbf{V}^{-1}(\mathbf{W}-\mathbf{M})^\top \mathbf{U}^{-1}(\mathbf{W}-\mathbf{M})])}{(2\pi)^{D_{in}D_{out}/2}|\mathbf{V}|^{D_{out}/2}|\mathbf{U}|^{D_{in}/2}}$$

$\mathbf{M} \in \mathbb{R}^{D_{in} \times D_{out}}$ is the mean, $\mathbf{U} \in \mathbb{R}^{D_{in} \times D_{in}}$ is the covariance matrix among rows and $\mathbf{V} \in \mathbb{R}^{D_{out} \times D_{out}}$ is the covariance matrix among columns.

Connected with Gaussian distribution:

$$\mathrm{vec}(\mathbf{W}) \sim \mathcal{N}(\mathrm{vec}(\mathbf{M}), \mathbf{V} \otimes \mathbf{U})$$

Louizos and Welling (2016). *Structured and Efficient Variational Deep Learning with Matrix Gaussian Posteriors*. ICML

Advances in computational efficient variational approximations

Matrix Variate Gaussian distribution

$$q(\mathbf{W}) = \mathcal{MN}(\mathbf{W} \,|\, \mathbf{M}, \mathbf{U}, \mathbf{V}) = \frac{\exp(\frac{1}{2}\mathrm{Tr}[\mathbf{V}^{-1}(\mathbf{W}-\mathbf{M})^\top \mathbf{U}^{-1}(\mathbf{W}-\mathbf{M})])}{(2\pi)^{D_{in}D_{out}/2}|\mathbf{V}|^{D_{out}/2}|\mathbf{U}|^{D_{in}/2}}$$

$\mathbf{M} \in \mathbb{R}^{D_{in} \times D_{out}}$ is the mean, $\mathbf{U} \in \mathbb{R}^{D_{in} \times D_{in}}$ is the covariance matrix among rows and $\mathbf{V} \in \mathbb{R}^{D_{out} \times D_{out}}$ is the covariance matrix among columns.

Reparameterization trick for $\mathbf{A} = \mathbf{XW}$:

$$q(\mathbf{A}) = \mathcal{MN}(\mathbf{XM}, \mathbf{XUX}^\top, \mathbf{V})$$

Louizos and Welling (2016). *Structured and Efficient Variational Deep Learning with Matrix Gaussian Posteriors*. ICML

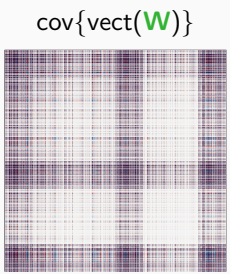
Structured posterior with connection to kernel methods

Impose a structure on the weight matrix inspired by scalable kernel methods

$$\mathbf{W} = \mathbf{S}_1 \, \mathbf{H} \, \mathbf{g} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \, \mathbf{H} \, \mathbf{S}_2$$

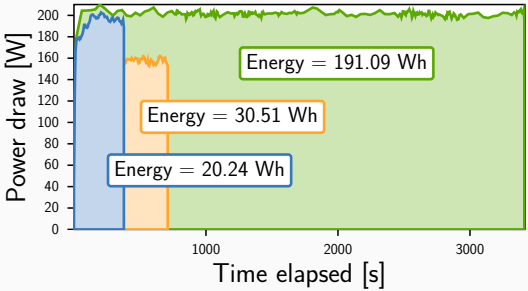
Benefit:

- Space complexity for storing $\mathbf{W}$: $\mathcal{O}(D^2) \longrightarrow \mathcal{O}(D)$.
- Time complexity for $\mathbf{Wx}$: $\mathcal{O}(D^2) \longrightarrow \mathcal{O}(D \log D)$.



Rossi at al. (2020). *Walsh-Hadamard Variational Inference for Bayesian Deep Learning*. NeurIPS

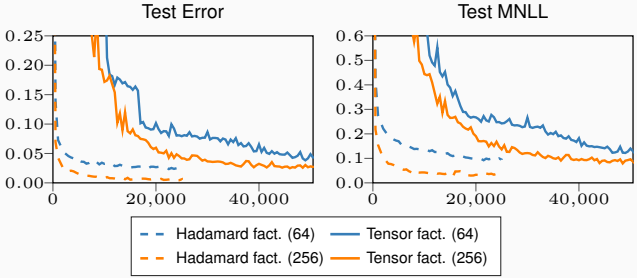
Keeping an eye to energy efficiency



Comparison between Monte Carlo dropout (●), Matrix Variate Gaussian (●) and Hadamard factorization (●).

Low rank factorization with variational inference

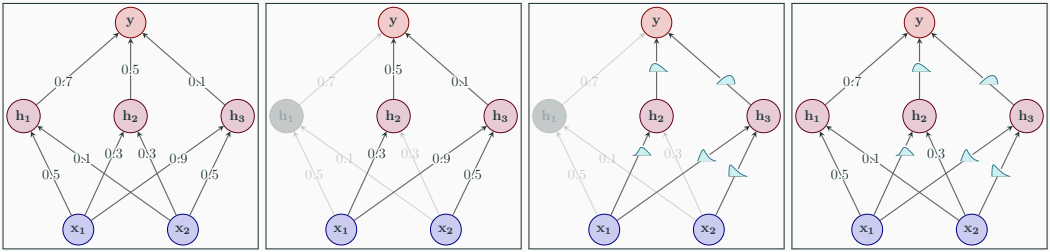
Perfoming tensor decomposition with variational inference is not straightforward.



The redundant variational parameterization induced by the tensor cores makes the optimization landscapes highly multi-modal, thus leading to slow convergence.

Rossi at al. (2020). *Walsh-Hadamard Variational Inference for Bayesian Deep Learning*. NeurIPS

Imposing low-rank structure by doing inference on a subset of weights



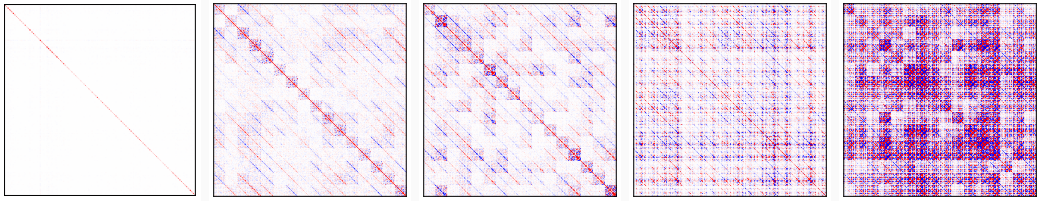
(a) Point estimation **(b)** Subnetwork selection **(c)** Bayesian inference **(d)** Prediction

(a) Train a neural network to a MAP solution
(b) Identify a small subset of the weights
(c) Estimate a posterior distribution over the selected subset
(d) Predict using the mix of Bayesian and deterministic weights

Daxberger et al. (2021). Bayesian Deep Learning via Subnetwork Inference. ICML

Complex covariances can raise from mean-field and depth

Covariance heatmap for mean-field approximate posteriors trained on FashionMNIST.



(a) One weight matrix.

(b) 5-layers (linear)

(c) 10-layers (linear)

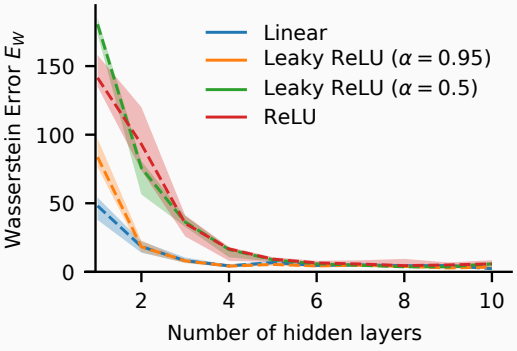
(d) 5-layers (Leaky ReLU)

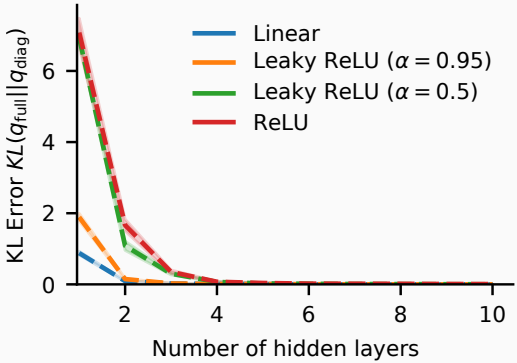
(e) 10-layers (Leaky ReLU)

Farquhar et al. (2020). *Liberty or Depth: Deep Bayesian Neural Nets Do Not Need Complex Weight Posterior Approximations*. NeurIPS

The tradeoff depth/structure

For any sufficiently deep and wide neural network, there exists a mean-field distribution which induces the same distribution over function values as that induced by the posterior predictive





Farquhar et al. (2020). *Liberty or Depth: Deep Bayesian Neural Nets Do Not Need Complex Weight Posterior Approximations.* NeurIPS

Are structured posteriors worth? (Use case: CIFAR10)

For low-medium depth models having structured posterior seems to be important.

Architecture	**Covariance**	**Accuracy** (↑)	**NLL** (↓)	**ECE** (↓)
AlexNet	Diagonal	75.5%	0.703	0.016
(low depth)	Low-rank (WHVI)	88.5%	0.490	0.009
ResNet-18	Diagonal	84.3%	0.477	0.040
(medium depth)	Low-rank (WHVI)	86.4%	0.616	0.029

Farquhar et al. (2020). *Liberty or Depth: Deep Bayesian Neural Nets Do Not Need Complex Weight Posterior Approximations*. NeurIPS

Rossi at al. (2020). *Walsh-Hadamard Variational Inference for Bayesian Deep Learning*. NeurIPS

Osawa et al. (2019). *Practical Deep Learning with Bayesian Principles*. NeurIPS

Are structured posteriors worth? (Use case: ImageNet)

The importance of structured covariance seems to be diminished in very large-scale models.

Architecture	**Covariance**	**Accuracy** (↑)	**NLL** (↓)	**ECE** (↓)
DenseNet-161	Diagonal	78.6%	0.86	0.046
	Low-rank	78.6%	0.83	0.020
ResNet-152	Diagonal	80.0%	0.86	0.057
	Low-rank	79.1%	0.82	0.028

Farquhar et al. (2020). *Liberty or Depth: Deep Bayesian Neural Nets Do Not Need Complex Weight Posterior Approximations*. NeurIPS

References i

Introduction to Variational Inference

- Jordan et al. (1999). *An Introduction to Variational Methods for Graphical Models*. Mach. Learn.
- Hoffman et al. (2013). *Stochastic Variational Inference*. JMLR
- Ranganath et al. (2014). *Black Box Variational Inference*. AISTATS
- Blei et al. (2017). *Variational Inference: A Review for Statisticians*. JASA

References ii

Monte-Carlo Dropout for Bayesian Neural Networks and follow-up

- Srivastava et al. (2014). *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*, JMLR
- Kingma et al. (2015). *Variational Dropout and the Local Reparameterization Trick*. NeurIPS
- Gal (2016). *Uncertainty in Deep Learning*. University of Cambridge (PhD Thesis)
- Gal and Ghahramani (2016). *Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference*. ICLR Workshop
- Gal and Ghahramani (2016). *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning*. ICML
- Kendall and Gal (2017). *What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?*. NeurIPS
- Li and Gal (2017). *Dropout Inference in Bayesian Neural Networks with Alpha-divergences*. ICML

References iii

- Hron et al. (2017). *Variational Gaussian Dropout is not Bayesian*. NeurIPS Workshop
- Hron et al (2018). *Variational Bayesian Dropout: Pitfalls and Fixes*. ICML

References iv

Variational Inference for Neural Networks

- Graves (2011). *Practical Variational Inference for Neural Networks*. NeurIPS
- Rezende et al. (2014). *Stochastic Backpropagation and Approximate Inference in Deep Generative Models*. ICML
- Hernández-Lobato et al. (2015). *Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks*. ICML
- Blundell et al. (2015). *Weight Uncertainty in Neural Networks*. ICML
- Rezende et al. (2015). *Variational Inference with Normalizing Flows*. ICML
- Louizos and Welling (2016). *Structured and Efficient Variational Deep Learning with Matrix Gaussian Posteriors*. ICML
- Kingma et al. (2016). *Improving Variational Inference with Inverse Autoregressive Flow*. NeurIPS

References v

- Liu et al. (2016). *Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm*. NeurIPS
- Miller et al. (2016). *Variational Boosting: Iteratively Refining Posterior Approximations*. ICML
- Louizos and Welling (2017). *Multiplicative Normalizing Flows for Variational Bayesian Neural Networks*. ICML
- Sun et al. (2017). *Learning Structured Weight Uncertainty in Bayesian Neural Networks*. AISTATS
- Khan et al. (2018). *Fast and Scalable Bayesian Deep Learning by Weight-Perturbation in ADAM*. ICML
- Rossi et al. (2018). *Good Initializations of Variational Bayes for Deep Models*. ICML
- Zhang et al. (2018). *Noisy Natural Gradient as Variational Inference*. ICML
- Ghosh et al. (2018). *Structured Variational Learning of Bayesian Neural Networks with Horseshoe Priors*. ICML
- Osawa et al. (2019). *Practical Deep Learning with Bayesian Principles*. NeurIPS

- Sun et al. (2019). *Functional Variational Bayesian Neural Networks*. ICLR
- Farquhar et al. (2020). *Liberty or Depth: Deep Bayesian Neural Nets Do Not Need Complex Weight Posterior Approximations*. NeurIPS
- Rossi et al. (2020). *Walsh-Hadamard Variational Inference for Bayesian Deep Learning*. NeurIPS
- Daxberger et al. (2021). *Bayesian Deep Learning via Subnetwork Inference*. ICML