Bayesian deep learning

Juho Lee
Grad school of AI, KAIST
Samsung AI expert course

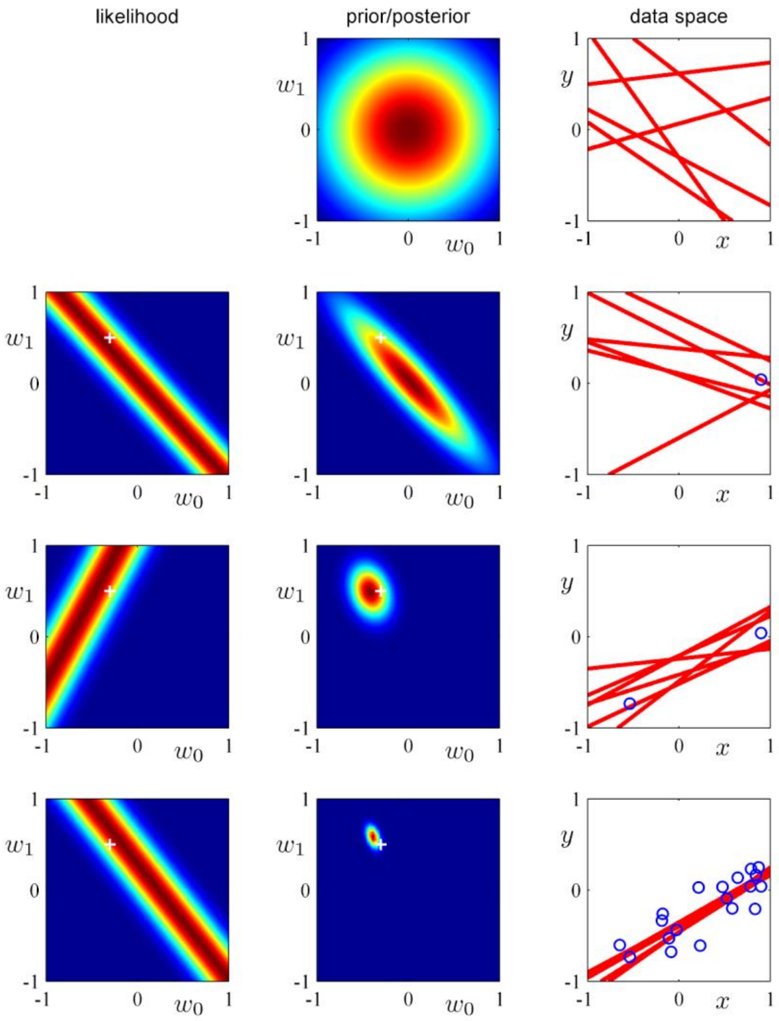
Why Bayesian deep learning?

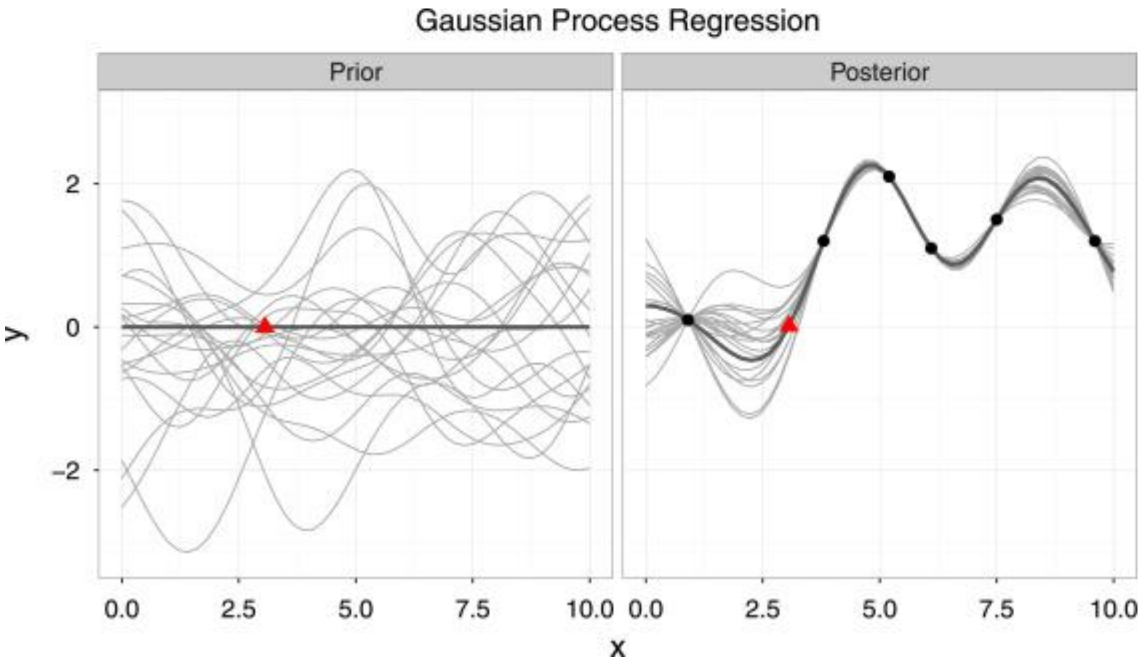
Bayesian methods

- Bayes' rule

$$\overset{\text{posterior}}{P(Y|X)} = \frac{\overset{\text{likelihood}}{P(X|Y)}\,\overset{\text{prior}}{P(X)}}{P(X)}$$

Bayesian methods for regressions



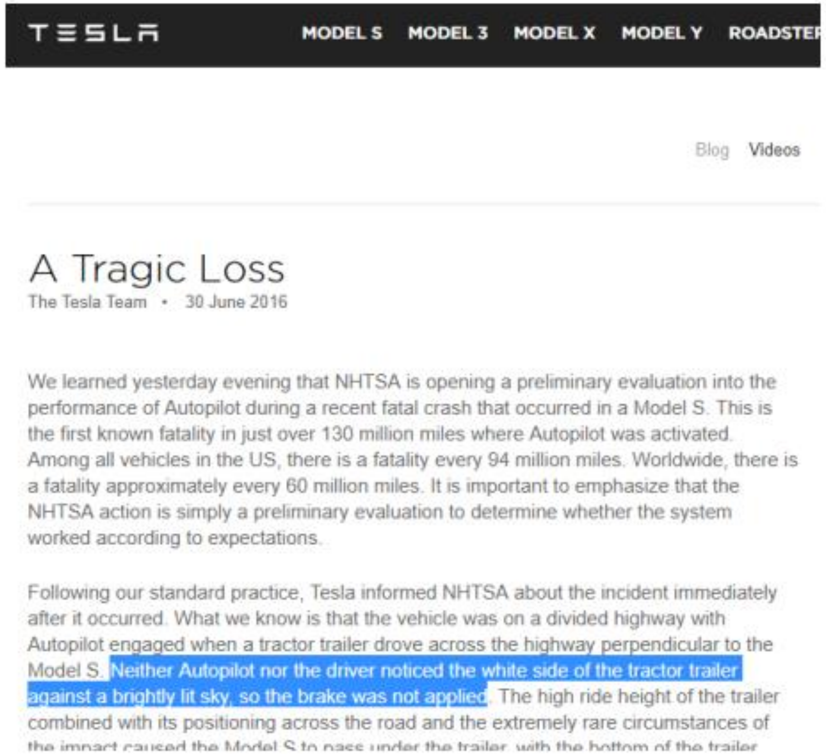


Bayesian deep learning?

- Deep learning with Bayesian methods
- But with much larger models (higher dimensional parameters) and larger datasets!

$$\underbrace{P(Y|X)}^{\text{posterior}} = \frac{\overbrace{P(X|Y)}^{\text{likelihood}}\overbrace{P(X)}^{\text{prior}}}{P(X)}$$

Why Bayesian (deep) learning?
Example: autonomous driving



TESLA MODEL S MODEL 3 MODEL X MODEL Y ROADSTER

Blog Videos

A Tragic Loss

The Tesla Team • 30 June 2016

We learned yesterday evening that NHTSA is opening a preliminary evaluation into the performance of Autopilot during a recent fatal crash that occurred in a Model S. This is the first known fatality in just over 130 million miles where Autopilot was activated. Among all vehicles in the US, there is a fatality every 94 million miles. Worldwide, there is a fatality approximately every 60 million miles. It is important to emphasize that the NHTSA action is simply a preliminary evaluation to determine whether the system worked according to expectations.

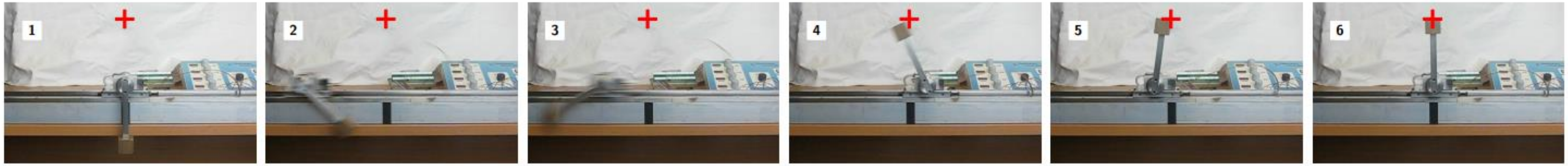
Following our standard practice, Tesla informed NHTSA about the incident immediately after it occurred. What we know is that the vehicle was on a divided highway with Autopilot engaged when a tractor trailer drove across the highway perpendicular to the Model S. Neither Autopilot nor the driver noticed the white side of the tractor trailer against a brightly lit sky, so the brake was not applied. The high ride height of the trailer combined with its positioning across the road and the extremely rare circumstances of the impact caused the Model S to pass under the trailer, with the bottom of the trailer

Image source: http://bdl101.ml/MLSS_2019_BDL_1.pdf

Why Bayesian (deep) learning?
Example: medical data

Image source: http://bdl101.ml/MLSS_2019_BDL_1.pdf

Why Bayesian (deep) learning? Example: Reinforcement learning



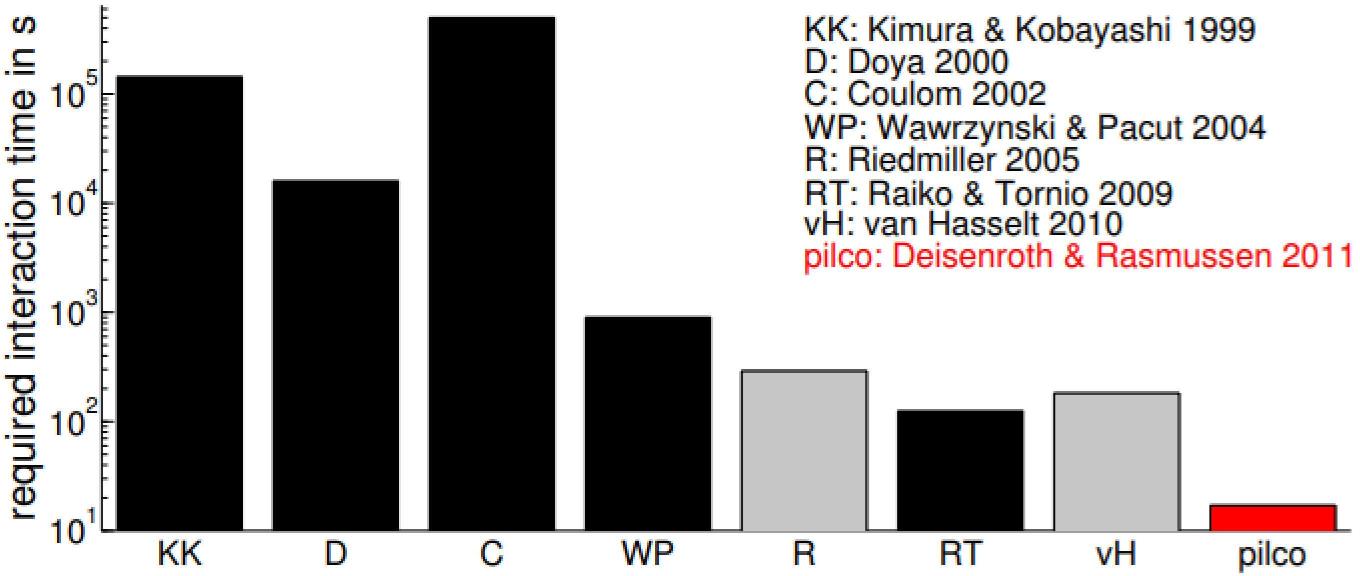


Image source: Deisenroth and Rasmussen, PILCO: a model-based and data-efficient approach to policy search, ICML, 2011

Why Bayesian (deep) learning?
Example: active learning

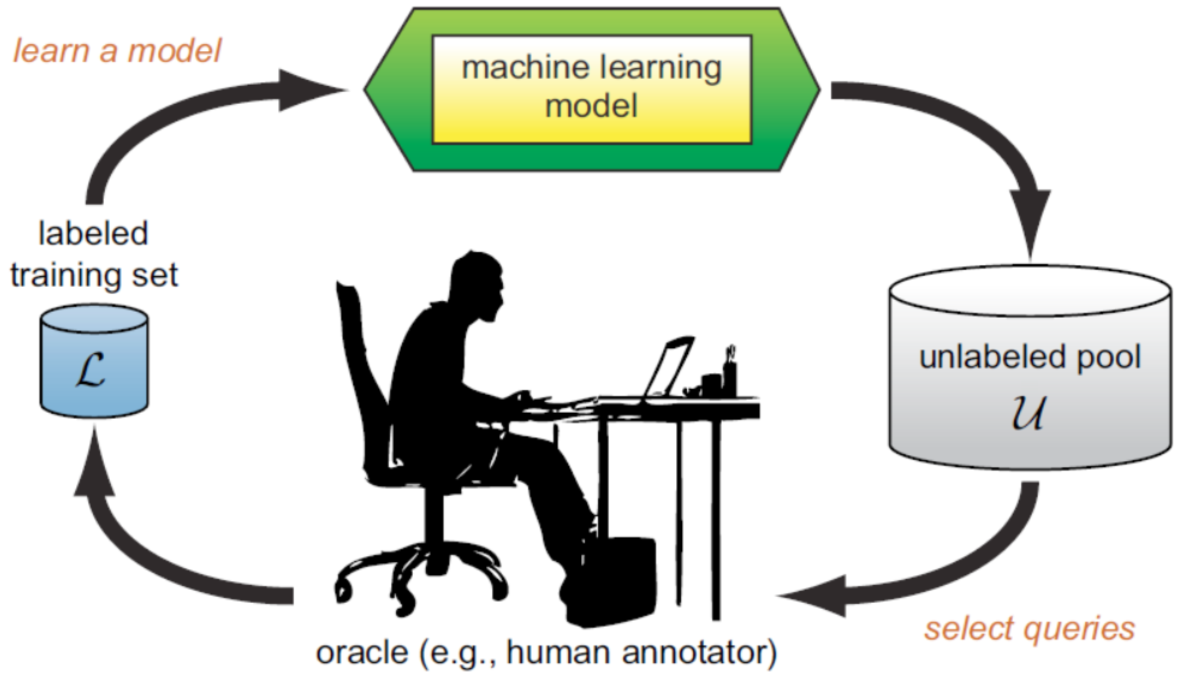


Figure 1: The pool-based active learning cycle.

Image source: http://seb.kr/w/Active_Learning

Bayes by backprop

Bayesian neural networks (BNN)

- Neural networks with priors on parameters

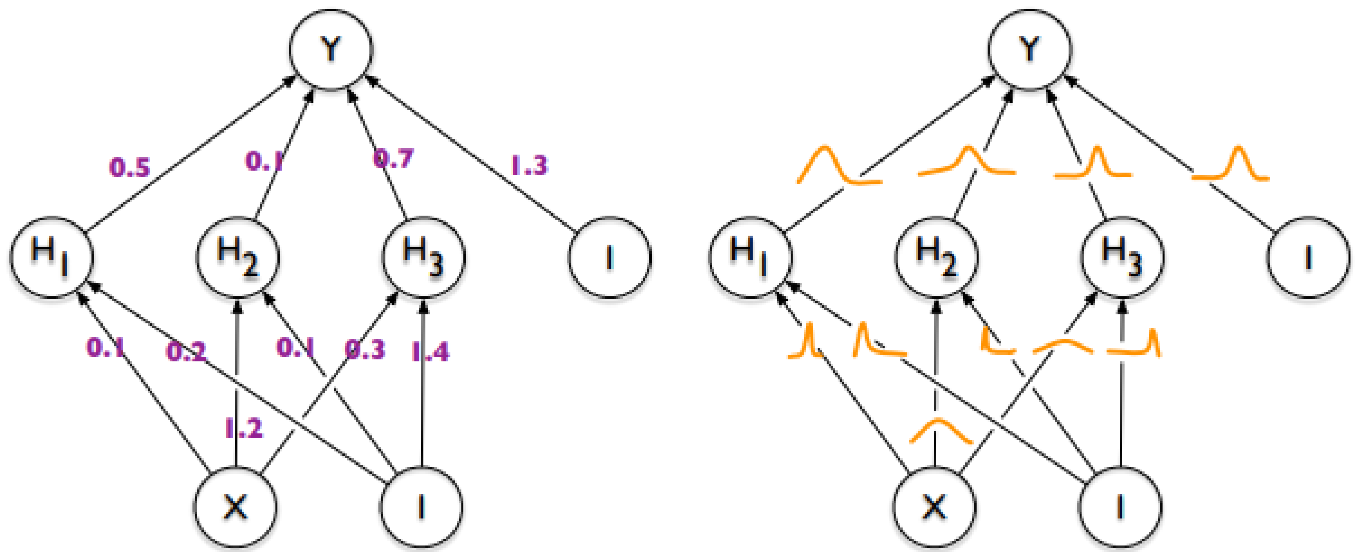


Image source: Blundell et al, “Weight uncertainty in neural network”, ICML, 2015

Bayesian neural networks (BNN)

- What makes it so hard?
  - Number of parameters
  - Number of data
- Posterior is not in a closed form, and even approximating it is not easy.

$$P(\underbrace{\boxed{\mathbf{W}}}_{100{,}000\sim1{,}000{,}000}|\mathbf{X},\mathbf{y}) = \frac{P(\mathbf{y}|\mathbf{X},\mathbf{W})P(\mathbf{W})}{P(\mathbf{y}|\underbrace{\boxed{\mathbf{X}}}_{10{,}000\sim1{,}000{,}000})}$$

Bayes by backprop (BBB)

- Blundell et al., “Weight uncertainty in neural network”, ICML, 2015
- Multivariate Gaussian prior on weights of neural networks
- Stochastic gradient variational inference for large-scale data

Bayes by backprop (BBB)

- A dataset, neural network, and its weight.

$$\mathcal{D} = (\mathbf{X}, \mathbf{y}) = (\mathbf{x}_i, y_i)_{i=1}^{n}.$$
$$\mathbf{f}(\mathbf{x}; \mathbf{W}) = \mathrm{NN}(\mathbf{x}; \mathbf{W}).$$

- Prior?

$$p(\mathbf{W}) = \mathcal{N}(\mathbf{W} | \mathbf{0}, \sigma^2 \mathbf{I}) = \prod_j \mathcal{N}(w_j | 0, \sigma^2).$$

Bayes by backprop (BBB)

- Likelihood?
  - Regression:

$$p(\mathbf{y}|\mathbf{X},\mathbf{W}) = \prod_{i=1}^{n} p(y_i|\mathbf{x}_i,\mathbf{W}) = \prod_{i=1}^{n} \mathcal{N}(y_i|\mathbf{f}(\mathbf{x}_i;\mathbf{W}),\rho^2)$$

$$\log p(\mathbf{y}|\mathbf{X},\mathbf{W}) = -\sum_{i=1}^{n} \frac{(y_i - \mathbf{f}(\mathbf{x}_i;\mathbf{W}))^2}{2\gamma^2} + \text{const}$$

  - Classification:

$$p(\mathbf{y}|\mathbf{X},\mathbf{W}) = \prod_{i=1}^{n} p(y_i|\mathbf{x}_i,\mathbf{W}) = \prod_{i=1}^{n} \text{Categorical}(y_i|\mathbf{f}(\mathbf{x}_i;\mathbf{W})).$$

$$\log p(\mathbf{y}|\mathbf{X},\mathbf{W}) = -\sum_{i=1}^{n} \text{CrossEntropy}(y_i,\mathbf{f}(\mathbf{x}_i;\mathbf{W})) + \text{const}$$

Bayes by backprop (BBB)

- Posterior?

$$P(\boxed{\mathbf{W}}|\mathbf{X},\mathbf{y}) = \frac{P(\mathbf{y}|\mathbf{X},\mathbf{W})P(\mathbf{W})}{P(\mathbf{y}|\boxed{\mathbf{X}})}$$

100,000~1,000,000

10,000~1,000,000

  - No closed form
  - High-dimensional, and possibly multimodal

Bayes by backprop (BBB)

- Introduce variational distribution which is easier to handle

$$q(\mathbf{W}) = \prod_j q(w_j | \mu_j, \rho_j^2), \qquad (\boldsymbol{\mu}, \boldsymbol{\rho}) = \boldsymbol{\phi}$$

- Make it as close as possible to the true posterior: minimize the KL-divergence.

$$\min_{\boldsymbol{\phi}} \mathrm{KL}[q(\mathbf{W}) \| p(\mathbf{W} | \mathbf{X}, \mathbf{y})]$$

KL-divergence?

- Kullback–Leibler divergence
- A divergence measure between two probability distributions
- Always positive, but not symmetric

$$\mathrm{KL}[q(x)\|p(x)] = \int q(x)\log\frac{q(x)}{p(x)}\mathrm{d}x.$$

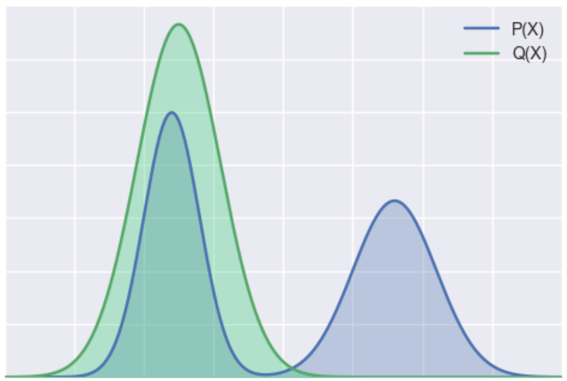
$$\mathrm{KL}[q(x)\|p(x)] \neq \mathrm{KL}[p(x)\|q(x)]$$

KL-divergence

- Implication of the order

$$\mathrm{KL}[q(x)\|p(x)] = \int q(x)\log\frac{q(x)}{p(x)}\mathrm{d}x.$$

$$\mathrm{KL}[q(x)\|p(x)] \neq \mathrm{KL}[p(x)\|q(x)]$$



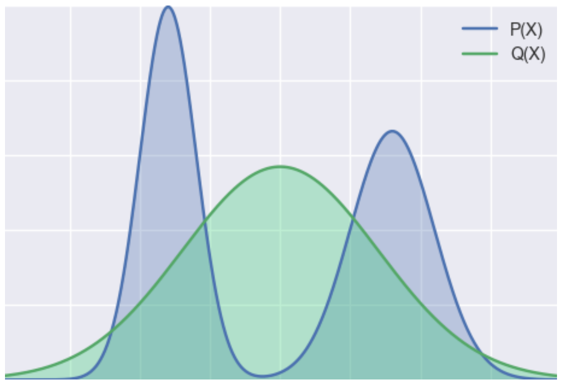


Image source: https://wiseodd.github.io/techblog/2016/12/21/forward-reverse-kl/

Evidence lower bound (ELBO)

$$\mathrm{KL}[q(\mathbf{W})\|p(\mathbf{W}|\mathbf{X},\mathbf{y})] = \int q(\mathbf{W})\left(\log q(\mathbf{W}) - \log \frac{p(\mathbf{y}|\mathbf{X},\mathbf{W})p(\mathbf{W})}{p(\mathbf{y}|\mathbf{X})}\right)\mathrm{d}\mathbf{W}$$

$$= \log p(\mathbf{y}|\mathbf{X}) - \left(\boxed{\mathbb{E}_{q(\mathbf{W})}[\log p(\mathbf{y}|\mathbf{X},\mathbf{W})] - \mathrm{KL}[q(\mathbf{W})\|p(\mathbf{W})]}\right).$$

Minimizing the KL-divergence is equivalent to maximizing the evidence lower bound.

$$\log p(\mathbf{y}|\mathbf{X}) \geq \underbrace{\boxed{\mathbb{E}_{q(\mathbf{W})}[\log p(\mathbf{y}|\mathbf{X},\mathbf{W})]}}_{\text{Expected likelihood}} - \underbrace{\boxed{\mathrm{KL}[q(\mathbf{W})\|p(\mathbf{W})]}}_{\text{regularization}}.$$

Computing & maximizing the ELBO

- The expected log-likelihood term and its gradient is not tractable

$$\mathbb{E}_{q(\mathbf{W})}[\log p(\mathbf{y}|\mathbf{X}, \mathbf{W})] = \int q(\mathbf{W}) \log p(\mathbf{y}|\mathbf{X}, \mathbf{W}) \mathrm{d}\mathbf{W}.$$

$$\begin{aligned}\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q(\mathbf{W})}[\log p(\mathbf{y}|\mathbf{X}, \mathbf{W})] = \int & (\nabla_{\boldsymbol{\phi}} q(\mathbf{W}) \log p(\mathbf{y}|\mathbf{X}, \mathbf{W}) \\ & + q(\mathbf{W}) \nabla_{\boldsymbol{\phi}} \log p(\mathbf{y}|\mathbf{X}, \mathbf{W})) \mathrm{d}\mathbf{W}.\end{aligned}$$

- The KL-divergence term is given in closed form (KL-divergence between two Gaussian distributions)

Reparametrization trick

- Kingma & Welling, "Auto-encoding variational Bayes", ICLR 2014
- By the elementary property of Gaussian distribution,

$$q(x) = \mathcal{N}(x|\mu, \rho^2)$$

$$\Downarrow$$

$$r(\varepsilon) = \mathcal{N}(0, 1), \quad x = \mu + \rho\varepsilon$$
$$\mathbb{E}_{q(x)}[f(x)] = \mathbb{E}_{r(\varepsilon)}[f(\mu + \rho\varepsilon)].$$

Reparameterization trick

- By the reparameterization trick,

$$\mathbb{E}_{q(\mathbf{W})}[\log p(\mathbf{y}|\mathbf{X},\mathbf{W})] = \int q(\mathbf{W}) \log p(\mathbf{y}|\mathbf{X},\mathbf{W}) \mathrm{d}\mathbf{W}.$$

$$= \int r(\boldsymbol{\varepsilon}) \log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\mu} + \boldsymbol{\rho}\boldsymbol{\varepsilon}) \mathrm{d}\boldsymbol{\varepsilon}$$

$$w_j = \mu_j + \rho_j \varepsilon_j, \quad \varepsilon_j \sim \mathcal{N}(0,1).$$

Computing the gradient

- With the reparameterization trick, we have

$$\nabla_{\boldsymbol{\phi}} \mathbb{E}_q[\log p(\mathbf{y}|\mathbf{X}, \mathbf{W})] = \int r(\boldsymbol{\varepsilon}) \nabla_{\boldsymbol{\phi}} \log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\mu} + \boldsymbol{\gamma}\boldsymbol{\varepsilon}) \mathrm{d}\boldsymbol{\varepsilon}.$$

- Monte-Carlo approximation:

$$\nabla_{\boldsymbol{\phi}} \mathbb{E}_q[\log p(\mathbf{y}|\mathbf{X}, \mathbf{W})] \approx \frac{1}{S} \sum_{s=1}^{S} \nabla_{\boldsymbol{\phi}} \log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\mu} + \boldsymbol{\gamma}\boldsymbol{\varepsilon}^{(s)}), \quad \boldsymbol{\varepsilon}^{(1)}, \ldots, \boldsymbol{\varepsilon}^{(S)} \overset{\text{i.i.d.}}{\sim} r(\boldsymbol{\varepsilon}).$$

- In practice, even a single sample is enough!

$$\nabla_{\boldsymbol{\phi}} \mathbb{E}_q[\log p(\mathbf{y}|\mathbf{X}, \mathbf{W})] \approx \frac{1}{S} \nabla_{\boldsymbol{\phi}} \log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\mu} + \boldsymbol{\gamma}\boldsymbol{\varepsilon}),$$

Training Neural networks with BBB

- Training objective

$$\frac{1}{n}(\mathbb{E}_q[\log p(\mathbf{y}|\mathbf{X},\mathbf{W})] - \mathrm{KL}[q(\mathbf{W})\|p(\mathbf{W})])$$
$$= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_q[\log p(y_i|\mathbf{x}_i,\mathbf{W})] - \frac{1}{n}\mathrm{KL}[q(\mathbf{W})\|p(\mathbf{W})]$$

- Stochastic gradient descent using mini-batches

$$\frac{1}{|B|}\sum_{i\in B}\nabla_{\boldsymbol{\phi}}\mathbb{E}_q[\log p(y_i|\mathbf{x}_i,\mathbf{W})] - \frac{1}{n}\nabla_{\boldsymbol{\phi}}\mathrm{KL}[q(\mathbf{W})\|p(\mathbf{W})]$$

- Reparameterization trick

$$\approx \frac{1}{|B|}\sum_{i\in B}\nabla_{\boldsymbol{\phi}}\log p(y_i|\mathbf{x}_i,\boldsymbol{\mu}+\boldsymbol{\rho}\boldsymbol{\varepsilon}) - \frac{1}{n}\nabla_{\boldsymbol{\phi}}\mathrm{KL}[q(\mathbf{W})\|p(\mathbf{W})].$$

Training Neural networks with BBB

Ordinary neural net training

- Sample a mini-batch.
- Compute the loss

$$-\frac{1}{|B|}\sum_{i\in B}\log p(y_i|\mathbf{x}_i,\mathbf{W})$$

- Compute the gradient w.r.t. $\mathbf{W}$.

$$-\frac{1}{|B|}\sum_{i\in B}\nabla_{\mathbf{W}}\log p(y_i|\mathbf{x}_i,\mathbf{W})$$

- Update the weights.

Training with BBB

- Sample a mini-batch.
- Sample noise $\boldsymbol{\varepsilon}$.
- Compute the (approximate) loss,

$$-\frac{1}{|B|}\sum_{i\in B}\log p(y_i|\mathbf{x}_i,\boldsymbol{\mu}+\boldsymbol{\rho}\boldsymbol{\varepsilon})+\frac{1}{n}\mathrm{KL}[q(\mathbf{W})\|p(\mathbf{W})]$$

- Compute the gradient w.r.t. $\boldsymbol{\phi}=(\boldsymbol{\mu},\boldsymbol{\rho})$.

$$-\frac{1}{|B|}\sum_{i\in B}\nabla_{\boldsymbol{\phi}}\log p(y_i|\mathbf{x}_i,\boldsymbol{\mu}+\boldsymbol{\rho}\boldsymbol{\varepsilon})+\frac{1}{n}\nabla_{\boldsymbol{\phi}}\mathrm{KL}[q(\mathbf{W})\|p(\mathbf{W})]$$

- Update the weights.

Prediction with BBB

- For a new observation to test,

$$
\begin{aligned}
p(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) &= \int p(y_*|\mathbf{x}_*, \mathbf{W}) p(\mathbf{W}|\mathbf{X}, \mathbf{y}) \mathrm{d}\mathbf{W} \\
&\approx \int p(y_*|\mathbf{x}_*, \mathbf{W}) \boxed{q(\mathbf{W})} \mathrm{d}\mathbf{W} \\
&\approx \frac{1}{S} \sum_{s=1}^{S} p(y_*|\mathbf{x}_*, \boldsymbol{\mu} + \boldsymbol{\rho}\boldsymbol{\varepsilon}^{(s)}), \quad \boldsymbol{\varepsilon}^{(1)}, \ldots, \boldsymbol{\varepsilon}^{(S)} \overset{\text{i.i.d.}}{\sim} r(\boldsymbol{\varepsilon}).
\end{aligned}
$$

Standard neural net vs BBB

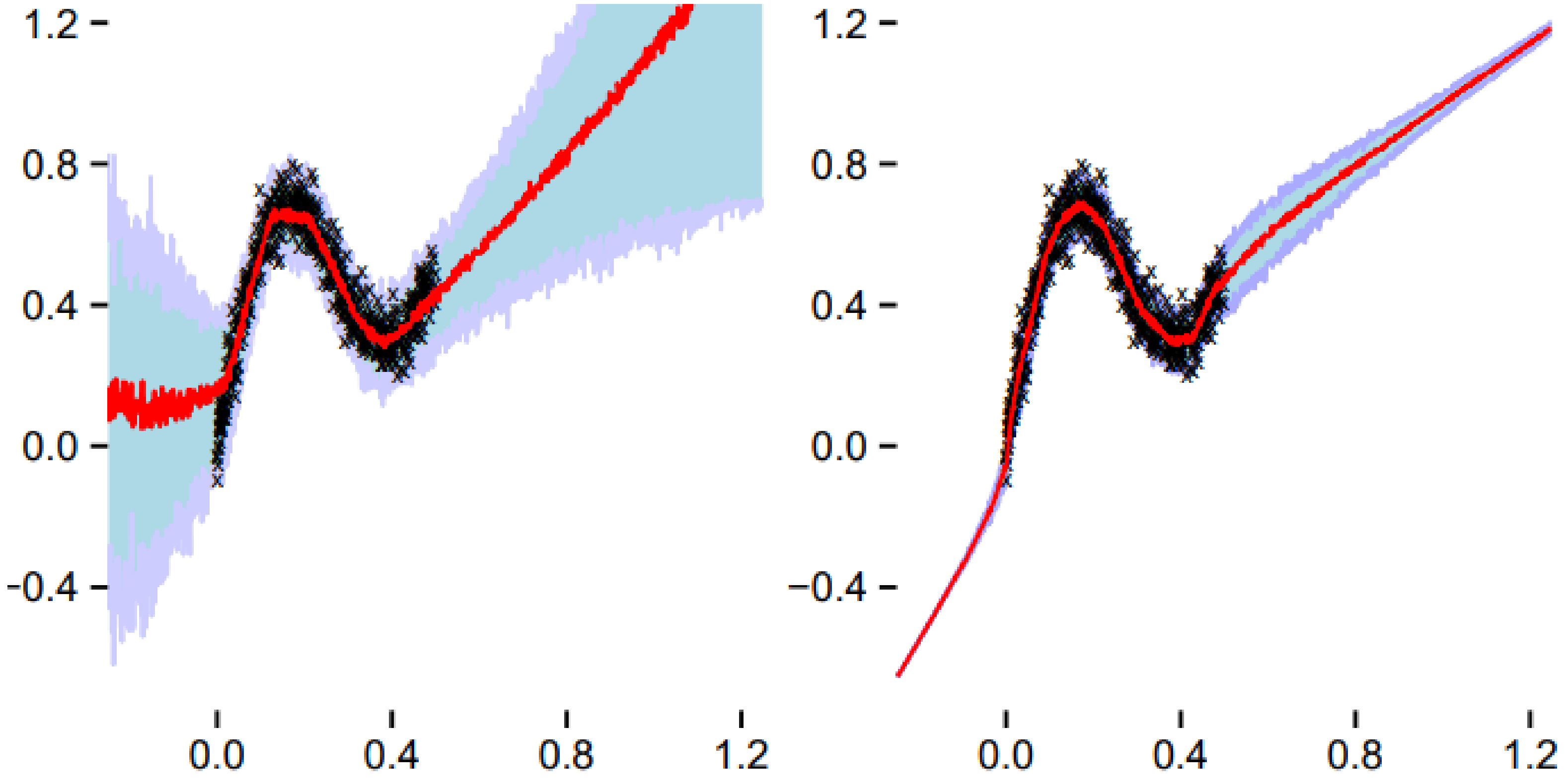


Image source: https://wiseodd.github.io/techblog/2016/12/21/forward-reverse-kl/

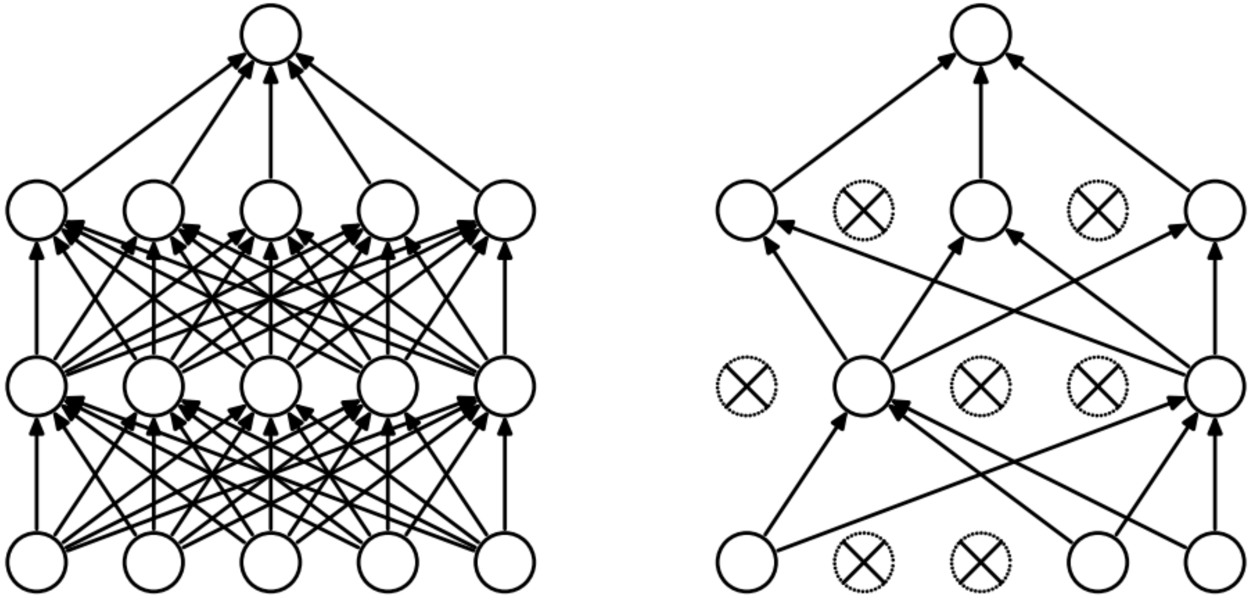
BBB in summary

- Optimize the parameters of posterior distribution of weights.
- Posterior as Gaussian, twice many parameters.
- Approximation by stochastic gradient descent + reparameterization trick.
- Cons: too simple approximate posterior distribution (independent posteriors), so underestimating posterior covariances.

Monte-Carlo Dropout

Dropout

- An easy way to regularize neural networks

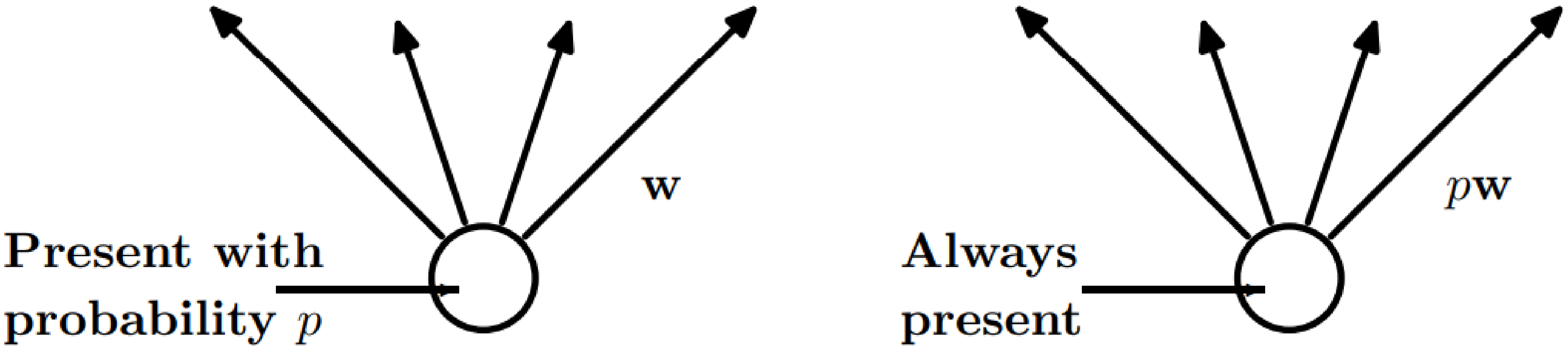


(a) Standard Neural Net (b) After applying dropout.

Image source: Srivastava et al, Dropout: A Simple Way to Prevent Neural Networks from Overfitting, JMLR, 2014

Dropout

- During training, drop a neuron with a fixed probability p
- Equivalent to training multiple models at the same time.
- Inference is done with rescaled weights.



(a) At training time (b) At test time

Image source: Srivastava et al, Dropout: A Simple Way to Prevent Neural Networks from Overfitting, JMLR, 2014

Dropout as a Bayesian approximation

- Assume a Gaussian prior for the neural network weights as before.

$$p(\mathbf{W}) = \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

- We want to do the variational inference as before, but use a special form of variational distribution.

$$\mathbf{z} \sim \prod_j \text{Bernoulli}(p_j) \quad \mathbf{W} = \mathbf{M} \odot \mathbf{z}.$$

$$q(\mathbf{W}) = \prod_j \text{Bernoulli}(z_j | p_j).$$

Dropout as a Bayesian approximation

- With the variational distribution defined as above, the training objective reduces to

$$-\frac{1}{n}\sum_{i=1}^{n}\log p(y_i|\mathbf{x}_i,\mathbf{M}\odot\mathbf{z})+\lambda\|\mathbf{W}\|^2.$$

- In other words, the neural network training with dropout is a Bayesian approximation using a variational distribution having specific form.

Prediction in dropout

- In principle, a prediction should be computed as

$$\begin{aligned} p(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) &= \int p(y_*|\mathbf{x}_*, \mathbf{W}) p(\mathbf{W}|\mathbf{X}, \mathbf{y}) \mathrm{d}\mathbf{W} \\ &\approx \int p(y_*|\mathbf{x}_*, \mathbf{W}) q(\mathbf{W}) \mathrm{d}\mathbf{W} \end{aligned}$$

- But what dropout is doing is a wrong approximation of the true predictive distribution.

$$p(y_*|\mathbf{x}_*, p \odot \mathbf{M}) = p(y_*|\mathbf{x}_*, \mathbb{E}_q[\mathbf{W}]).$$

Monte-Carlo (MC) dropout

- We should properly approximate the expectation

$$
\begin{aligned}
p(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) &= \int p(y_*|\mathbf{x}_*, \mathbf{W}) p(\mathbf{W}|\mathbf{X}, \mathbf{y}) \mathrm{d}\mathbf{W} \\
&\approx \int p(y_*|\mathbf{x}_*, \mathbf{W}) q(\mathbf{W}) \mathrm{d}\mathbf{W} \\
&\approx \frac{1}{S} \sum_{s=1}^{S} p(y_*|\mathbf{x}_*, \mathbf{z}^{(s)} \odot \mathbf{M}), \quad \mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(S)} \overset{\text{i.i.d.}}{\sim} \prod_j \mathrm{Bernoulli}(p_j).
\end{aligned}
$$

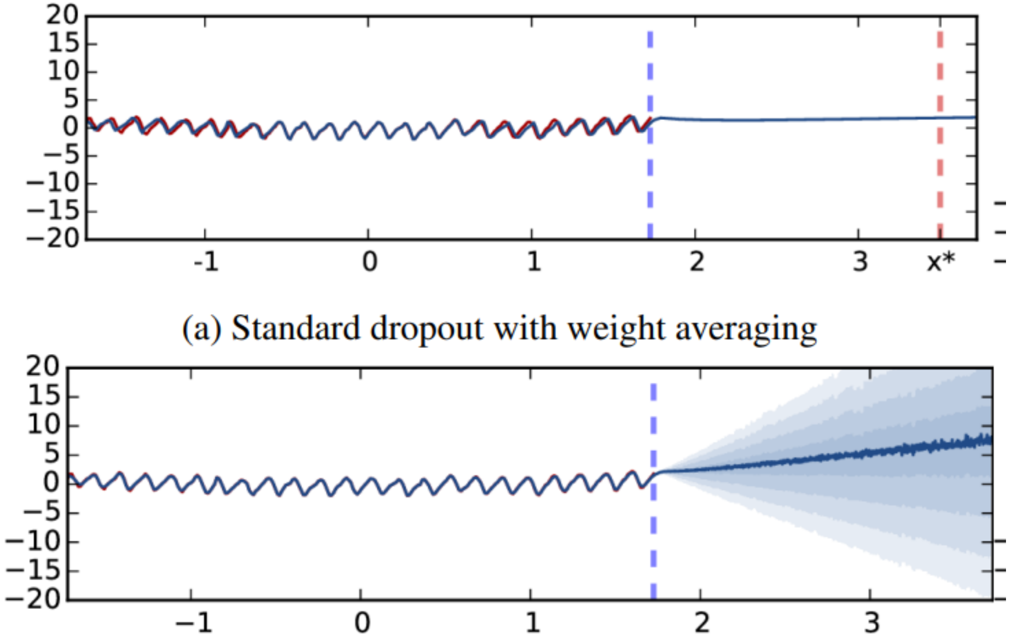
- In other words, do multiple forward pass with dropout and average them.

MC dropout - summary

- Nothing new but new interpretation of an existing approach
- Dropout is an approximate inference scheme for Bayesian neural network
- The inference should be done accordingly – instead of rescaling weights, do multiple forward passes with dropout as in training and average them.

MC dropout – some results

- Standard dropout vs MC dropout



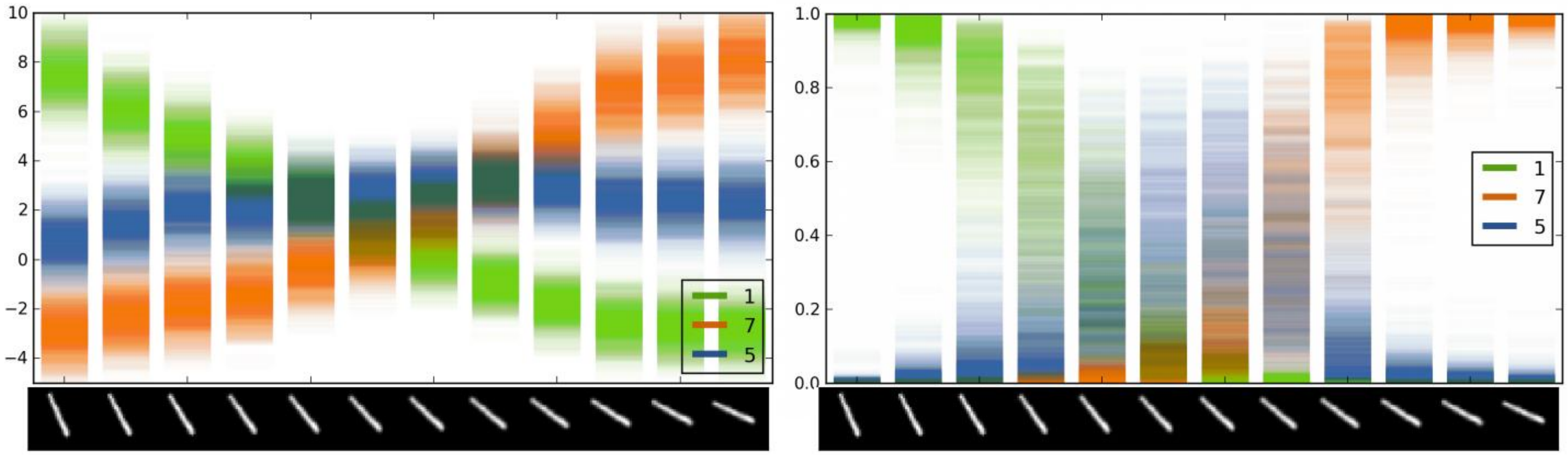
(a) Standard dropout with weight averaging

(c) MC dropout with ReLU non-linearities

Image source: Gal and Ghahramani 2016, Dropout as a Bayesian approximation: representing model uncertainty in deep learning, ICML 2016

MC dropout – some results

- Uncertainties in MC dropout networks



(a) Softmax *input* scatter

(b) Softmax *output* scatter

Image source: Gal and Ghahramani 2016, Dropout as a Bayesian approximation: representing model uncertainty in deep learning, ICML 2016

Deep Ensembles

Deep ensembles

- Lakshminarayanan et al., Simple and scalable predictive uncertainty estimation using deep ensembles, NIPS 2017.
- Idea: train same network multiple times with different random initializations. Then just average the output of those multiple results,
- Surprisingly, this simple method is still the state-of-the-art for many tasks.

Deep ensembles

- Is deep ensemble a Bayesian method?
- Not exactly, but it is a form of bagging, a data-driven way of defining Bayesian posterior.
- Bagging (Bootstrap aggregating): train the same model multiple times with different random initializations and different training data generated from a same pool, and average the prediction results.

Deep ensembles

- In practice, we usually train 5~10 models.
- This means that we need 5~10 times space to store parameters.
- Also 5~10 times training time (if not done in parallel).

Why deep ensemble is so powerful?

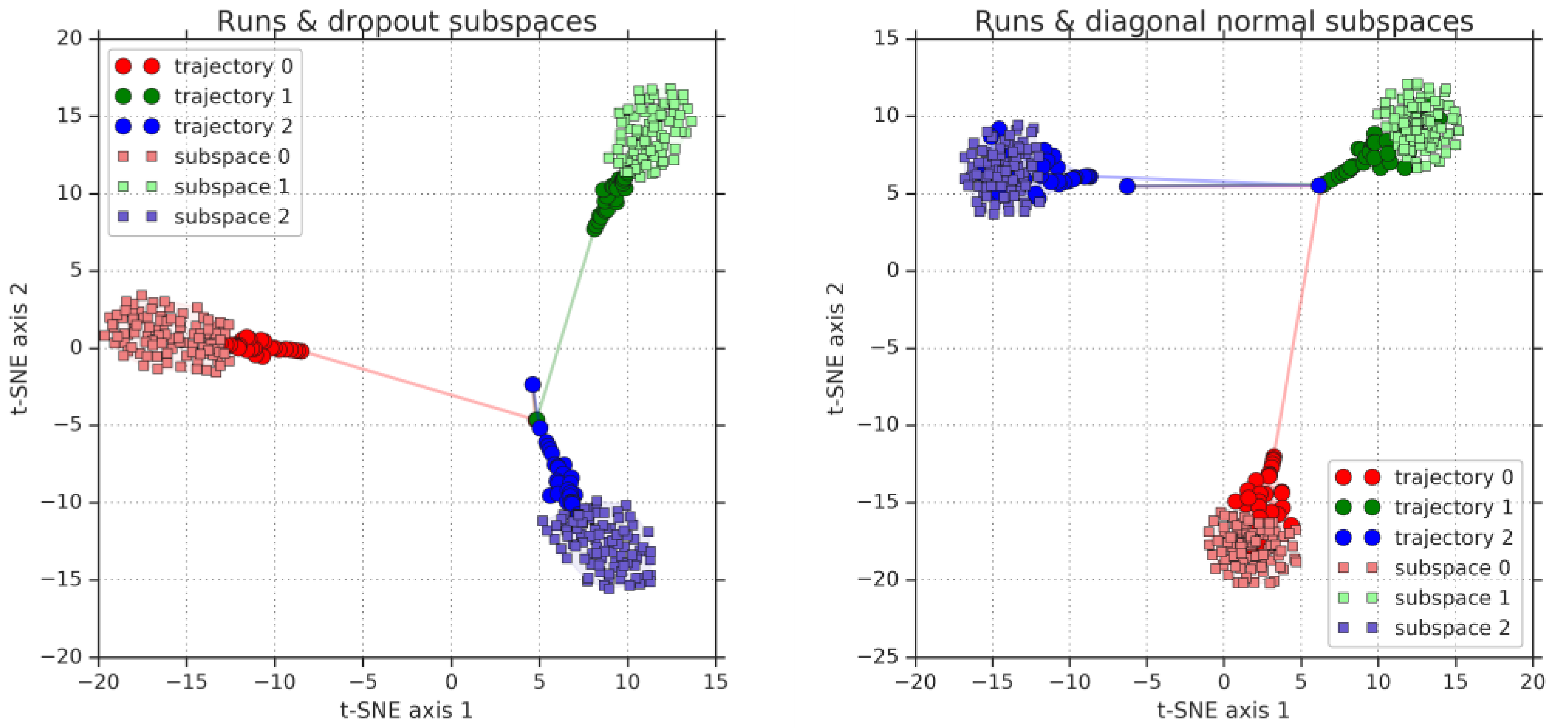


Image source: Fort et al., Deep ensembles: a loss landscape perspective, arXiv 2019.

Bayesian deep learning in practice

Comparing the algorithms

- BBB
  - Good predictive performance for relatively small networks
  - Fails for large-scale data or large networks
  - Underestimating variances
- MC-dropout
  - Simple and easy to implement
  - Too simple approximation of posterior, so performance is often bad
- Deep ensemble
  - Good predictive performance
  - Requires much more training time & parameters.

Further readings

- Practical Bayesian deep learning for ImageNet scale networks

https://papers.nips.cc/paper/8681-practical-deep-learning-with-bayesian-principles.pdf

- A simple Bayesian approximation via weight averaging

https://arxiv.org/pdf/1902.02476.pdf

- Complex variational distribution for better capturing uncertainty

https://arxiv.org/pdf/1703.01961.pdf

Coding practice