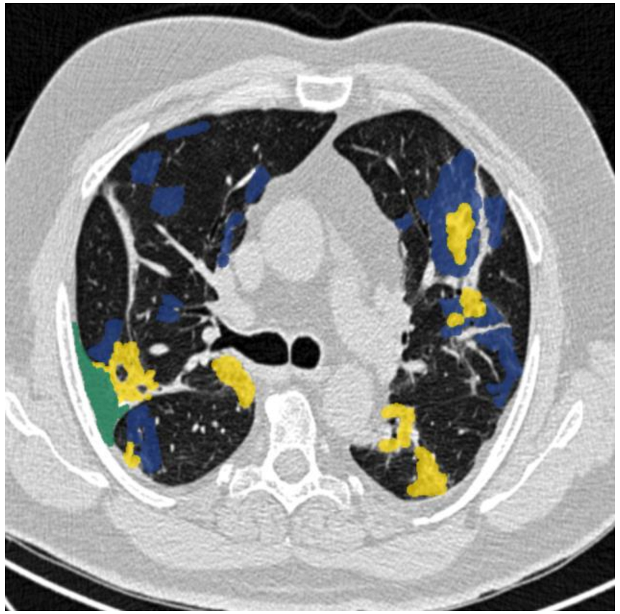
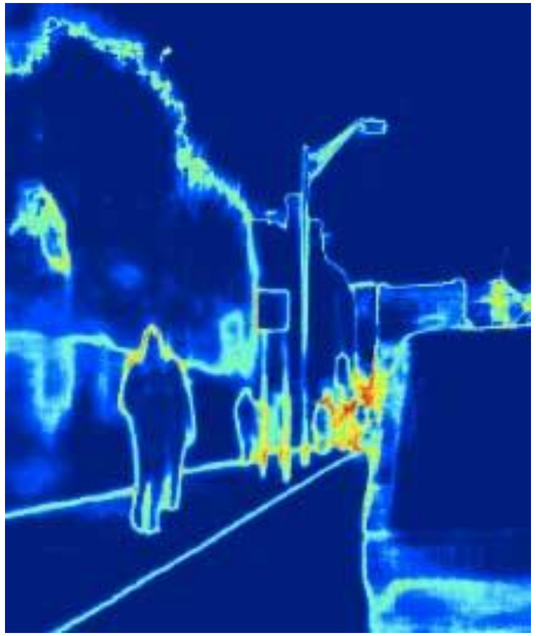
Bayesian Neural Networks

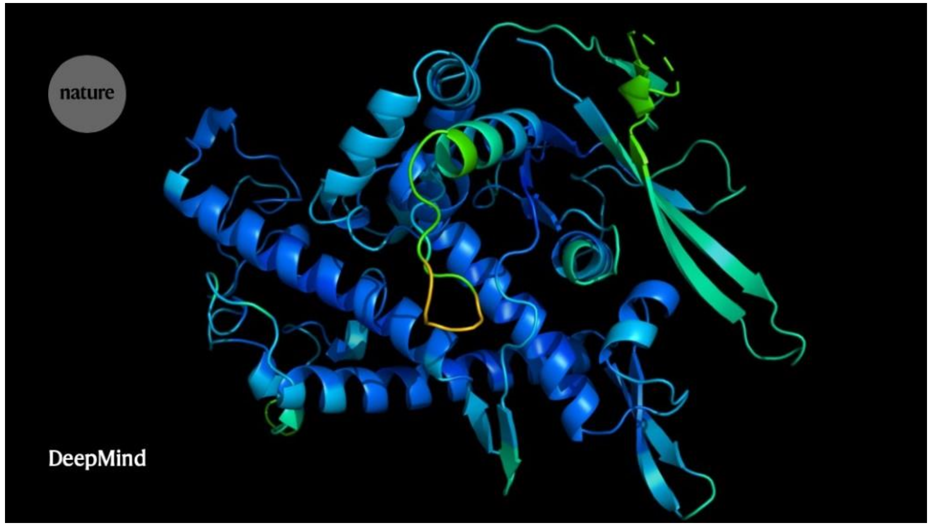
Hrushikesh Loya*

*Contact: hrushikesh.loya@balliol.ox.ac.uk

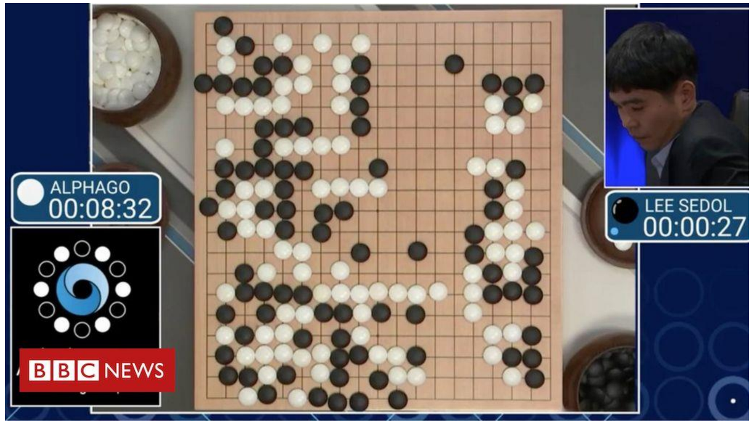
NNs in real-life





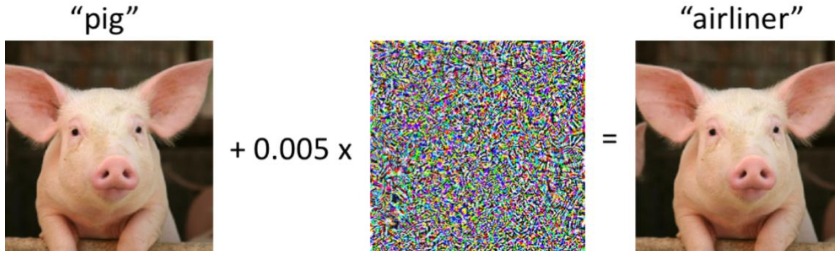


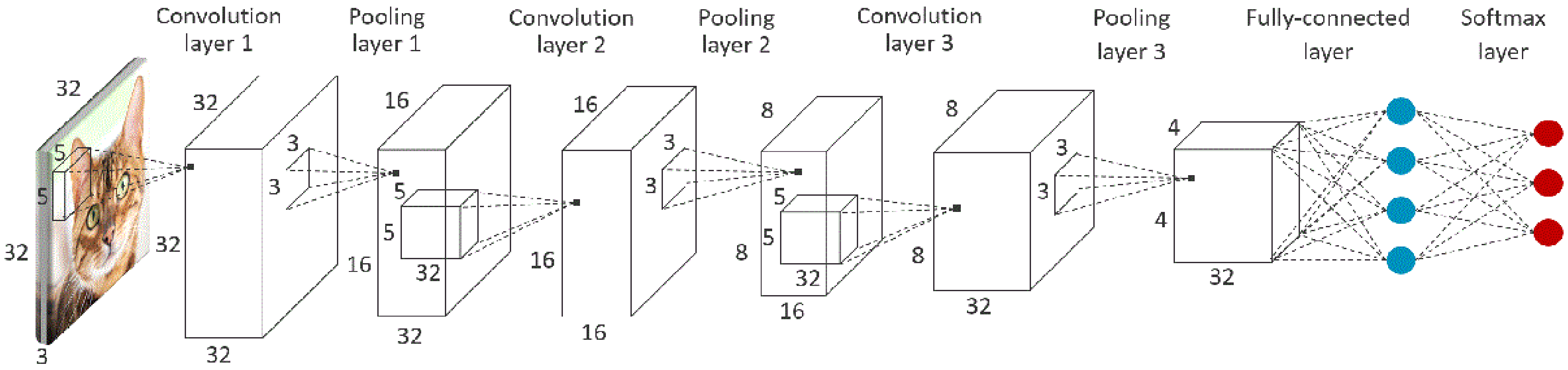
Automatically find names of people, places, products, and organizations in text across many languages.



Problems with Vanilla NNs

- Prone to over-fitting
- Incapable of assessing uncertainty in the data
- A vanilla NN can be easily fooled (AI Safety)
- Relies on big-data heavily





Neural Network model definition

Bayesian Inference

- Input data: $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$
- Output: $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$

- Let's assume a prior on the NN weights: $p(\boldsymbol{\omega})$

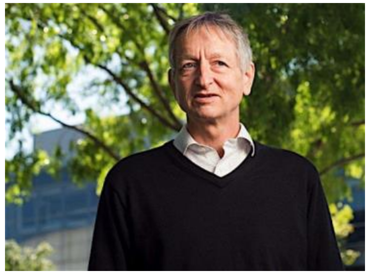
- And pose this as a posterior estimation problem:

$$p(\mathbf{y}|\mathbf{x}, \boldsymbol{\omega}) \Longrightarrow p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y}) = \frac{p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega})p(\boldsymbol{\omega})}{p(\mathbf{Y}|\mathbf{X})}.$$

Early Work

- Denker and LeCun, 1991: Propose Laplace method for posterior estimation in NNs
  - Identify the mode using MLE on NN weights
  - Fit a Gaussian to the discovered mode, with the width of the Gaussian determined by the Hessian at that mode

- Hinton and Vancamp, 1993: Propose MDL (minimum description length) as a regularization for NN weights.
  - Analytical loss for 1-hidden layer NN
  - Similar loss to the VI objective later



Early Work

- Neal, 1995: Infinite-width NNs
  - A single hidden layer NN => stable stochastic processes
  - Gaussian Prior => Gaussian Process
- Neal, 1995: Hamiltonian Monte Carlo
  - Sampling based approach
  - Does not rely on any assumptions about the form of the posterior
  - Considered a **gold-standard** for low-dimension problems

Modern Approximate Inference

- Minimize the KL divergence between approximating posterior and the true posterior:

$$\mathrm{KL}\Big(q_\theta(\boldsymbol{\omega})||p(\boldsymbol{\omega}|\mathbf{X},\mathbf{Y})\Big) \propto -\int q_\theta(\boldsymbol{\omega})\log p(\mathbf{Y}|\mathbf{X},\boldsymbol{\omega})\mathrm{d}\boldsymbol{\omega} + \mathrm{KL}(q_\theta(\boldsymbol{\omega})||p(\boldsymbol{\omega}))$$
$$= -\sum_{i=1}^{N}\int q_\theta(\boldsymbol{\omega})\log p(\mathbf{y}_i|\mathbf{f}^{\boldsymbol{\omega}}(\mathbf{x}_i))\mathrm{d}\boldsymbol{\omega} + \mathrm{KL}(q_\theta(\boldsymbol{\omega})||p(\boldsymbol{\omega}))$$

- Hinton and Vancamp, 1993 used a **fully-factorized Gaussian** posterior and prior:

$$q_\theta(\boldsymbol{\omega}) = \prod_{i=1}^{L} q_\theta(\mathbf{W}_i) = \prod_{i=1}^{L}\prod_{j=1}^{K_i}\prod_{k=1}^{K_{i+1}} q_{m_{ijk},\sigma_{ijk}}(w_{ijk}) = \prod_{i,j,k}\mathcal{N}(w_{ijk}; m_{ijk}, \sigma^2_{ijk}).$$

- But, expected log likelihood is intractable for most BNN model structures
- Came up with analytical closed-form solutions for 1 hidden layer BNN

Modern Approximate Inference

- Until 2011, BNNs were:
  - Computationally Intensive
  - Didn't scale to Big-data
  - Only limited to shallow NNs
- Graves, 2011: Used data sub-sampling techniques in a fully factorized VI objective
  - True gradients replaced by noisy estimates from the mini-batch
  - Monte-Carlo sampling for approximating expected likelihood
  - **Adv:** Scalable to big-data and any NN structure
  - **Disadv:** Suffers from the high variance of the gradients (not easy to converge)

Weight Uncertainty in NN

- "Backpropagation-compatible algorithm for learning a probability distribution on the weights of a neural network"
- Minimizes the variational free energy or maximizes the ELBO:

$$\begin{aligned}\theta^\star &= \arg\min_\theta \mathrm{KL}[q(\mathbf{w}|\theta)||P(\mathbf{w}|\mathcal{D})] \\ &= \arg\min_\theta \int q(\mathbf{w}|\theta)\log\frac{q(\mathbf{w}|\theta)}{P(\mathbf{w})P(\mathcal{D}|\mathbf{w})}\mathrm{d}\mathbf{w} \\ &= \arg\min_\theta \mathrm{KL}\left[q(\mathbf{w}|\theta)\;||\;P(\mathbf{w})\right] - \mathbb{E}_{q(\mathbf{w}|\theta)}\left[\log P(\mathcal{D}|\mathbf{w})\right].\end{aligned}$$

- ELBO consists of (1) log-likelihood and (2) Regularization term

$$\mathcal{F}(\mathcal{D},\theta) = \mathrm{KL}\left[q(\mathbf{w}|\theta)\;||\;P(\mathbf{w})\right] - \mathbb{E}_{q(\mathbf{w}|\theta)}\left[\log P(\mathcal{D}|\mathbf{w})\right].$$

Unbiased Monte-Carlo gradients:

Remember Robbins-Monro conditions for SGD

- Let $\mathbf{w} = t(\theta, \epsilon)$, where:
  - t is a deterministic transform
  - θ is a parameter
  - ϵ is an easy to sample RV

For example:

$$t(\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\epsilon}) = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$$

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

- Then:

$$\frac{\partial}{\partial\theta}\mathbb{E}_{q(\mathbf{w}|\theta)}[f(\mathbf{w},\theta)] = \mathbb{E}_{q(\epsilon)}\left[\frac{\partial f(\mathbf{w},\theta)}{\partial \mathbf{w}}\frac{\partial \mathbf{w}}{\partial\theta} + \frac{\partial f(\mathbf{w},\theta)}{\partial\theta}\right]$$

Noisy **unbiased** gradients

- Applying the above result to approximate ELBO:

$$\mathcal{F}(\mathcal{D},\theta) \approx \sum_{i=1}^{n} \log q(\mathbf{w}^{(i)}|\theta) - \log P(\mathbf{w}^{(i)}) - \log P(\mathcal{D}|\mathbf{w}^{(i)}) \qquad \mathbf{w}^{(i)} \sim q(\mathbf{w}^{(i)}|\theta)$$

Algorithm

log(1+exp) transform for positivity

1. Sample $\epsilon \sim \mathcal{N}(0, I)$.
2. Let $\mathbf{w} = \mu + \log(1 + \exp(\rho)) \circ \epsilon$.
3. Let $\theta = (\mu, \rho)$.
4. Let $f(\mathbf{w}, \theta) = \log q(\mathbf{w}|\theta) - \log P(\mathbf{w})P(\mathcal{D}|\mathbf{w})$.
5. Calculate the gradient with respect to the mean

$$\Delta_\mu = \frac{\partial f(\mathbf{w}, \theta)}{\partial \mathbf{w}} + \frac{\partial f(\mathbf{w}, \theta)}{\partial \mu}. \qquad (3)$$

6. Calculate the gradient with respect to the standard deviation parameter ρ

$$\Delta_\rho = \frac{\partial f(\mathbf{w}, \theta)}{\partial \mathbf{w}} \frac{\epsilon}{1 + \exp(-\rho)} + \frac{\partial f(\mathbf{w}, \theta)}{\partial \rho}. \qquad (4)$$

7. Update the variational parameters:

$$\mu \leftarrow \mu - \alpha \Delta_\mu \qquad (5)$$

$$\rho \leftarrow \rho - \alpha \Delta_\rho. \qquad (6)$$

Derivatives calculated through back-prop

Weight Pruning

1. Calculate Signal to Noise for all weights (mean/sigma)
2. Remove weights with low SNR (hard thresholding)

Table 2. Classification Errors after Weight pruning

Proportion removed	# Weights	Test Error
0%	2.4m	1.24%
50%	1.2m	1.24%
75%	600k	1.24%
95%	120k	1.29%
98%	48k	1.39%

Regression

1. After training the NN with 1000 inputs in [0, 0.5]
2. Perform multiple forward passes to assess mean and uncertainty in predictions from test set in [-0.2, 1.2]

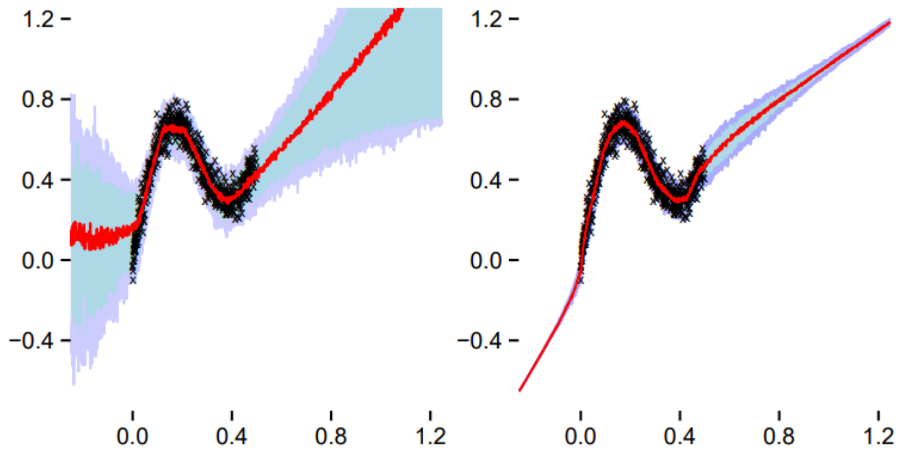
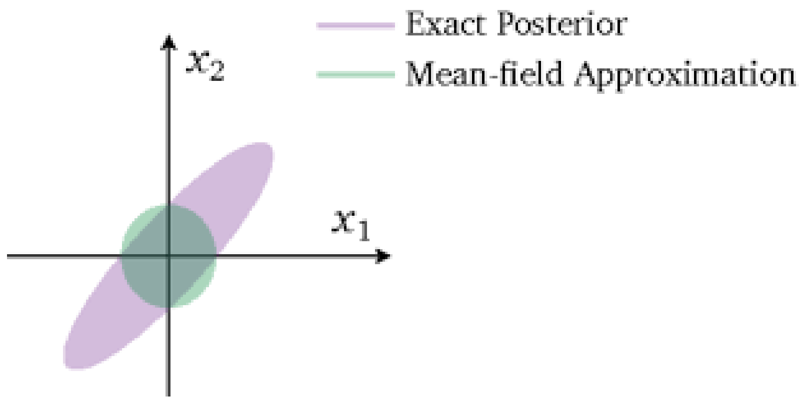
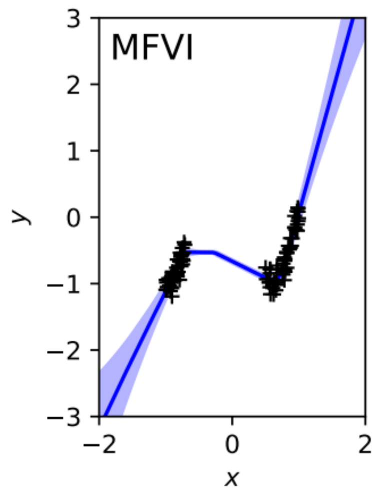


Fig 1: **Model** uncertainty increases as data is OOD

Problems with BBB

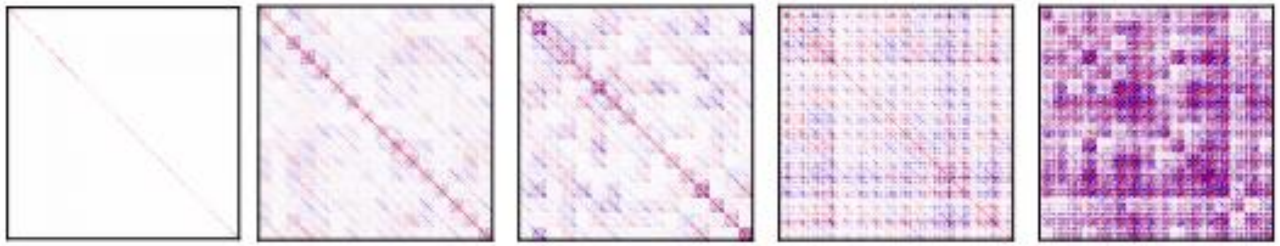
- Mean-field assumption limits the flexibility of approximate posterior
- Requires twice the number of parameters – memory cost
- Uninformative prior don't serve meaningful information to NN training
- Training anamolies: "In-between" Uncertainty (Foong 2019)
- Hyperparameter sensitivity and robust initilization schemes unknown





Going Ahead

- Until recently, mean-field approximation was considered limiting
- Farquhar, 2020 showed deep mean-field NNs can capture all weight correlations if more than 2 hidden layers (with RELU/ELU kind of activations)
- Vladimirova, 2019 showed deep mean-field Gaussian NNs produce sub-Weibull posterior tails – hence promoting sparsity
- Matthews, 2018 shows under Infinte-width mean-field Gaussian BNN behave like GPs (NNGP)



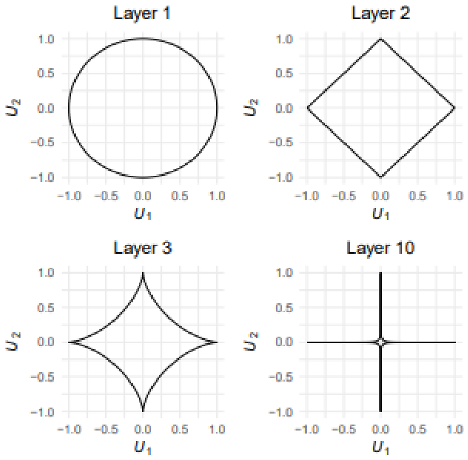
(a) One weight matrix.

(b) 5-layer product matrix. (Linear)

(c) 10-layer product matrix. (Linear)

(d) 5-layer local product matrix. (Leaky ReLU)

(e) 10-layer local product matrix. (Leaky ReLU)



Dropout as a Bayesian Approximation

- Gal and Ghahramani, 2015 show training with Bernoulli dropout is equivalent to training a deep Gaussian process
- Kingma and Welling, 2015 show Gaussian dropouts imposes a Gaussian posterior on model weights.

$$B = (A \odot \Xi)W, \text{ with } \xi_{mi} \sim p(\xi) \qquad \xi_{mi} \sim \mathcal{N}(1, \alpha = \tfrac{p}{1-p})$$

$$w_{ij} = \theta_{ij}\xi_{ij} = \theta_{ij}(1 + \sqrt{\alpha}\epsilon_{ij}) \sim \mathcal{N}(w_{ij} \,|\, \theta_{ij}, \alpha\theta_{ij}^2)$$
$$\epsilon_{ij} \sim \mathcal{N}(0, 1)$$

- **Training**: Similar to dropout training
- **Testing**: Perform multiple forward passes with random dropouts

Concrete Dropouts

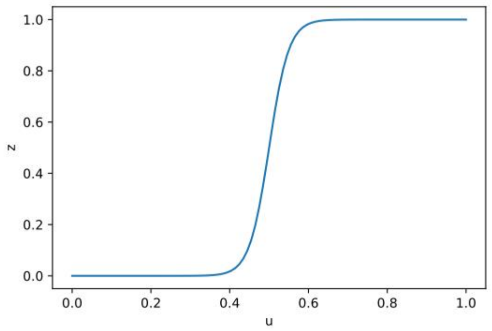
- The ELBO estimated using MC samples: $\widehat{\mathcal{L}}_{\text{MC}}(\theta) = -\frac{1}{M}\sum_{i \in S} \log p(\mathbf{y}_i | \mathbf{f}^{\boldsymbol{\omega}}(\mathbf{x}_i)) + \frac{1}{N}\text{KL}(q_\theta(\boldsymbol{\omega}) || p(\boldsymbol{\omega}))$
- Assuming the approximate posterior factorizes over layers: $q_\theta(\boldsymbol{\omega}) = \prod_l q_{\mathbf{M}_l}(\mathbf{W}_l)$
- Assuming a tunable-dropout parameter: $q_{\mathbf{M}_l}(\mathbf{W}_l) = \mathbf{M}_l \cdot \text{diag}[\text{Bernoulli}(1-p_l)^{K_l}]$
- The KL divergence can be calculated in closed form:

$$\text{KL}(q_\theta(\boldsymbol{\omega}) || p(\boldsymbol{\omega})) = \sum_{l=1}^{L} \text{KL}(q_{\mathbf{M}_l}(\mathbf{W}_l) || p(\mathbf{W}_l))$$

$$\text{KL}(q_{\mathbf{M}}(\mathbf{W}) || p(\mathbf{W})) \propto \frac{l^2(1-p)}{2} ||\mathbf{M}||^2 - K\mathcal{H}(p)$$

- Concrete/Gumbel-Softmax trick: to sample from Bernoulli

$$\tilde{\mathbf{z}} = \text{sigmoid}\left(\frac{1}{t} \cdot \left(\log p - \log(1-p) + \log u - \log(1-u)\right)\right)$$



(a) Relation between $\mathbf{z} \sim \text{Concrete}(p)$ and $u \sim \text{Uniform}[0, 1]$, given by a sigmoid function.

Stochastic Gradient Langevin Dynamics (SGLD)

- Welling and Teh, 2011: Show adding the right amount of noise to a standard SGD will converge to the true posterior distribution

$$\Delta\theta_t = \frac{\epsilon_t}{2}\left(\nabla \log p(\theta_t) + \frac{N}{n}\sum_{i=1}^{n}\nabla \log p(x_{ti}|\theta_t)\right) + \eta_t$$

$$\eta_t \sim N(0, \epsilon_t)$$

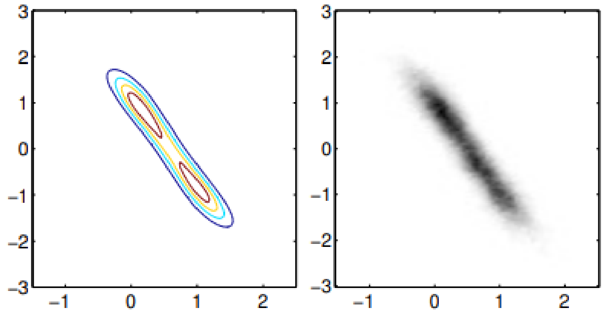


Figure 1. True and estimated posterior distribution.

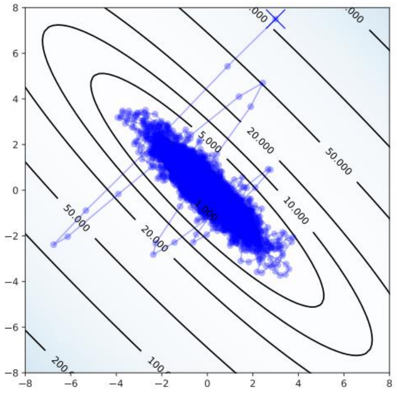
- Disadvantage: Often leads to mode collapse, and only explores one-mode
- Recent advances try to reduce variance of the gradient estimator, and tries to make SGD capture multi-modality.

Many more interesting approaches..

- **SWAG:** Maddox, 2019 builds on stochastic weight averaging by fitting a low-rank + diagonal to the posterior covariance
- **Deep Kernel Learning:** Wilson, 2016 provide a scalable way to learn deep representations of spectral mixture kernels
- **Functional Space VI:** Sun, 2019 perform VI on the outputs of the NN (rather than weights)
- **Bayesian/non-Bayesian Ensembling:** Lakshminarayan, 2017 propose deep ensemble
- **Many more:** Bayesian Boosting, Neural-net GPs, NF, etc.

(a) Confident Prediction (b) High data uncertainty (c) Out-of-distribution

Figure 2: Desired behaviors of a distribution over distributions



Thank You!!