Probabilistic tools for improving deep learning models

Dmitry P. Vetrov

Research professor at HSE, Lab lead in SAIC-Moscow

Head of BayesGroup

http://bayesgroup.ru

Outline

- Introductory remarks
- Implicit Jeffreys Auto-encoder
- Implicit Metropolis-Hastings algorithm
- Ensembles for uncertainty estimation

Modeling of probabilistic distributions

- In many ML applications we need to model or samplers for complicated distributions

- **Density-based setting.** The distribution is given in explicit form with unknown normalization constant

$$p(x) = \frac{1}{Z}\hat{p}(x)$$

- Example: Bayesian DNN

$$p(w|X,T) = \frac{p(T|X,w)p(w)}{\int p(T|X,w)p(w)dw}$$

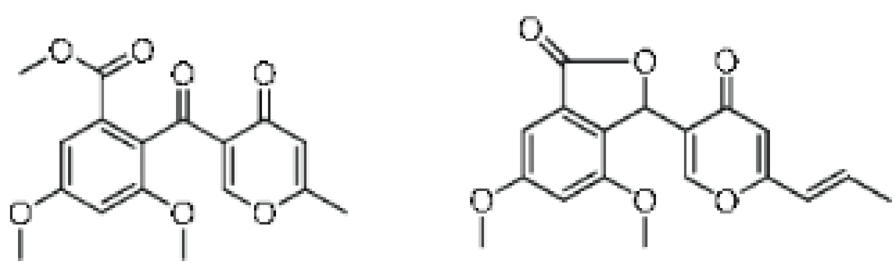
- The dimension of w is huge and $p(w|X,T)$ has exponentially many modes

Modeling of probabilistic distributions

- In many ML applications we need to models or samplers for complicated distributions
- **Samlple-based setting.** The distribution is given in implicit form as set of samples generated from it

$$(x_1, \ldots, x_n) \sim p(x), \quad x \in \mathcal{X}$$

- Example: Generative adversarial network, variational auto-encoder
- The space $\mathcal{X}$ usually is very complex, e.g. pictures, music, molecules, etc.



KL-divergence

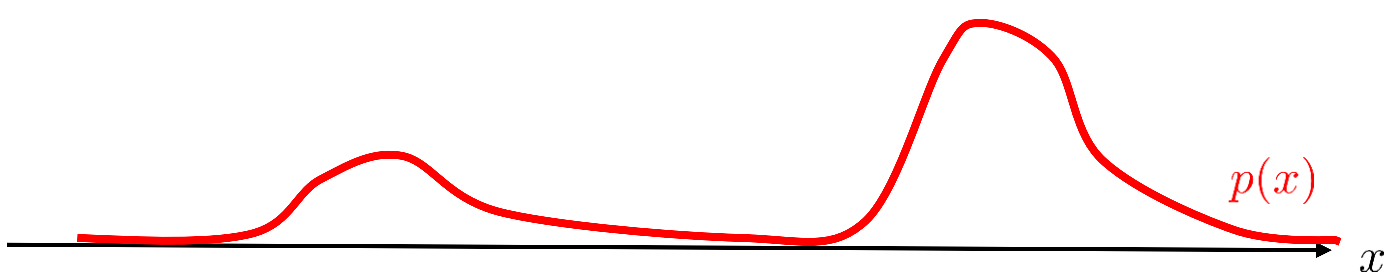
- A good mismatch measure between two distributions over **the same domain**

$$KL(q(x)||p(x)) = \int q(x) \log \frac{q(x)}{p(x)} dx = \mathbb{E}_q \log \frac{q(x)}{p(x)} \geq 0$$

- Information-theoretic interpretation

$$\mathtt{KL} = \mathtt{CrossEntropy} - \mathtt{Entropy}$$

- If we minimize $KL(q||p)$ w.r.t. $q(.)$ the approximation should be good where $q(x)$ has large values



KL-divergence

- A good mismatch measure between two distributions over **the same domain**

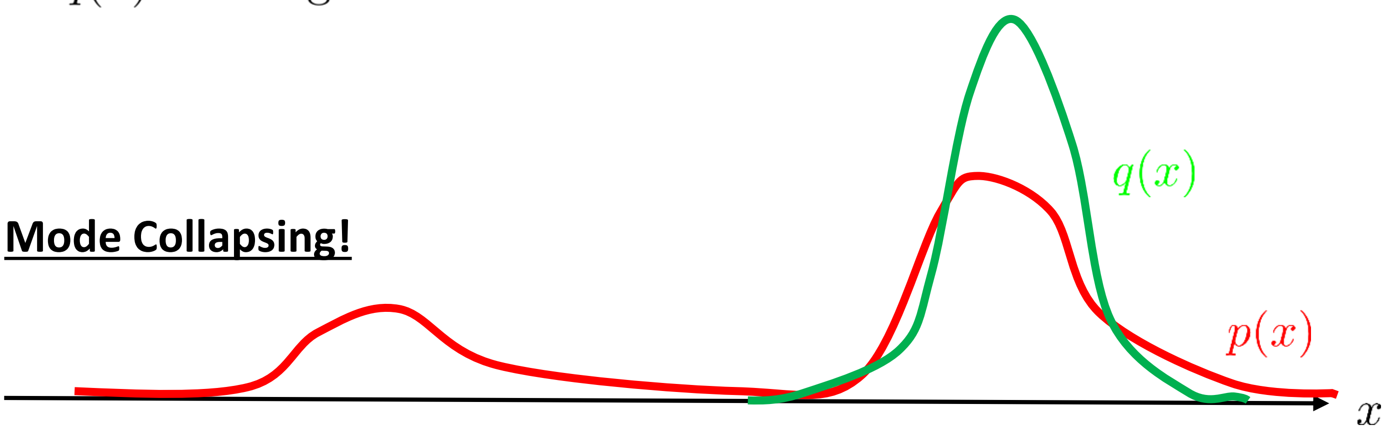
$$KL(q(x)||p(x)) = \int q(x) \log \frac{q(x)}{p(x)} dx = \mathbb{E}_q \log \frac{q(x)}{p(x)} \geq 0$$

- Information-theoretic interpretation

$$\texttt{KL} = \texttt{CrossEntropy} - \texttt{Entropy}$$

- If we minimize $KL(q||p)$ w.r.t. $q(.)$ the approximation should be good where $q(x)$ has large values

Mode Collapsing!



KL-divergence

- A good mismatch measure between two distributions over **the same domain**

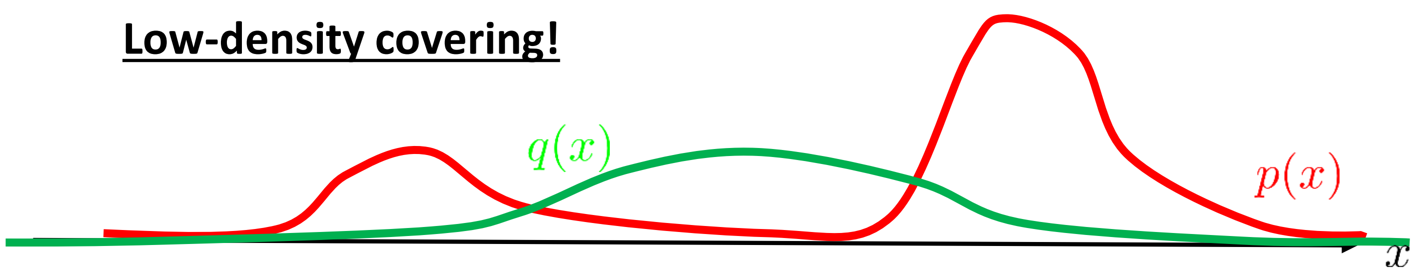
$$KL(q(x)||p(x)) = \int q(x) \log \frac{q(x)}{p(x)} dx = \mathbb{E}_q \log \frac{q(x)}{p(x)} \geq 0$$

- Information-theoretic interpretation

$$\mathtt{KL} = \mathtt{CrossEntropy} - \mathtt{Entropy}$$

- If we minimize $KL(p||q)$ w.r.t. $q(.)$ the approximation should be good where $p(x)$ has large values

Low-density covering!



GAN

- We are given set of objects x_i from $p(x)$
- We have generator $G_\theta(\xi)$, $\xi \sim \mathcal{N}(0, I)$, that induces distribution $q(x)$
- We train discriminator to distinguish between real and synthetic objects

$$D_\eta(x): \ \mathbb{E}_{p(x)} \log D_\eta(x) + \mathbb{E}_\xi \log(1 - D_\eta(G_\theta(\xi)) \to \max_\eta$$

- Optimal discriminator

$$D_*(x) = \frac{p(x)}{q(x)}$$

- Generator is trained to cheat discriminator

$$G_\theta(\xi) : \mathbb{E}_\xi \log(1 - D_\eta(G_\theta(\xi))) \to \min_\theta$$

GAN

- We are given set of objects x_i from $p(x)$
- We have generator $G_\theta(\xi)$, $\xi \sim \mathcal{N}(0, I)$, that induces distribution $q(x)$
- We train discriminator to distinguish between real and synthetic objects

$$D_\eta(x):\ \mathbb{E}_{p(x)} \log D_\eta(x) + \mathbb{E}_\xi \log(1 - D_\eta(G_\theta(\xi)) \to \max_\eta$$

- Optimal discriminator

$$D_*(x) = \frac{p(x)}{q(x)}$$

- Better gradients **(heuristic!)**

$$G_\theta(\xi) : -\mathbb{E}_\xi \log(D_\eta(G_\theta(\xi))) \to \min_\theta$$

GAN

- We are given set of objects x_i from $p(x)$
- We have generator $G_\theta(\xi)$, $\xi \sim \mathcal{N}(0, I)$, that induces distribution $q(x)$
- We train discriminator to distinguish between real and synthetic objects

$$D_\eta(x): \ \mathbb{E}_{p(x)} \log D_\eta(x) + \mathbb{E}_\xi \log(1 - D_\eta(G_\theta(\xi)) \to \max_\eta$$

- Optimal discriminator

$$D_*(x) = \frac{p(x)}{q(x)}$$

- If we sum the two

$$G_\theta(\xi) : \mathbb{E}_\xi \log(1 - D_\eta(G_\theta(\xi))) - \mathbb{E}_\xi \log(D_\eta(G_\theta(\xi))) \to \min_\theta$$

- Then assuming the optimal $D_*(x)$ we may show that

$$\theta = \arg\min KL(q(x)||p(x))$$

VAE

- Variational auto-encoder maximizes so-called Evidence Lower BOund (ELBO)
- This is lower bound on
$$\mathbb{E}_{p(x)} \log q_\theta(x),$$
where $q_\theta(x)$ is a distribution induced by VAE
- We may easily show

$$\arg\max_\theta \mathbb{E}_{p(x)} \log q_\theta(x) = \arg\max_\theta \mathbb{E}_{p(x)} \left[\log q_\theta(x) - \log p(x)\right] =$$
$$\arg\min_\theta \mathbb{E}_{p(x)} \log \frac{p(x)}{q_\theta(x)} = \arg\min_\theta KL(p(x)||q_\theta(x))$$

- VAE **always** covers the whole dataset

Pros and cons

VAE

- Reconstruction term
- Learned latent representations
- Unrealistic explicit likelihood of decoder

GAN

- More realistic implicit likelihood
- No covering of training data

Taking the best of the two worlds

- Implicit encoder $z = E_\phi(x, \xi) \sim q_\phi(z|x)$, where $\xi \sim \mathcal{N}(0, I)$
- Implicit generator (decoder) $\hat{x} = G_\theta(z)$
- Objective for generator

$$(1-\lambda)\mathbb{E}_{p(z)} \log \frac{D_\tau(G_\theta(z))}{1 - D_\tau(G_\theta(z))} + \lambda \mathbb{E}_{p(x)} \mathbb{E}_{q(z|x)} \frac{1 - D_\psi(x, z, G_\theta(z))}{D_\psi(x, z, G_\theta(z))}$$

Taking the best of the two worlds

- Implicit encoder $z = E_\phi(x, \xi) \sim q_\phi(z|x)$, where $\xi \sim \mathcal{N}(0, I)$
- Implicit generator (decoder) $\hat{x} = G_\theta(z)$
- Objective for generator

$$(1-\lambda)\boxed{\mathbb{E}_{p(z)} \log \frac{D_\tau(G_\theta(z))}{1 - D_\tau(G_\theta(z))}} + \lambda\boxed{\mathbb{E}_{p(x)} \mathbb{E}_{q(z|x)} \frac{1 - D_\psi(x, z, G_\theta(z))}{D_\psi(x, z, G_\theta(z))}}$$

GAN objective – ensures realistic quality of generated samples

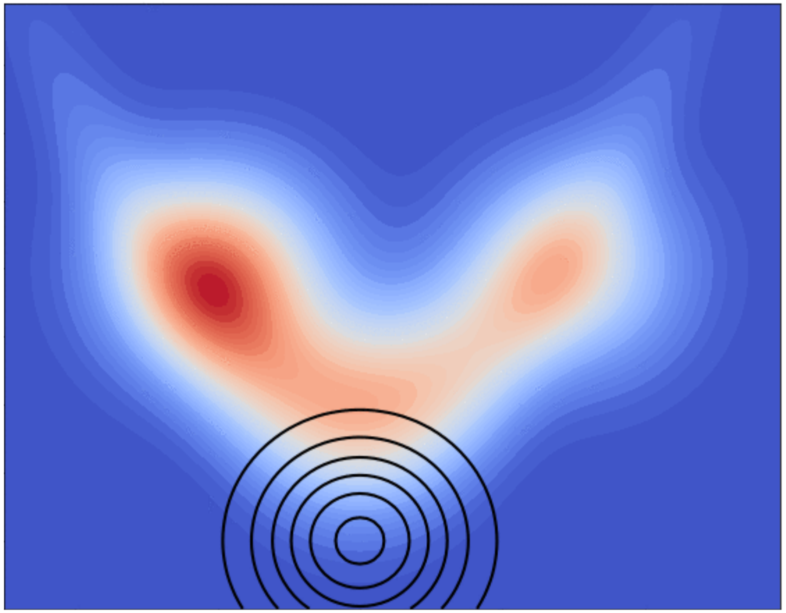
Implicit reconstruction term – ensures coverage of the whole dataset

Results

Method	Generation Quality IS ↑	Reconstruction Quality LPIPS ↓
CIFAR 10		
WAE (Tolstikhin et al., 2017)	4.18 ± 0.04	
ALI (Dumoulin et al., 2017))	5.34 ± 0.04	
ALICE (Li et al., 2017)	6.02 ± 0.03	
AS-VAE (Pu et al., 2017b)	6.3	
VAE (resnet)	3.45 ± 0.02	0.09 ± 0.03
2Stage-VAE (Dai & Wipf, 2019)	3.85 ± 0.03	0.06 ± 0.03
α-GAN (Rosca et al., 2017)	5.20 ± 0.08	0.04 ± 0.02
AGE (Ulyanov et al., 2018)	5.90 ± 0.04	0.06 ± 0.02
SVAE (Chen et al., 2018)	6.56 ± 0.07	0.19 ± 0.08
λ-IJAE ($\lambda = 0.3$)	$\mathbf{6.98} \pm \mathbf{0.1}$	0.07 ± 0.03
TinyImagenet		
AGE (Ulyanov et al., 2018)	6.75 ± 0.09	0.27 ± 0.09
SVAE (Chen et al., 2018)	5.09 ± 0.05	0.28 ± 0.08
2Stage-VAE (Dai & Wipf, 2019)	4.22 ± 0.05	$\mathbf{0.09} \pm \mathbf{0.05}$
λ-IJAE ($\lambda = 0.3$)	$\mathbf{6.87} \pm 0.09$	$\mathbf{0.09} \pm \mathbf{0.03}$

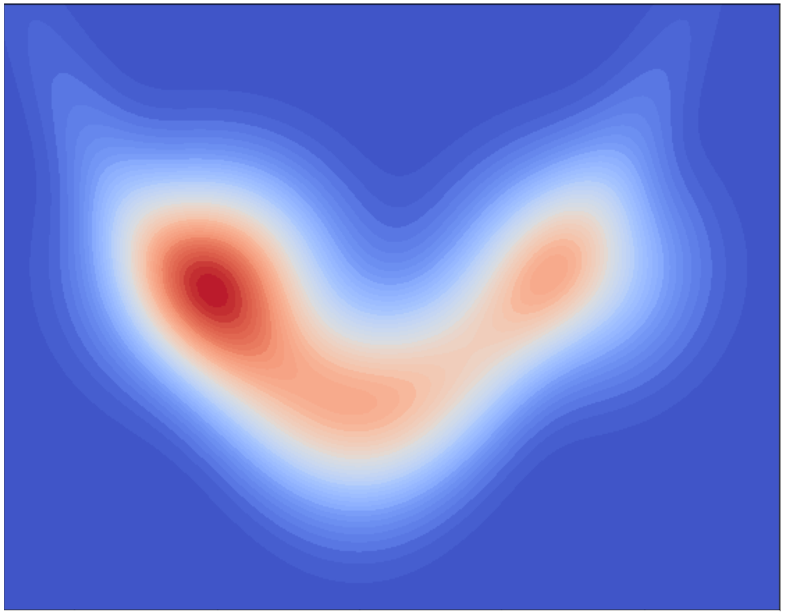
Main approximate inference tools

Variational inference



- Approximates intractable true posterior with a tractable variational distribution
- Typically KL-divergence is minimized
- Can be scaled up by stochastic optimization

Markov Chain Monte Carlo



- Generates samples from the true posterior
- No bias even if the true distribution is intractable
- Quite slow in practice
- Problematic scaling to large data

Metropolis-Hastings algorithm

- Supppose we want to sample from (un-normalized) distribution $\hat{p}(x)$
- Establish so-called **proposal distribution** $q(y)$ that is easy to sample from
- Generate sequence of points $y_1, \ldots, y_n, \ldots$ from $q(y)$ and form a chain $x_1, \ldots, x_n, \ldots$ as follows

$$x_n = \begin{cases} y_n, & \text{with probability } A = \min\left(1, \frac{\hat{p}(y_n)q(x_{n-1})}{q(y_n)\hat{p}(x_{n-1})}\right) \\ x_{n-1}, & \text{otherwise} \end{cases}$$

- Extendable for the case when proposal distribution depends on the current point $q(y|x_{n-1})$

Metropolis-Hastings algorithm

- Generate sequence of points $y_1, \ldots, y_n, \ldots$ from $q(y)$ and form a chain $x_1, \ldots, x_n, \ldots$ as follows

$$x_n = \begin{cases} y_n, & \text{with probability } A = \min\left(1, \frac{\hat{p}(y_n)q(x_{n-1})}{q(y_n)\hat{p}(x_{n-1})}\right) \\ x_{n-1}, & \text{otherwise} \end{cases}$$

- The scheme is efficient when the number of rejections of y_n is small

- Let us try to maximize **acceptance rate** in MH algorithm w.r.t. the parameters θ of proposal distribution $q_\theta(y)$

Metropolis-Hastings algorithm

- Generate sequence of points $y_1, \dots, y_n, \dots$ from $q(y)$ and form a chain $x_1, \dots, x_n, \dots$ as follows

$$x_n = \begin{cases} y_n, & \text{with probability } A = \min\left(1, \frac{\hat{p}(y_n)q(x_{n-1})}{q(y_n)\hat{p}(x_{n-1})}\right) \\ x_{n-1}, & \text{otherwise} \end{cases}$$

- The scheme is efficient when the number of rejections of y_n is small

- Let us try to maximize **acceptance rate** in MH algorithm w.r.t. the parameters θ of proposal distribution $q_\theta(y)$

$$AR = \int p(x)q_\theta(y) \min\left(1, \frac{\hat{p}(y)q_\theta(x)}{q_\theta(y)\hat{p}(x)}\right) dydx$$

Acceptance rate maximization

Let us try to maximize **acceptance rate** in MH algorithm w.r.t. the parameters θ of proposal distribution $q_\theta(y)$

$$AR = \int p(x) q_\theta(y) \min\left(1, \frac{q_\theta(y)\hat{p}(x)}{\hat{p}(y)q_\theta(x)}\right) dydx$$

Acceptance rate maximization

Let us try to maximize **acceptance rate** in MH algorithm w.r.t. the parameters θ of proposal distribution $q_\theta(y)$

$$AR = \int p(x)q_\theta(y)\min\left(1, \frac{q_\theta(y)\hat{p}(x)}{\hat{p}(y)q_\theta(x)}\right) dydx =$$

$$= 1 - \mathrm{TV}(p(x)q_\theta(y)||q_\theta(x)p(y))$$

Acceptance rate maximization

Let us try to maximize **acceptance rate** in MH algorithm w.r.t. the parameters θ of proposal distribution $q_\theta(y)$

$$AR = \int p(x)q_\theta(y)\min\left(1, \frac{q_\theta(y)\hat{p}(x)}{\hat{p}(y)q_\theta(x)}\right)dydx =$$

$$= 1 - \mathrm{TV}(p(x)q_\theta(y)||q_\theta(x)p(y)) \geq 1 - \sqrt{\frac{1}{2}KL(p(x)q_\theta(y)||q_\theta(x)p(y))}$$

Acceptance rate maximization

Let us try to maximize **acceptance rate** in MH algorithm w.r.t. the parameters θ of proposal distribution $q_\theta(y)$

$$AR = \int p(x) q_\theta(y) \min\left(1, \frac{q_\theta(y)\hat{p}(x)}{\hat{p}(y) q_\theta(x)}\right) dy dx =$$

$$= 1 - \mathrm{TV}(p(x)q_\theta(y) || q_\theta(y)p(x)) \geq 1 - \sqrt{\frac{1}{2} KL(p(x)q_\theta(y) || q_\theta(y)p(x))} \geq$$

$$\geq 1 - \sqrt{\frac{1}{2}\left(KL(p(x)||q_\theta(x)) + KL(q_\theta(y)||p(y))\right)}$$

Acceptance rate maximization

Let us try to maximize **acceptance rate** in MH algorithm w.r.t. the parameters θ of proposal distribution $q_\theta(y)$

$$AR = \int p(x)q_\theta(y)\min\left(1, \frac{q_\theta(y)\hat{p}(x)}{\hat{p}(y)q_\theta(x)}\right)dydx =$$

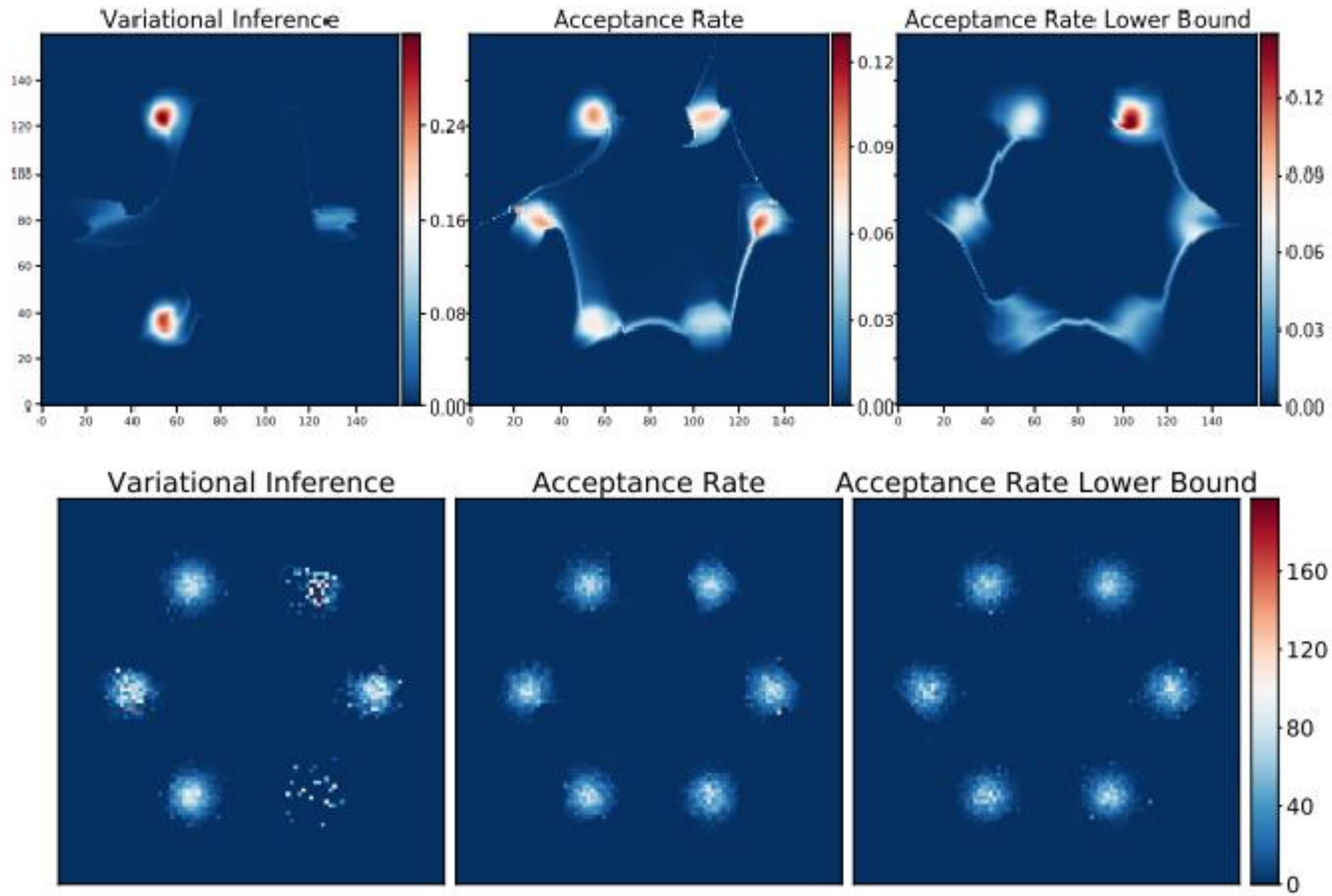
$$= 1 - \mathrm{TV}(p(x)q_\theta(y)||q_\theta(x)p(y)) \geq 1 - \sqrt{\frac{1}{2}KL(p(x)q_\theta(y)||q_\theta(x)p(y))} \geq$$

$$\geq 1 - \sqrt{\frac{1}{2}\left(KL(p(x)||q_\theta(x)) + KL(q_\theta(y)||p(y))\right)}$$

Maximization of acceptance rate in MH algorithm is equivalent to minimization of Jeffreys divergence!

$$KL(p||q_\theta) + KL(q_\theta||p) \to \min_\theta$$

Results on toy problem



Implicit setting

- Now consider the case when we're given only a set of samples $(x_1, \ldots, x_n)$ from the distribution $p(x)$
- Let us try to use **implicit proposal** $q(y)$, where $y = G_\theta(\xi)$, and $\xi \sim \mathcal{N}(0, I)$
- In MH algorithm the probability of acceptance $A = \min\left(1, \frac{q(y_n)\hat{p}(x_{n-1})}{\hat{p}(y_n)q(x_{n-1})}\right)$ is given by **density ratios**

Implicit setting

- Now consider the case when we're given only a set of samples $(x_1, \ldots, x_n)$ from the distribution $p(x)$
- Let us try to use **implicit proposal** $q(y)$, where $y = G_\theta(\xi)$, and $\xi \sim \mathcal{N}(0, I)$
- In MH algorithm the probability of acceptance $A = \min\left(1, \frac{q(y_n)\hat{p}(x_{n-1})}{\hat{p}(y_n)q(x_{n-1})}\right)$ is given by **density ratios**
- To get density ratios we may train discriminator to distinguish between x and y
- The optimal discriminator yields

$$D_*(x) = \frac{p(x)}{p(x) + q(x)}, \quad \text{i.e.} \quad \frac{p(x)}{q(x)} = \frac{D_*(x)}{1 - D_*(x)}$$

Reduction to adversarial training

- In MH algorithm the probability of acceptance $A = \min\left(1, \frac{q(y_n)p(x_{n-1})}{p(y_n)q(x_{n-1})}\right)$ is given by **density ratios**
- The optimal discriminator yields

$$D_*(x) = \frac{p(x)}{p(x) + q(x)}, \quad \text{i.e.} \quad \frac{p(x)}{q(x)} = \frac{D_*(x)}{1 - D_*(x)}$$

Reduction to adversarial training

- In MH algorithm the probability of acceptance $A = \min\left(1, \frac{q(y_n)p(x_{n-1})}{p(y_n)q(x_{n-1})}\right)$ is given by **density ratios**
- The optimal discriminator yields

$$D_*(x) = \frac{p(x)}{p(x) + q(x)}, \quad \text{i.e.} \quad \frac{p(x)}{q(x)} = \frac{D_*(x)}{1 - D_*(x)}$$

- Remember acceptance rate is lower bounded by

$$-KL(p(x)q(y)||p(y)q(x)) = -\int p(x)q(y) \log\left[\left(\frac{p(x)}{q(x)}\right)\left(\frac{q(y)}{p(y)}\right)\right] dydx$$

Reduction to adversarial training

- In MH algorithm the probability of acceptance $A = \min\left(1, \frac{q(y_n)p(x_{n-1})}{p(y_n)q(x_{n-1})}\right)$ is given by **density ratios**
- The optimal discriminator yields

$$D_*(x) = \frac{p(x)}{p(x)+q(x)}, \quad \text{i.e.} \quad \frac{p(x)}{q(x)} = \frac{D_*(x)}{1-D_*(x)}$$

- Remember acceptance rate is lower bounded by

$$-KL(p(x)q(y)||p(y)q(x)) = -\int p(x)q(y)\log\left[\left(\frac{p(x)}{q(x)}\right)\left(\frac{q(y)}{p(y)}\right)\right]dydx$$

- Using the fact that $y = G_\theta(\xi)$, $\xi \sim r(\xi)$ we may rewrite the lower bound

$$-\int p(x)r(\xi)\log\left[\left(\frac{D_*(x)}{1-D_*(x)}\right)\left(\frac{D_*(G_\theta(\xi))}{1-D_*(G_\theta(\xi))}\right)\right]d\xi dx \to \min_\theta$$

Reduction to adversarial training

- Using the fact that $y = G_\theta(\xi)$, $\xi \sim r(\xi)$ we may rewrite the lower bound

$$-\int p(x)r(\xi)\log\left[\left(\frac{D_*(x)}{1-D_*(x)}\right)\left(\frac{D_*(G_\theta(\xi))}{1-D_*(G_\theta(\xi))}\right)\right]d\xi dx \to \min_\theta$$

Reduction to adversarial training

- Using the fact that $y = G_\theta(\xi)$, $\xi \sim r(\xi)$ we may rewrite the lower bound

$$-\int p(x)r(\xi)\log\left[\left(\frac{D_*(x)}{1-D_*(x)}\right)\left(\frac{D_*(G_\theta(\xi))}{1-D_*(G_\theta(\xi))}\right)\right]d\xi dx \to \min_\theta$$

- Now remove everything that does not depend on θ

$$\mathbb{E}_\xi\left[\log D_*(G_\theta(\xi)) - \log(1 - D_*(G_\theta(\xi))\right] \to \max_\theta$$

- The same objective as in GAN!

MH GAN

Metropolis-Hastings GAN

- Train discriminator and generator

$$\tau = \arg\max \left[\sum_{i=1}^{n} \log D_\tau(x_i) + n\mathbb{E}_\xi \log(1 - D_\tau(G_\theta(\xi))) \right]$$

$$\theta = \arg\max \mathbb{E}_\xi \left[\log D_\tau(G_\theta(\xi)) - \log(1 - D_\tau(G_\theta(\xi))) \right]$$

- Generate new samples $(z_1, \ldots, z_m)$ via filtering procedure

MH GAN

Metropolis-Hastings GAN

- Train discriminator and generator

$$\tau = \arg\max \left[\sum_{i=1}^{n} \log D_\tau(x_i) + n\mathbb{E}_\xi \log(1 - D_\tau(G_\theta(\xi))) \right]$$

$$\theta = \arg\max \mathbb{E}_\xi \left[\log D_\tau(G_\theta(\xi)) - \log(1 - D_\tau(G_\theta(\xi))) \right]$$

- Generate new samples $(z_1, \ldots, z_m)$ via filtering procedure
- First generate $(y_1, \ldots, y_m)$ from $G(\xi)$, then set

$$z_k = \begin{cases} y_k, & \text{with probability} \quad A = \min\left(1, \frac{D_\tau(y_k)(1-D_\tau(z_{k-1}))}{D_\tau(z_{k-1})(1-D_\tau(y_k))}\right) \\ z_{k-1}, & \text{otherwise} \end{cases}$$

MH GAN

Table 3: Comparison of sampling using the MH algorithm and using the generator for different models. Low FID and high IS are better. For a single evaluation of metrics on CIFAR-10 and CelebA datasets, we use 10k samples, and on ImageNet, we use 50k samples. Then we average all the values across 5 independent runs. See the description of models in the text.

		DCGAN		**WPGAN**		**ARLB**		**BigGAN-100k**	**BigGAN-138k**
		CIFAR-10	CelebA	CIFAR-10	CelebA	CIFAR-10	CelebA	ImageNet	ImageNet
FID	Generator	49.06 ± 0.34	14.91 ± 0.16	47.38 ± 0.21	39.09 ± 0.33	46.55 ± 0.21	17.25 ± 0.07	11.74 ± 0.06	9.92 ± 0.06
	MH(ours)	$\mathbf{46.12} \pm 0.29$	$\mathbf{12.70} \pm 0.13$	$\mathbf{36.65} \pm 0.28$	$\mathbf{25.41} \pm 0.28$	$\mathbf{45.71} \pm 0.46$	$\mathbf{16.57} \pm 0.16$	$\mathbf{10.80} \pm 0.04$	$\mathbf{9.52} \pm 0.04$
IS	Generator	3.64 ± 0.02	2.51 ± 0.01	3.52 ± 0.02	2.05 ± 0.01	3.59 ± 0.01	2.38 ± 0.02	74.03 ± 0.74	97.73 ± 0.55
	MH(ours)	$\mathbf{3.86} \pm 0.06$	$\mathbf{2.73} \pm 0.01$	$\mathbf{4.02} \pm 0.03$	$\mathbf{2.54} \pm 0.01$	$\mathbf{3.72} \pm 0.04$	$\mathbf{2.47} \pm 0.01$	$\mathbf{82.10} \pm 0.56$	$\mathbf{105.62} \pm 0.74$

Acceptance rate is about 10%

Only unique objects were used for evaluating the metrics

Uncertainty estimation

- Modern deep neural networks are highly over-confident even when they are wrong
- The obvious way to improve their uncertainty estimation is to use ensembles

$$p(t_*|x_*) = \mathbb{E}_w p(t_*|x_*, w) = \int p(t_*|x_*, w) q(w) dw,$$

  but how to get the distribution $q(w)$?

- Two main strategies: Non-Bayesian and Bayesian

Non-Bayesian way: deep ensembles

- In Non-Bayesian setting we simply K independently trained DNNs

$$q(w) = \frac{1}{K}\sum_{k=1}^{K}\delta(w - w^k)$$

- Requires K times more computational time
- Requires K times more memory

Bayesian way: inferring posterior

- In Bayesian setting we simply apply Bayes theorem

$$q(w) = \frac{p(T_{train}|X_{train}, w)p(w)}{\int p(T_{train}|X_{train}, w)p(w)dw}$$

- Posterior is extremely complicated
- Exact inference is far from being tractable
- We can come up with approximations using computationally efficient stochastic variational inference tools

Exponential number of symmetries

- There is exponential number of equivalent solutions due to huge over-parameterization
- Maybe they are just reflections of single mode?..

Estimation metrics

- Validation log-likelihood
- Brier score
- Misclassification detection
- Threshold-based rejection
- Calibration metrics

Estimation metrics

- Validation log-likelihood - requires temperature scaling
- Brier score - correlated with previous metric
- Misclassification detection - cannot compare different tools for UE
- Threshold-based rejection - cannot compare different tools for UE
- Calibration metrics - gives different answers depending on the choice of hyperparameters

Temperature scaling

- After training is finished we need to recalibrate the output probabilities $\hat{p}(t|x) = \mathrm{softmax}(logit_1(x), \ldots, logit_L(x))$
- We set temperature τ and recompute probabilities as follows

$$p(t|x) = \mathrm{softmax}\left(\frac{1}{\tau} logit_1(x), \ldots, \frac{1}{\tau} logit_L(x)\right)$$

- Temperature is adjusted to maximize likelihood on validation set

$$\tau = \arg\max p(T_{val}|X_{val})$$

- We use 2-fold cross-validation to remove bias
- Even if all networks are calibrated we need to perform additional calibration after ensembling

Experiment design

We focus on in-domain uncertainty and explore most successful techniques for building ensembles

- Deep ensembles
- Snapshot ensembles (SSE)
- Cyclic SGLD (CSGLD)
- Fast geometric ensembling (FGE)
- SWA-gaussian (SWAG)
- VI with gaussian (FFG VI)
- Dropout
- Test time data augmentation

Covers multiple modes

Memory-efficient

Covers single mode

Deep ensemble equivalent

- We measure the quality of UE using calibrated log-likelihood (CLL) on validation set
- To compare different ensembling techniques we establish **deep ensemble equivalent**

$$\mathrm{DEE}_m(k) = \min\left\{ l \in \mathbb{R}, l \geq 1 \,\middle|\, \mathrm{CLL}_{DE}^{\mathbf{mean}}(l) \geq \mathrm{CLL}_m^{\mathbf{mean}}(k) \right\}$$

- That is how many samples from deep ensemble provide us the same UE as k samples from ensemble m

Deep ensemble equivalent

- We measure the quality of UE using calibrated log-likelihood (CLL) on validation set
- To compare different ensembling techniques we establish **deep ensemble equivalent**

$$\mathrm{DEE}_m(k) = \min\left\{ l \in \mathbb{R}, l \geq 1 \,\middle|\, \mathrm{CLL}_{DE}^{\mathbf{mean}}(l) \geq \mathrm{CLL}_{m}^{\mathbf{mean}}(k) \right\}$$

- That is how many samples from deep ensemble provide us the same UE as k samples from ensemble m
- Empirical study of VGG16, PreResNet110/164, WideResNet28x10 on CIFAR10/100 dataset
- Empirical study of ResNet50 on ImageNet
- Several thousands of different DNNs trained!

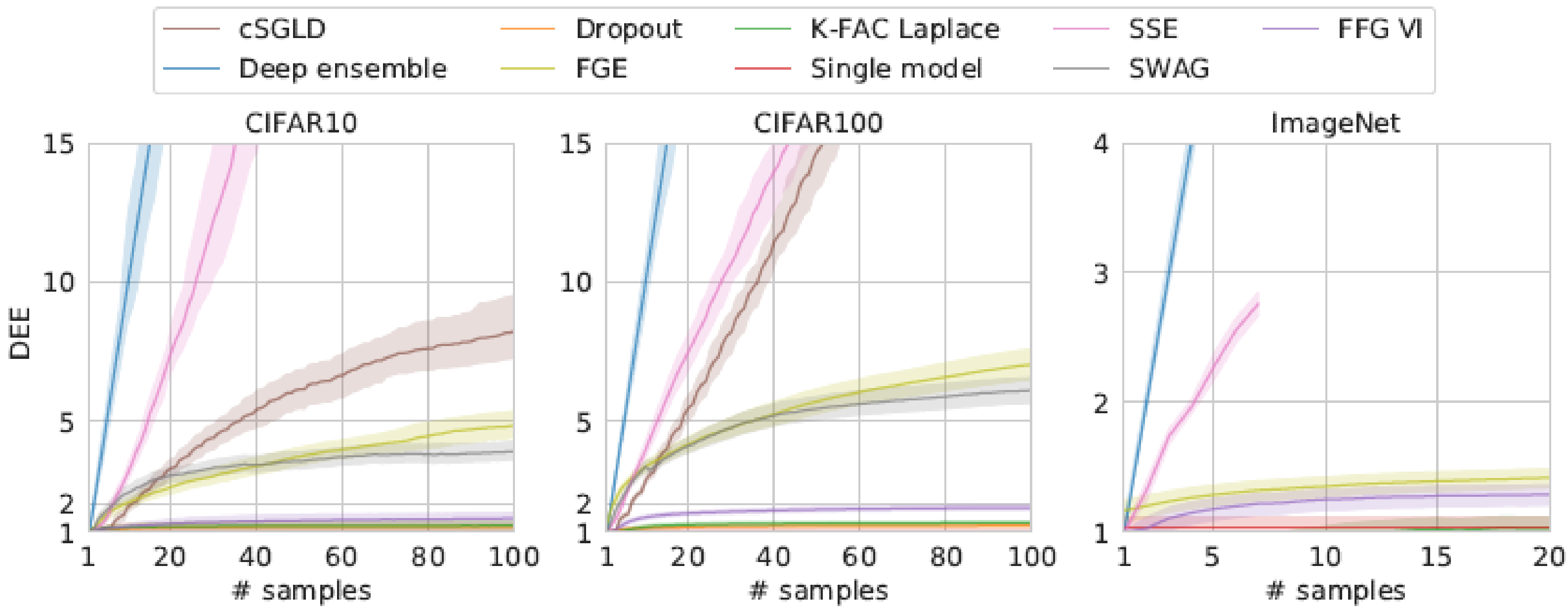
Test time data augmentation

Model	cSGLD	Deep ensemble	FGE	K-FAC-L	Single model	SSE	SWAG	FFG VI	Dropout
ResNet110	0.115 vs 0.111↓	0.106 vs 0.105↓	0.121 vs 0.121≈	0.147 vs 0.130↓	0.150 vs 0.129↓	0.106 vs 0.106≈	0.126 vs 0.126↓	0.142 vs 0.130↓	
ResNet164	0.110 vs 0.108↓	0.100 vs 0.100≈	0.115 vs 0.115≈	0.142 vs 0.127↓	0.144 vs 0.124↓	0.104 vs 0.104≈	0.116 vs 0.116≈	0.141 vs 0.128↓	
VGG16	0.147 vs 0.146↓	0.138 vs 0.139↑	0.150 vs 0.150≈	0.200 vs 0.164↓	0.229 vs 0.170↓	0.137 vs 0.138↑	0.152 vs 0.152≈	0.184 vs 0.160↓	0.226 vs 0.171↓
WideResNet	0.099 vs 0.100↑	0.090 vs 0.094↑	0.102 vs 0.102≈	0.120 vs 0.111↓	0.124 vs 0.113↓		0.101 vs 0.101≈	0.117 vs 0.117≈	0.118 vs 0.111↓

Table 1: Negative calibrated log-likelihood for CIFAR10, w/o vs with test-time data augmentation.

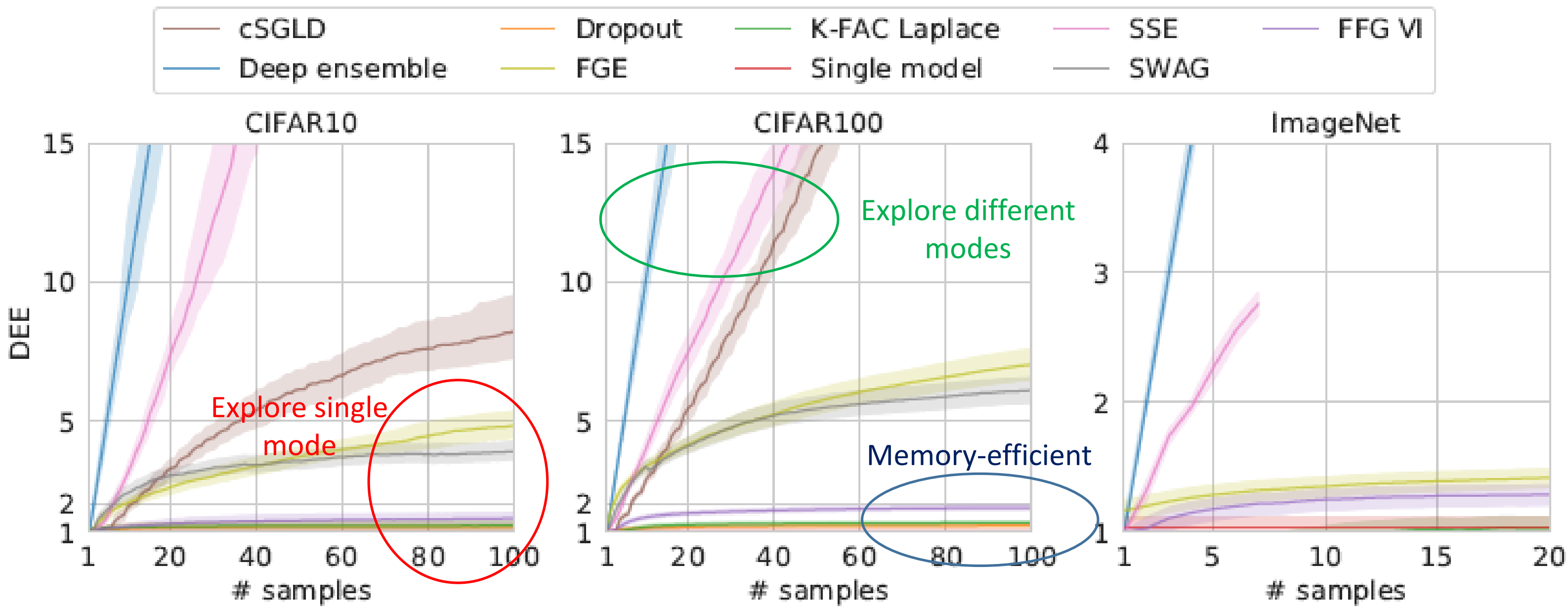
Data augmentation surprisingly helps to almost all ensembling tools

Results



cSGLD
Deep ensemble
Dropout
FGE
K-FAC Laplace
Single model
SSE
SWAG
FFG VI
CIFAR10
CIFAR100
ImageNet
DEE
samples

Results



- Deep ensembles are far beyond all analogues given time budget
- Global methods are better than local
- Cannot do good ensembling without memory costs
- Similar conclusions for out-domain UE independently reported by Google