Nadia Chirkova

Bayesian neural networks

2021

Slides are partially based on lectures of Dmitry Vetrov, Dmitry Kropotov and Kirill Struminsky, deepbayes.ru/2018

Plan

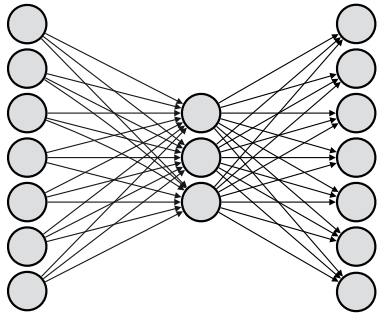
- Advantages of using Bayesian neural networks (30 min)
- Training Bayesian neural networks (30 min)
- Q&A + exercises (30 min)

Plan

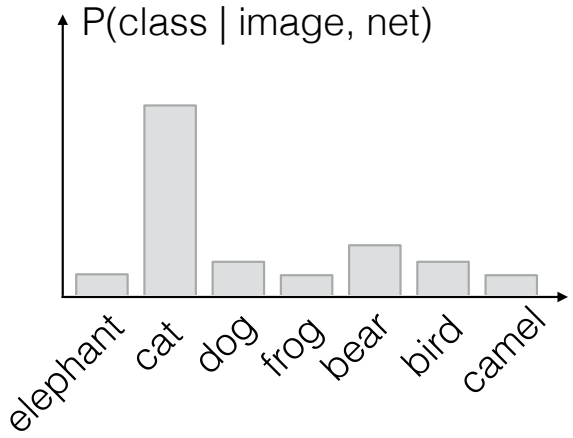
- Advantages of using Bayesian neural networks
- Training Bayesian neural networks
- Q&A + exercises

Neural networks

input x

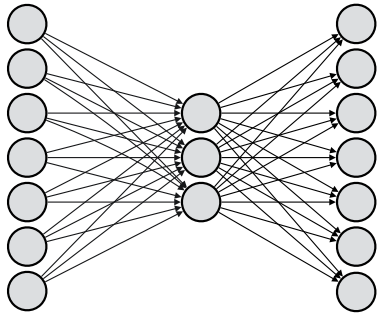


neural network
with weights w

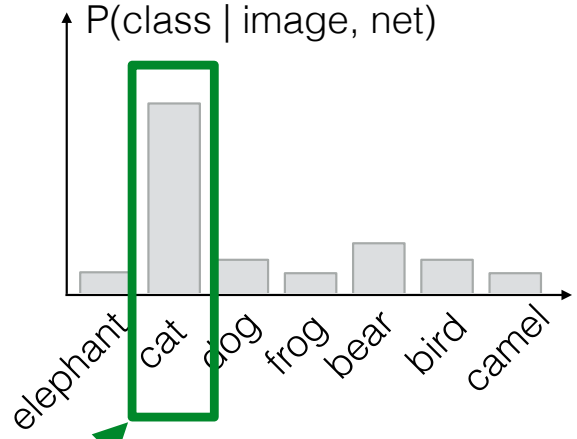


Neural networks

input x



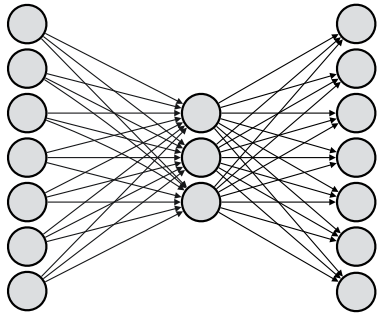
neural network
with weights w



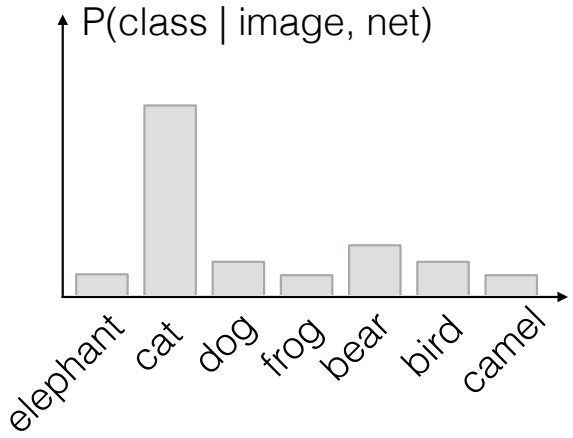
$$p(y = \text{“}cat\text{”} \,|\, x, w)$$

Neural networks

input x



neural network
with weights w

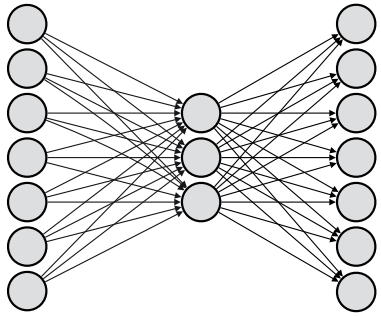


Training — optimization over weights w using stochastic gradient descend:

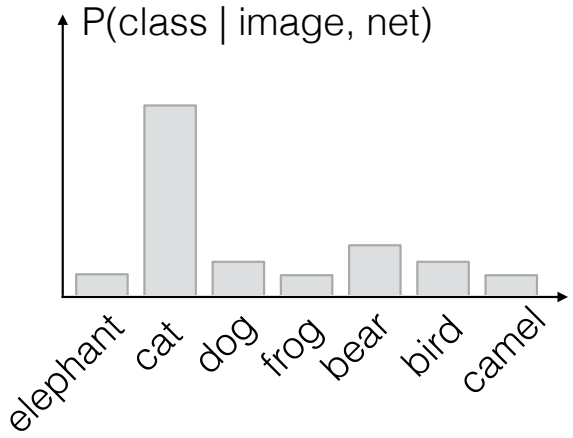
$$-DataLoss(X, Y, w) \to \max_w$$

Neural networks

input x



neural network
with weights w



Training — optimization over weights w using stochastic gradient descend:

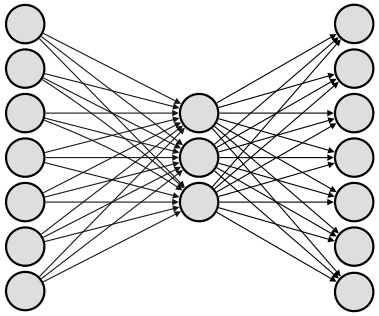
$$\sum_{i=1}^{N} \log p(y^i | x^i, w) \to \max_w$$

sum over objects (images)

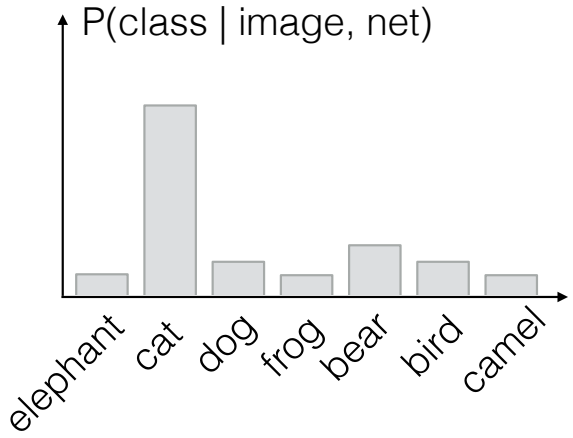
probability of true class

Neural networks

input x



neural network
with weights w



Training — optimization over weights w using stochastic gradient descend:

$$w^{new} = w^{old} + \eta \frac{\partial}{\partial w} \sum_{j=1}^{m} \log p(y^{i_j} | x^{i_j}, w^{old})$$

$$i_j \sim \text{Unif}(1, \ldots, N)$$

m — mini-batch size η — learning rate

Regularization by noise

- Traditional regularization: add some penalty for model complexity
  - L_2, L_1 - regularization, max norm constraint

$$Objective = DataLoss(X, Y, w) + Regularizer(w)$$

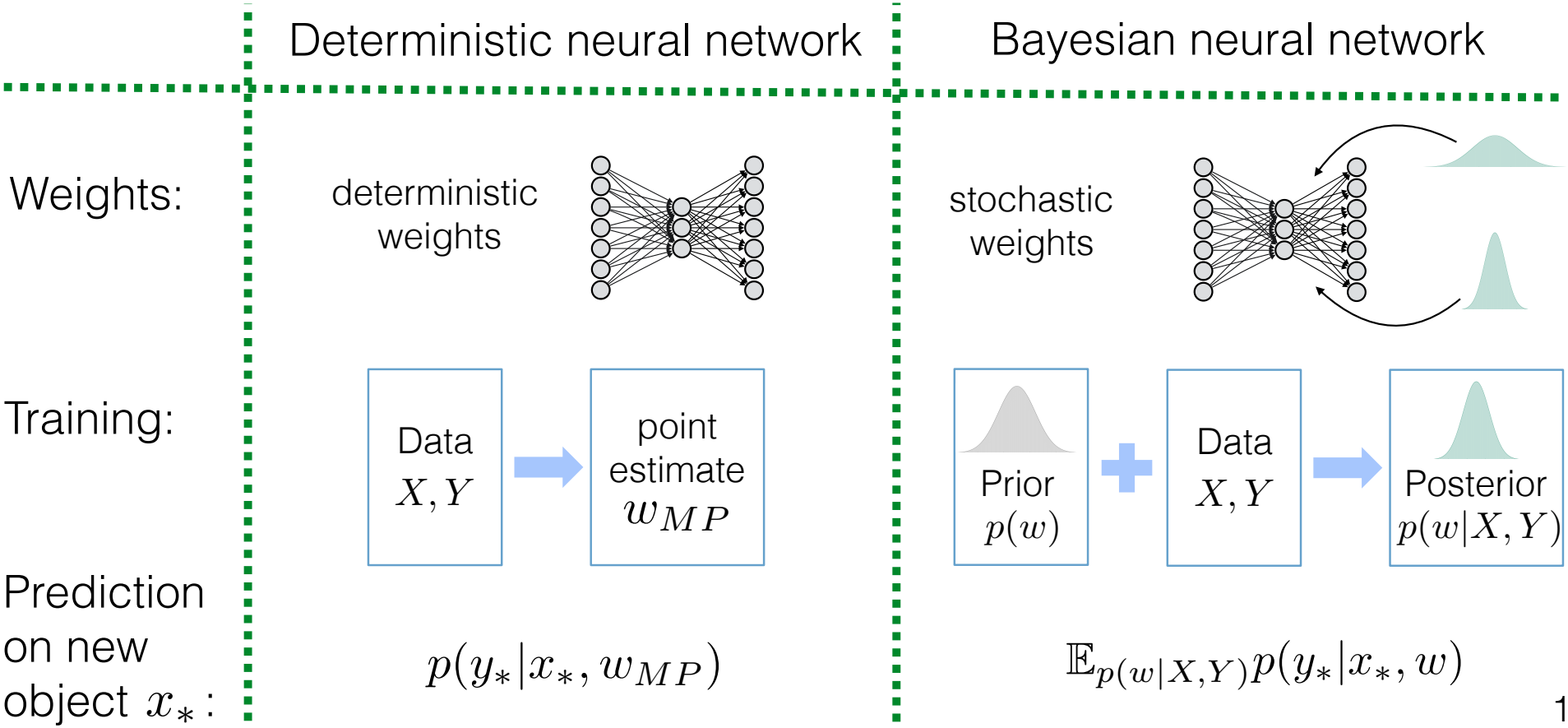
- More recent approaches: regularization by noise
  - Data augmentation, dropout, gradient noise

$$Objective = \mathbb{E}_{p(\Omega)} DataLoss(X, Y, w, \Omega)$$

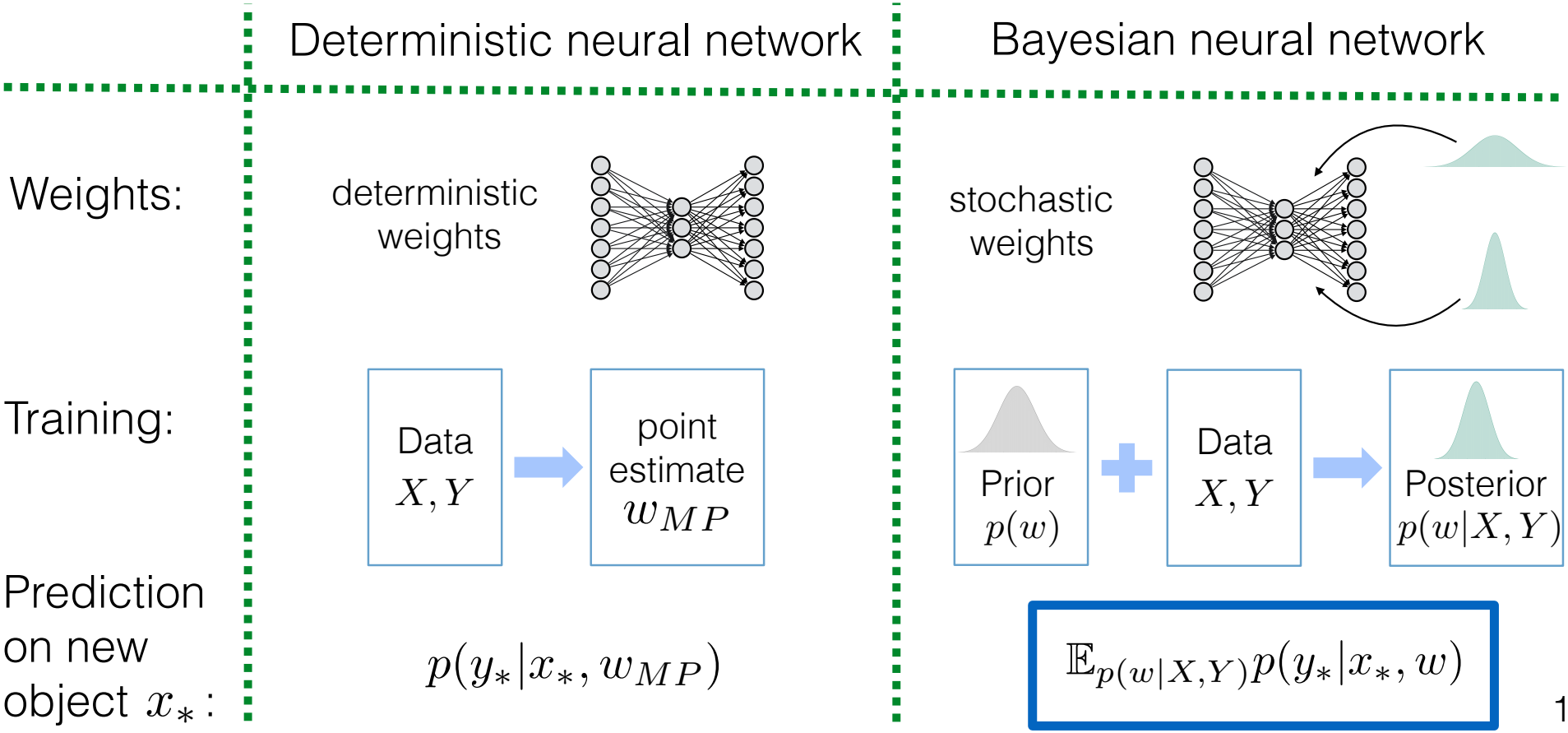
Bayesian framework provides a principled approach to training with noise!

Slide From Dmitry Molchanov's presentation, deepbayes.ru/2018

Bayesian neural networks



Bayesian neural networks



BNN as an ensemble of neural networks

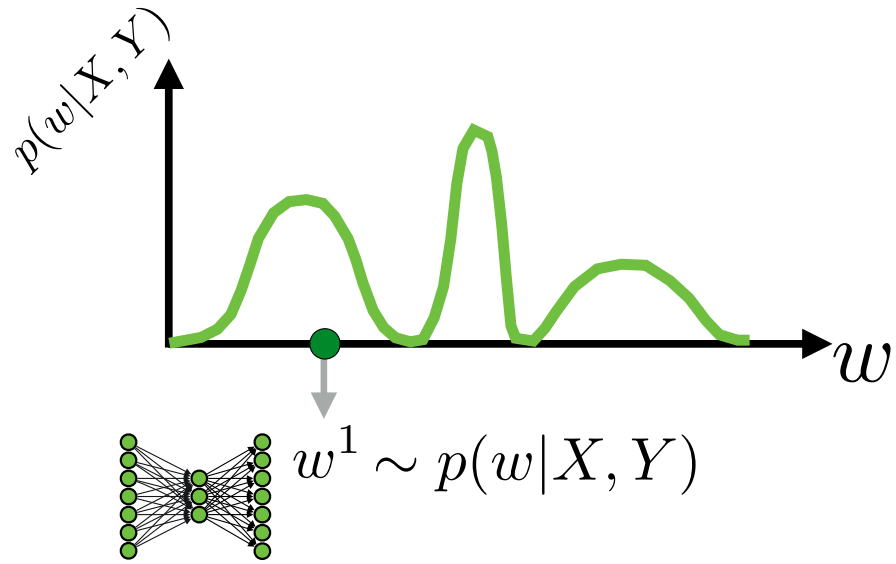
Prediction on a new object x_*:

$$\mathbb{E}_{p(w|X,Y)} p(y_*|x_*, w) \approx \frac{1}{K} \sum_{k=1}^{K} p(y_*|x_*, w^k), \quad w^k \sim p(w|X, Y)$$

BNN as an ensemble of neural networks

Prediction on a new object x_*:

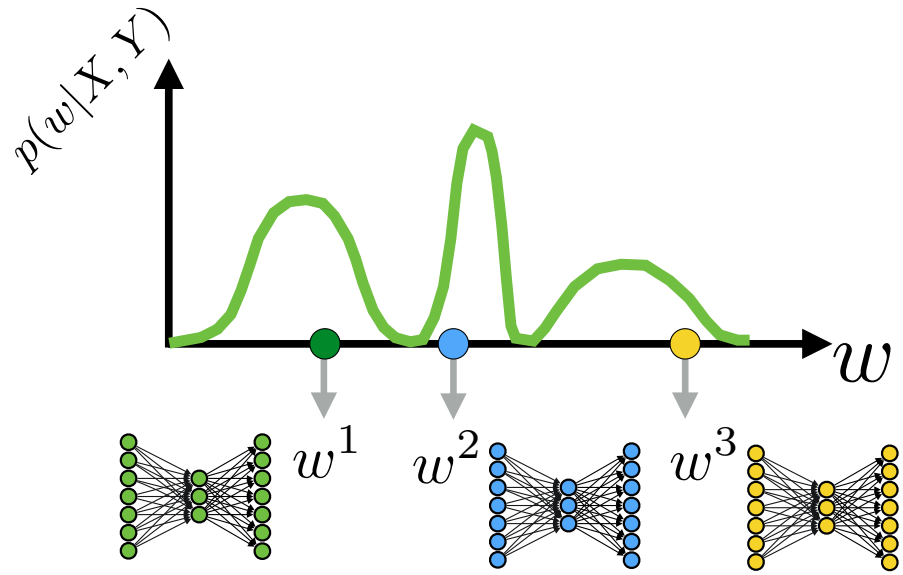
$$\mathbb{E}_{p(w|X,Y)} p(y_*|x_*, w) \approx \frac{1}{K} \sum_{k=1}^{K} p(y_*|x_*, w^k), \quad w^k \sim p(w|X, Y)$$



BNN as an ensemble of neural networks

Prediction on a new object x_*:

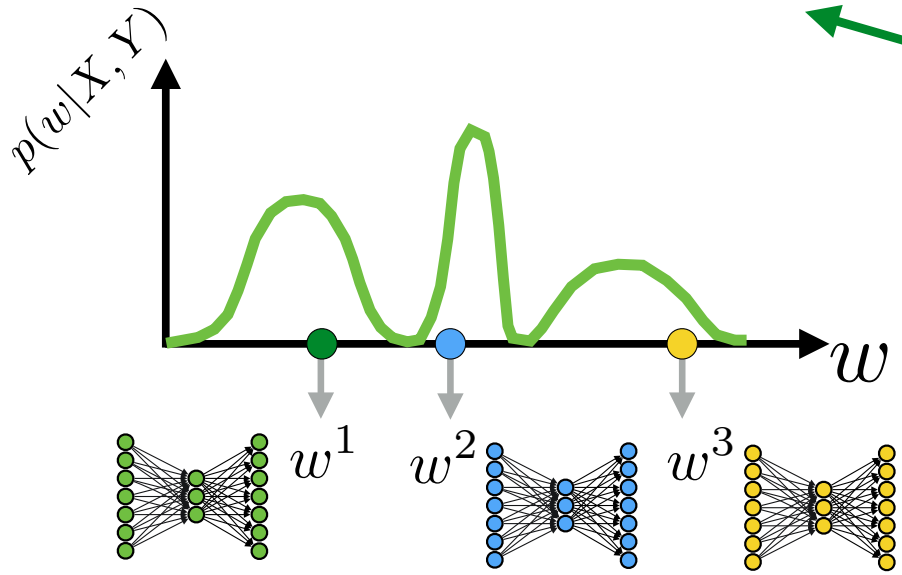
$$\mathbb{E}_{p(w|X,Y)} p(y_*|x_*, w) \approx \frac{1}{K} \sum_{k=1}^{K} p(y_*|x_*, w^k), \quad w^k \sim p(w|X, Y)$$



BNN as an ensemble of neural networks

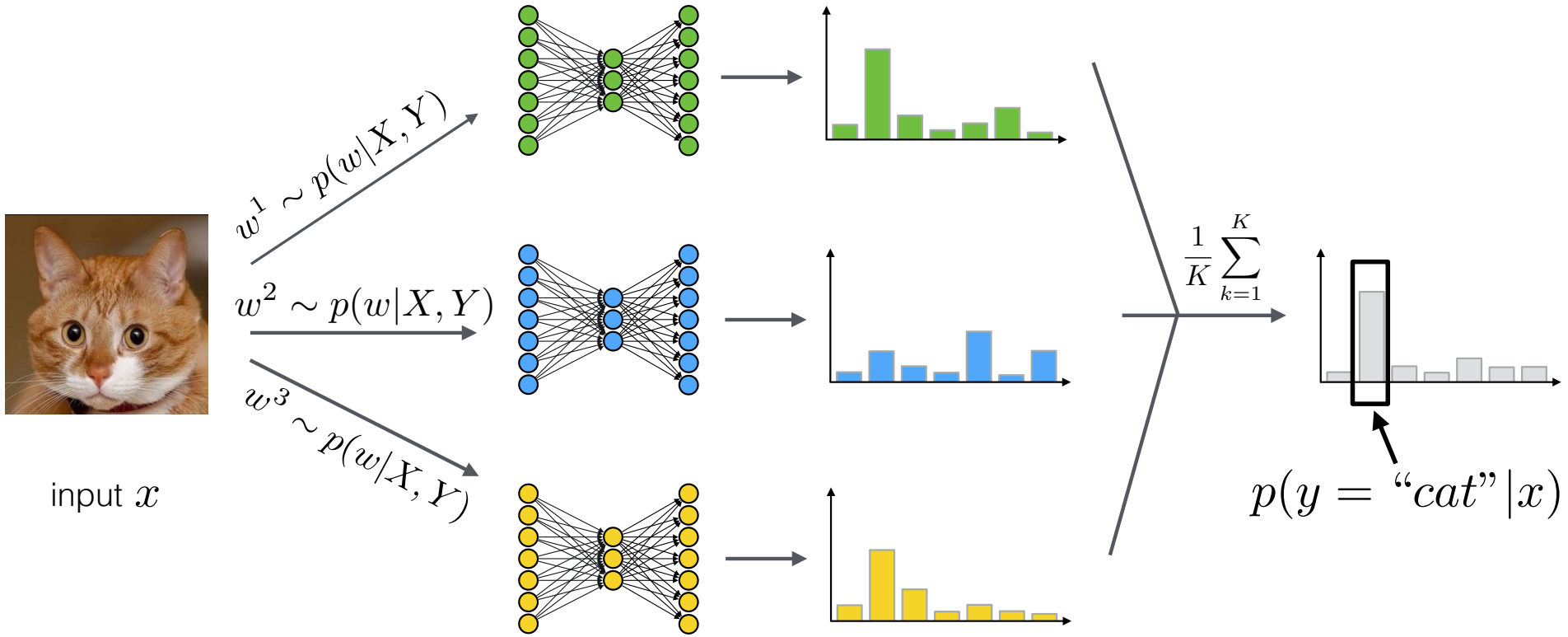
Prediction on a new object x_*:

$$\mathbb{E}_{p(w|X,Y)} p(y_*|x_*, w) \approx \frac{1}{K} \sum_{k=1}^{K} p(y_*|x_*, w^k), \quad w^k \sim p(w|X, Y)$$



average net's output across several weight samples

BNN as an ensemble of neural networks



BNN as an ensemble of neural networks

Prediction on a new object x_*:

$$\mathbb{E}_{p(w|X,Y)} p(y_*|x_*, w) \approx \frac{1}{K} \sum_{k=1}^{K} p(y_*|x_*, w^k), \quad w^k \sim p(w|X, Y)$$

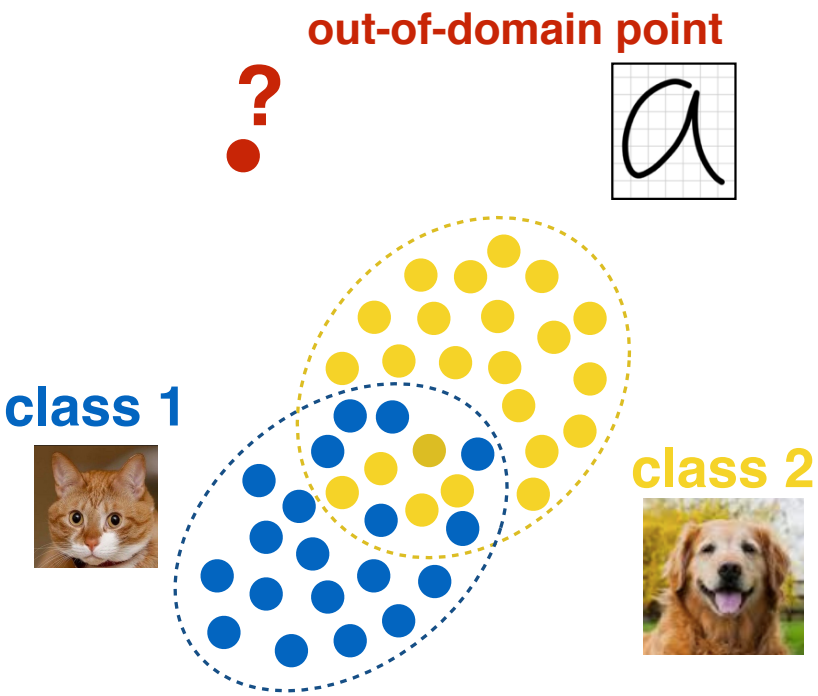
- Higher quality (models compensate each other's errors)
- Better uncertainty estimation

Why go Bayesian?

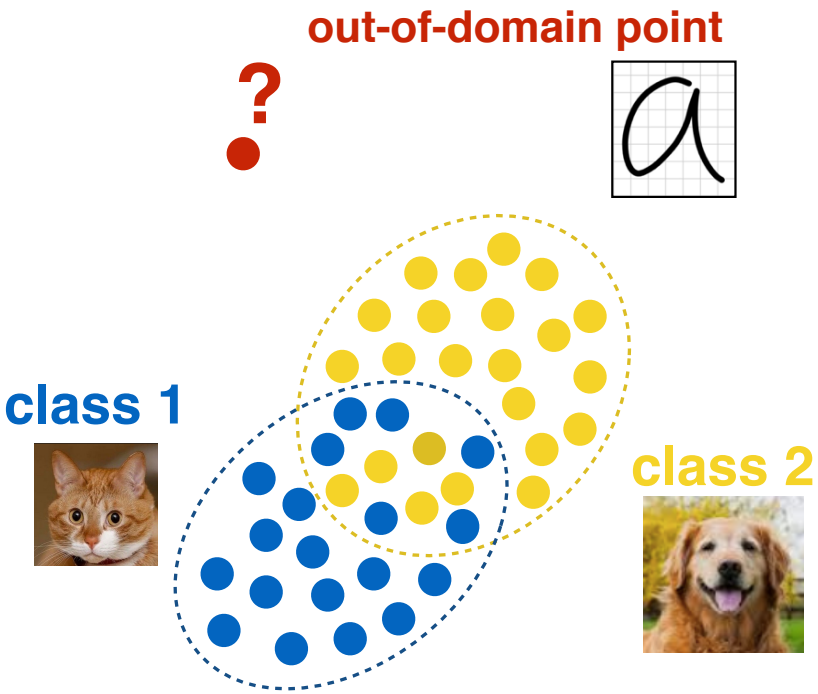
A principled framework with many useful applications

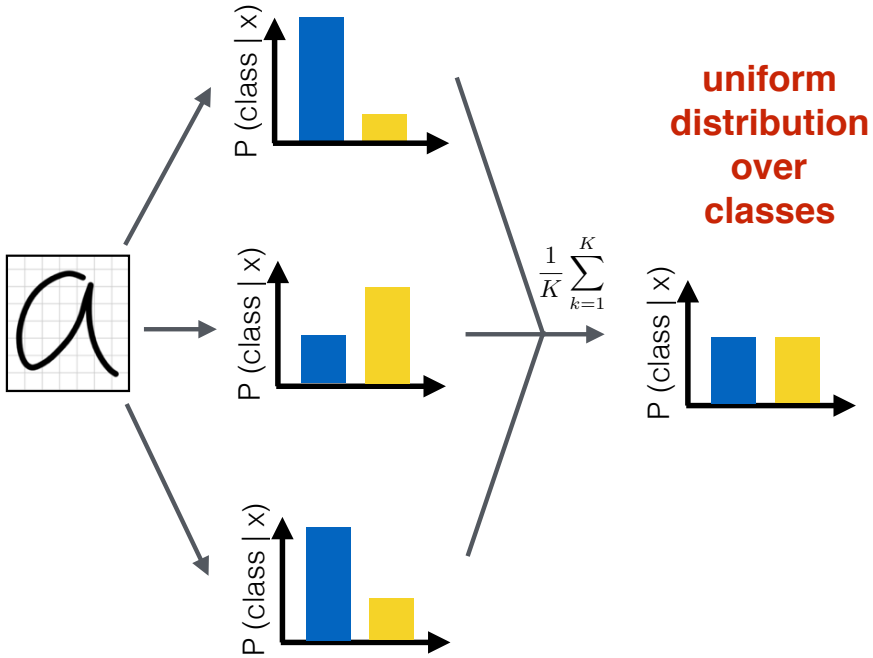
- Regularization
- Ensembling
- Uncertainty estimation
- On-line / continual learning
- Different priors lead to different properties of the network
- Automatic hyperparameter choice

Uncertainty estimation

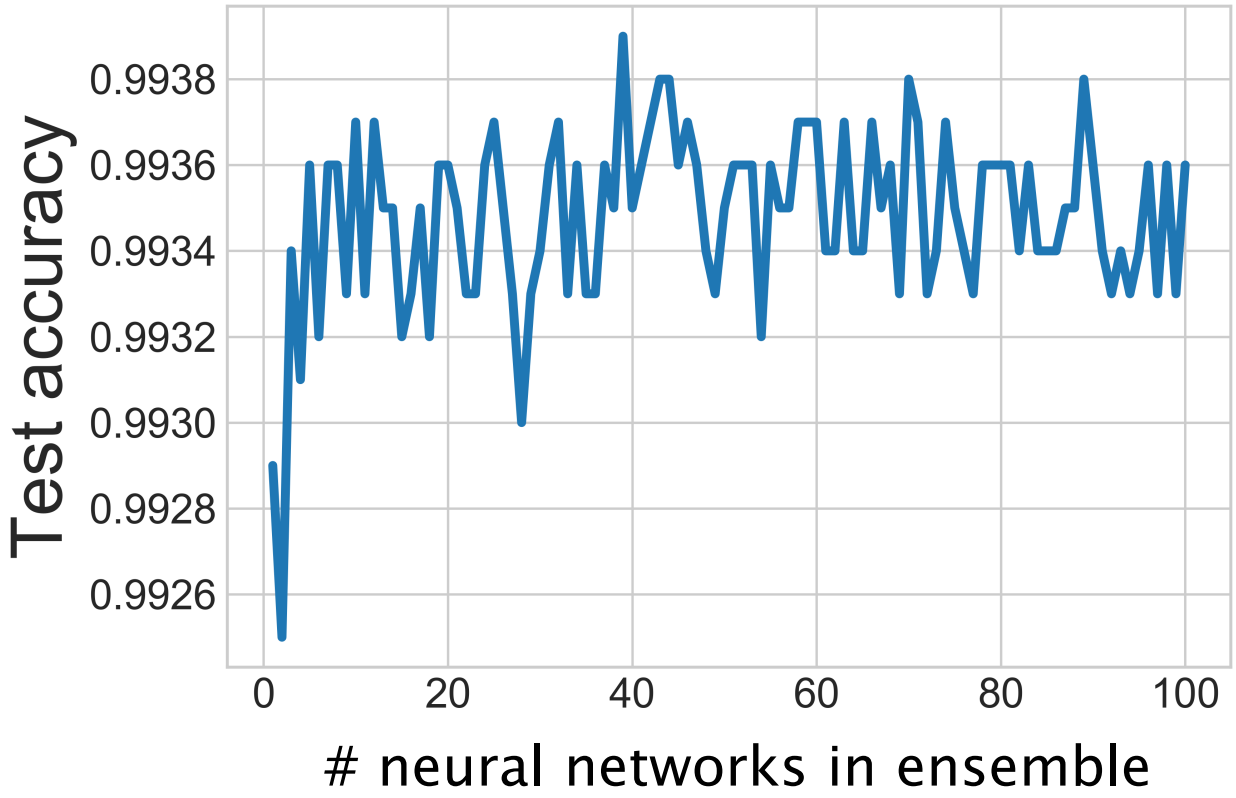


Uncertainty estimation





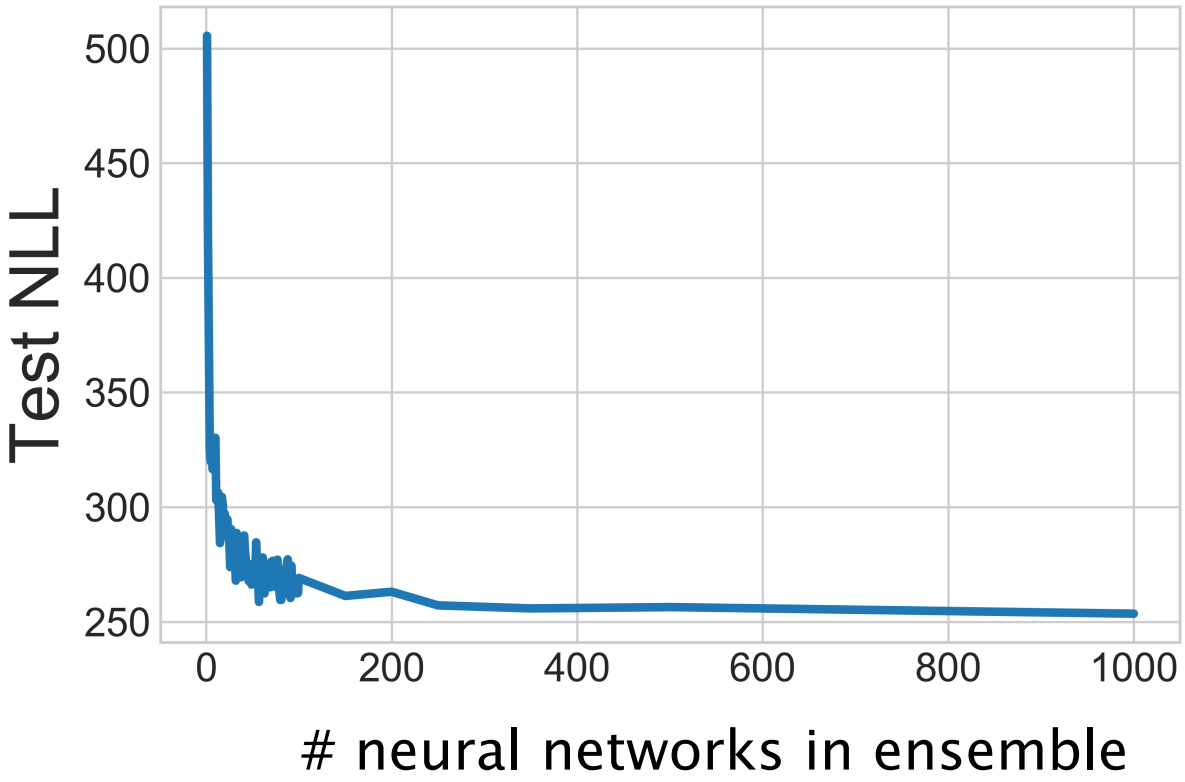
Ensembling: accuracy



Accuracy quickly saturates

Slide From Dmitry Molchanov's presentation, deepbayes.ru/2018

Ensembling: uncertainty estimation

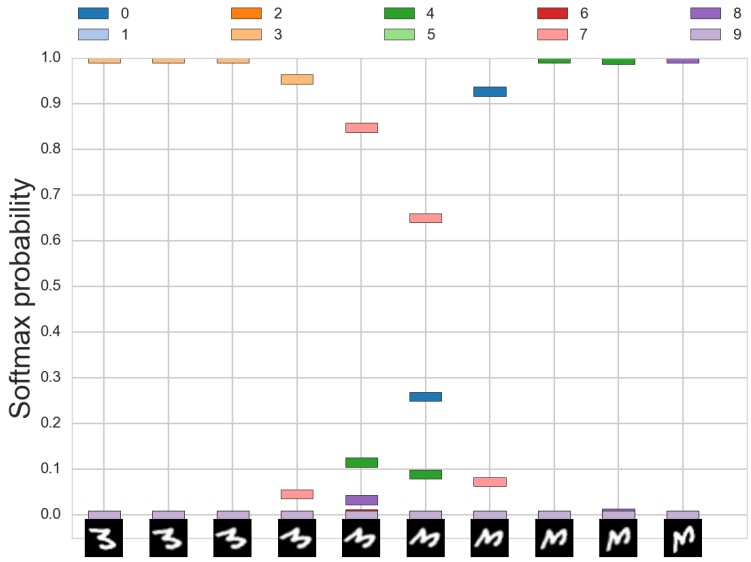


But the negative log—likelihood keeps improving! This is a measure of "**uncertainty**"

Slide From Dmitry Molchanov's presentation, deepbayes.ru/2018

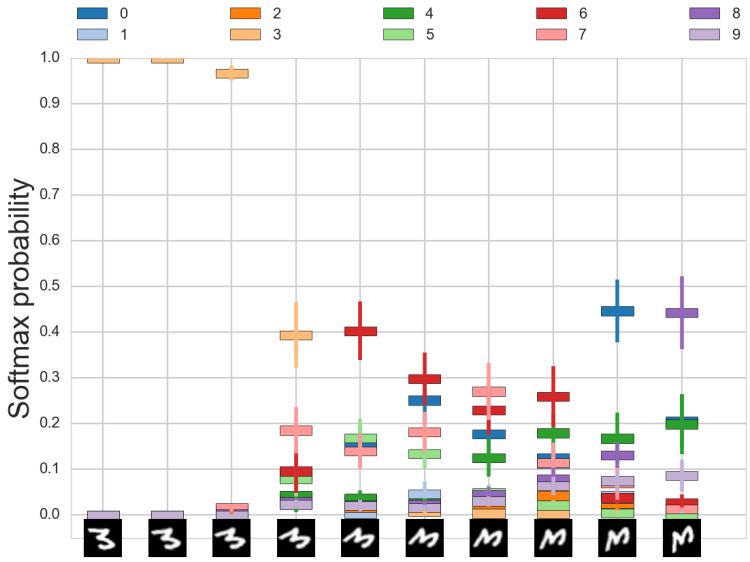
Uncertainty in classification: experiment

Deterministic NN



(a) LeNet with weight decay

Bayesian NN



(b) LeNet with multiplicative formalizing flows

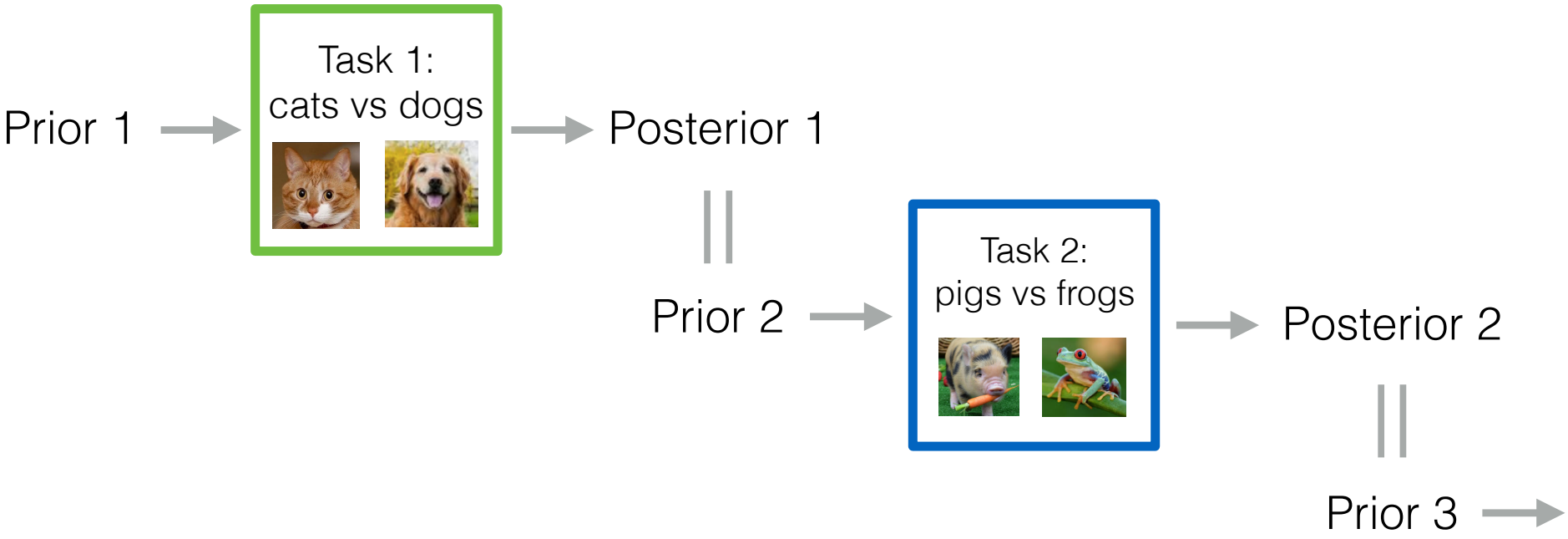
Louizos, Christos, and Max Welling. Multiplicative normalizing flows for variational Bayesian neural networks. ICML 2017

Why go Bayesian?

A principled framework with many useful applications

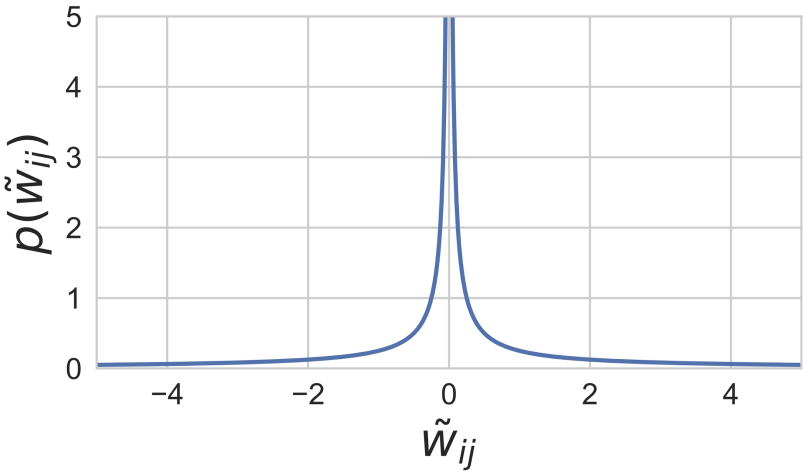
- Regularization ✓
- Ensembling ✓
- Uncertainty estimation ✓
- On-line / continual learning
- Different priors lead to different properties of the network
- Automatic hyperparameter choice

On-line / continual learning

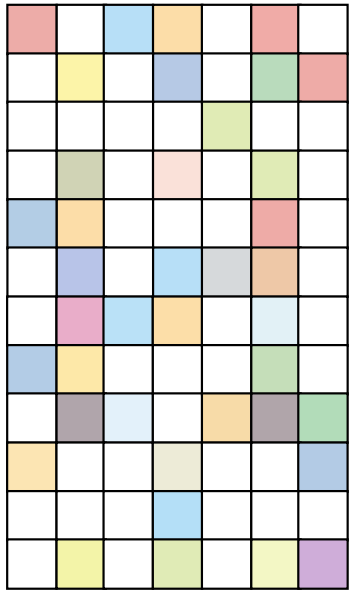


Prior can encode our desirable model properties

Prior concentrated at zero



A lot of zero weights



Weight matrix W

Why go Bayesian?

A principled framework with many useful applications

- Regularization ✓
- Ensembling ✓
- Uncertainty estimation ✓
- On-line / continual learning ✓
- Different priors lead to different properties of the network ✓
- Automatic hyperparameter choice

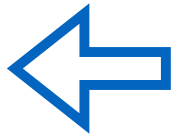
Plan

- Advantages of using Bayesian neural networks
- Training Bayesian neural networks
- Q&A + exercises

Training methods: summary

Probabilistic model: $p(Y|X,w)p(w)$ We want to compute: $p(w|X,Y)$

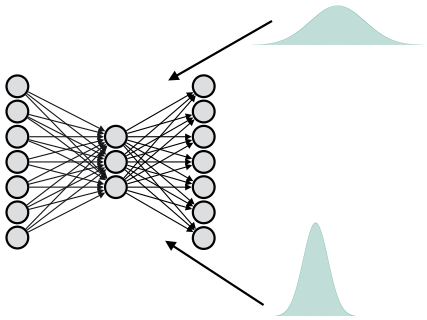
Approximation		Inference
Exact	$p(w\|X,Y)$	Full Bayesian inference
Parametric	$p(w\|X,Y) \approx q(w\|\lambda)$	Parametric Var. Inference
Delta function	$p(w\|X,Y) \approx \delta(w_{MP})$	Max. posterior inference
No prior	w_{ML}	Max. likelihood inference



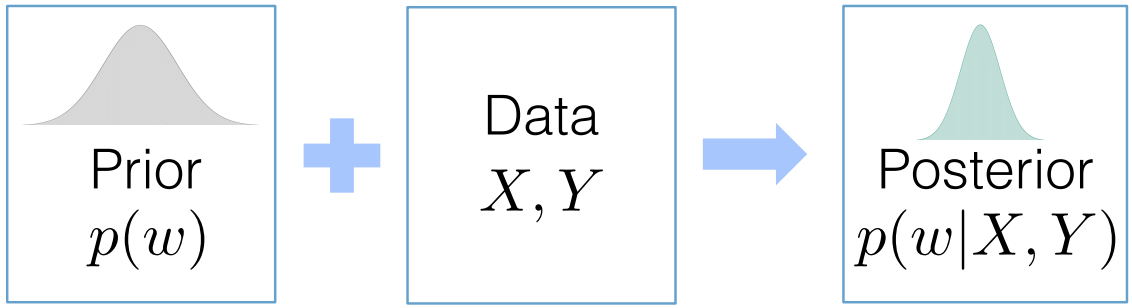
Modified slide from Ekaterina Lobacheva's presentation

Training Bayesian neural networks

Stochastic weights:



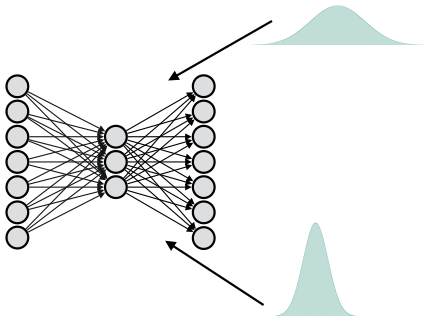
Bayesian Inference:



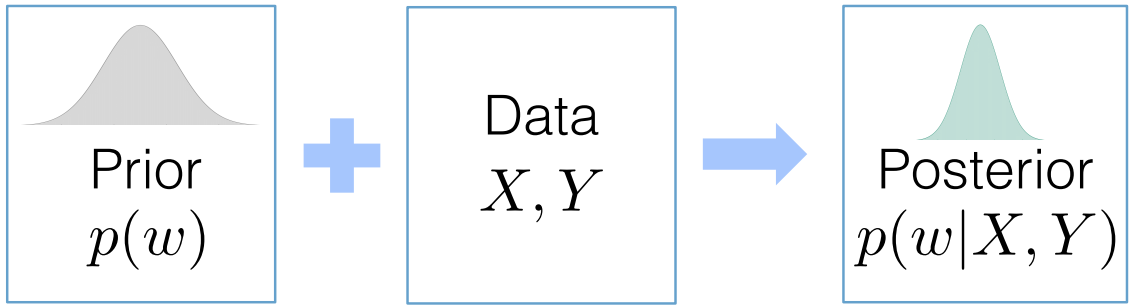
$$p(w|X,Y) = \frac{p(Y|X,w)p(w)}{\int p(Y|X,\tilde{w})p(\tilde{w})d\tilde{w}}$$

Training Bayesian neural networks

Stochastic weights:



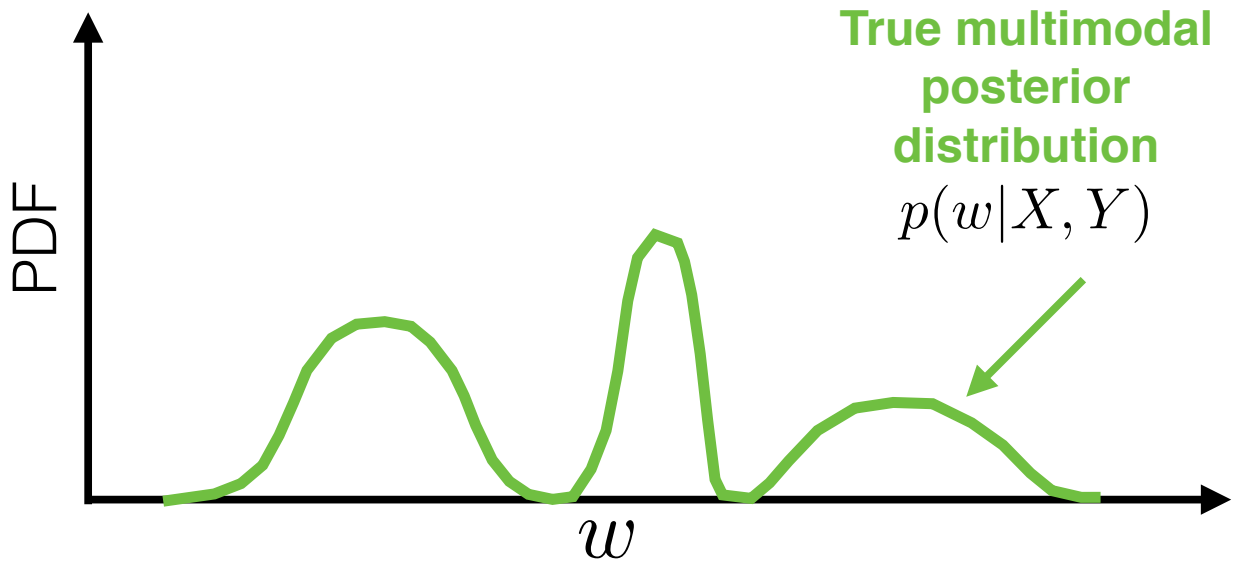
Bayesian Inference:



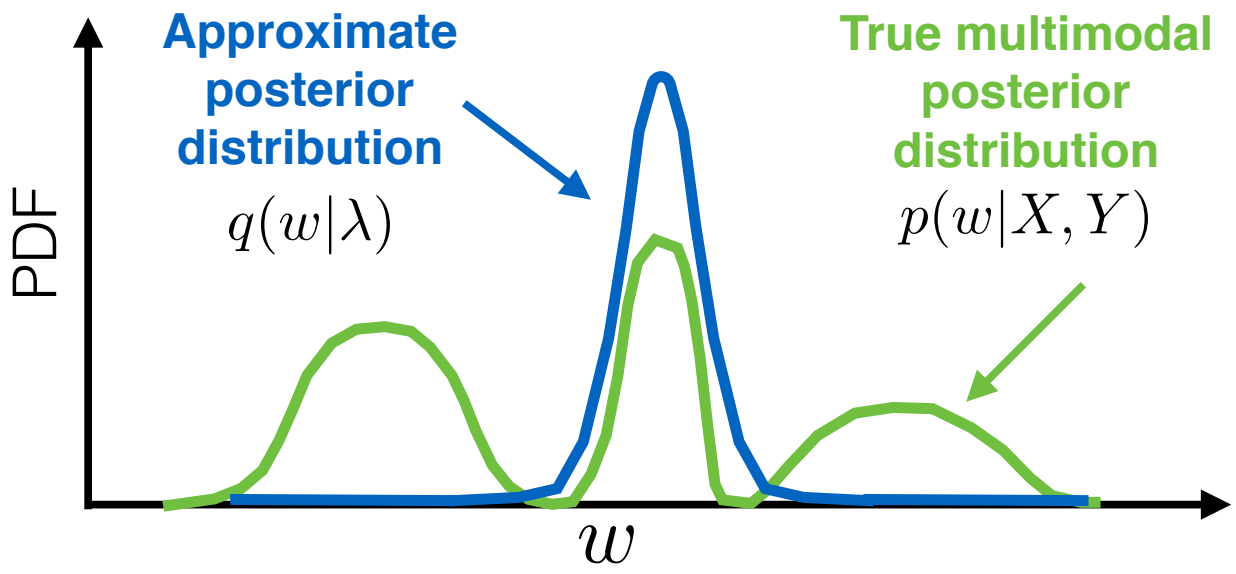
$$p(w|X,Y) = \frac{p(Y|X,w)p(w)}{\int p(Y|X,\tilde{w})p(\tilde{w})d\tilde{w}}$$

Intractable!

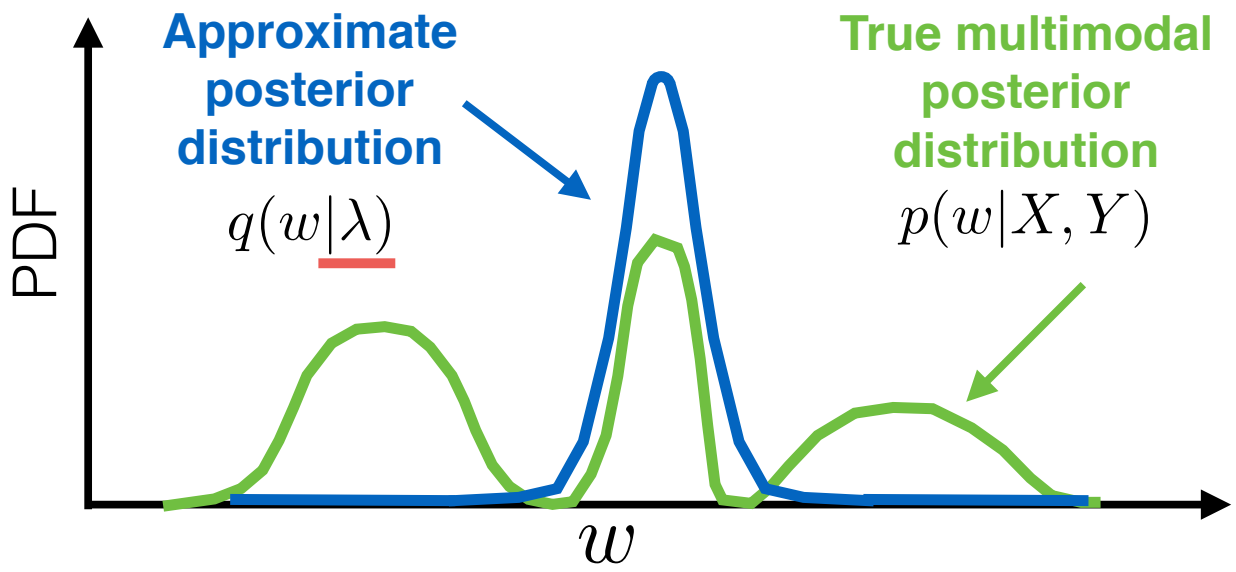
Training Bayesian neural networks



Training Bayesian neural networks



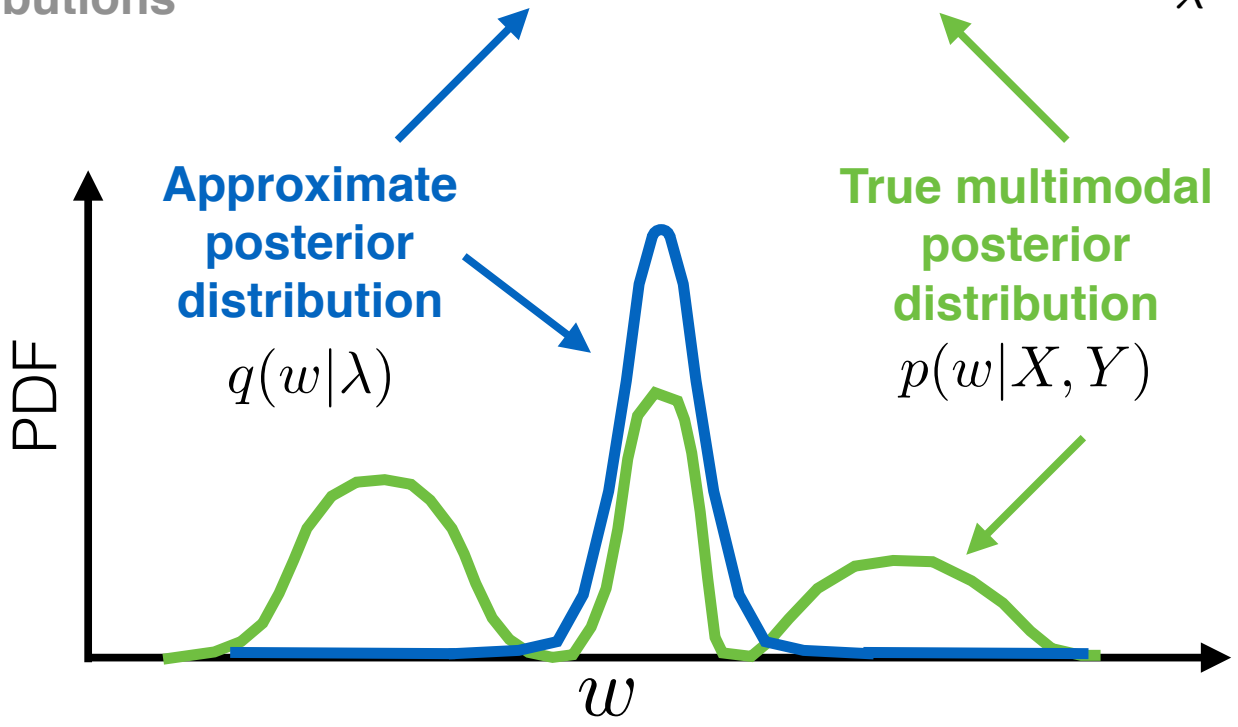
Training Bayesian neural networks



Training Bayesian neural networks

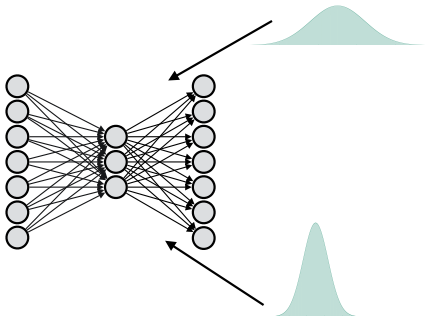
"distance" between two distributions

$$KL(q(w|\lambda)||p(w|X,Y)) \rightarrow \min_{\lambda}$$

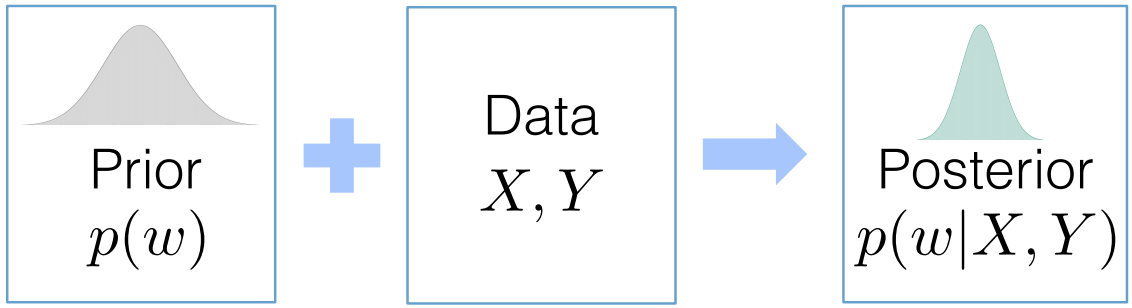


Training Bayesian neural networks

Stochastic weights:



Bayesian Inference:

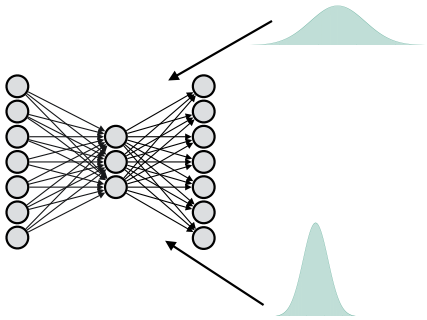


Posterior is intractable in neural networks → approximate it with $q(w|\lambda)$:

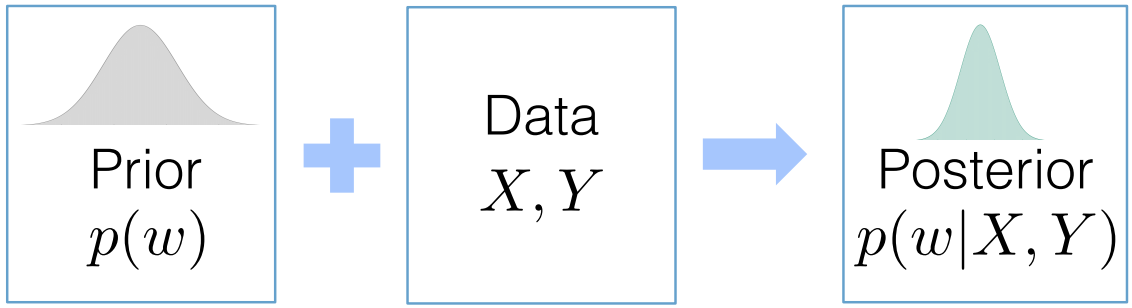
$$KL(q(w|\lambda)||p(w|X,Y)) \rightarrow \min_{\lambda}$$

Training Bayesian neural networks

Stochastic weights:



Bayesian Inference:



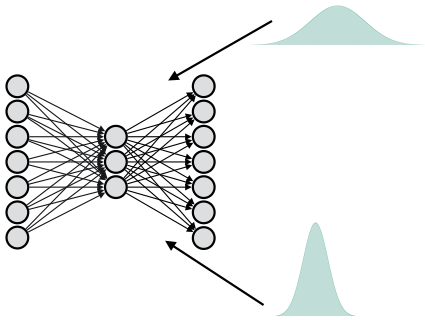
Posterior is intractable in neural networks → approximate it with $q(w|\lambda)$:

$$KL(q(w|\lambda)||p(w|X,Y)) \rightarrow \min_{\lambda}$$

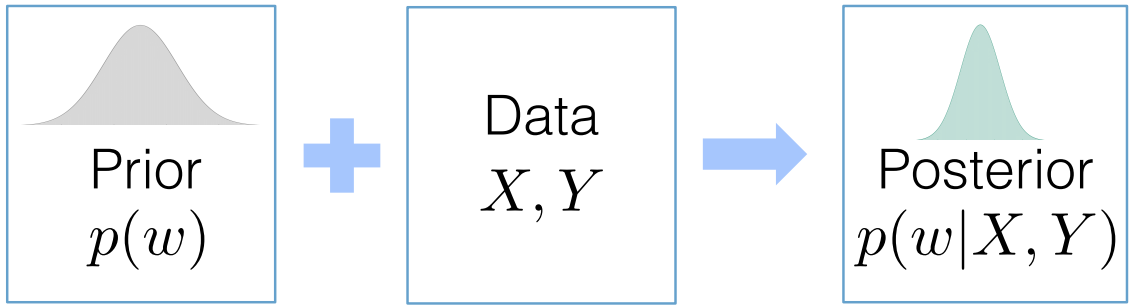
Intractable!

Training Bayesian neural networks

Stochastic weights:



Bayesian Inference:



Equivalently, we can optimize ELBO to find approximate posterior $q(w|\lambda)$:

$$\sum_{i=1}^{N} \underbrace{\mathbb{E}_{q(w|\lambda)} \log p(y^i|x^i, w)}_{\text{Data term}} - \underbrace{KL(q(w|\lambda)||p(w))}_{\text{Regularizer}} \to \max_{\lambda}$$

Doubly stochastic variational inference

How to estimate the data term?

$$\sum_{i=1}^{N} \underbrace{\mathbb{E}_{q(w|\lambda)} \log p(y^i|x^i, w)}_{\text{Data term}} - \underbrace{KL(q(w|\lambda)||p(w))}_{\text{Regularizer}} \to \max_{\lambda}$$

???

Doubly stochastic variational inference

How to estimate the data term?

$$\sum_{i=1}^{N} \underbrace{\mathbb{E}_{q(w|\lambda)} \log p(y^i|x^i, w)}_{\text{Data term}} - \underbrace{KL(q(w|\lambda)||p(w))}_{\text{Regularizer}} \to \max_{\lambda}$$

sample mini-batch of the training objects

sample weights from q

Doubly stochastic variational inference

How to estimate the data term?

$$\sum_{i=1}^{N} \underbrace{\mathbb{E}_{q(w|\lambda)} \log p(y^i|x^i, w)}_{\text{Data term}} - \underbrace{KL(q(w|\lambda)||p(w))}_{\text{Regularizer}} \to \max_{\lambda}$$

sample mini-batch of the training objects

sample weights from q

$$\sum_{j=1}^{m} \log p(y^{i_j}|x^{i_j}, \hat{w}_j), \qquad \hat{w}_j \sim q(w|\lambda) \qquad i_j \sim \text{Unif}(1, \ldots, N)$$

Doubly stochastic variational inference

How to estimate the data term?

$$\sum_{i=1}^{N} \mathbb{E}_{q(w|\lambda)} \log p(y^i|x^i, w) - KL(q(w|\lambda)||p(w)) \to \max_{\lambda}$$

sample mini-batch of the training objects

sample weights from q

$$\sum_{j=1}^{m} \log p(y^{i_j}|x^{i_j}, \hat{w}_j), \qquad \hat{w}_j \sim q(w|\lambda) \qquad i_j \sim \text{Unif}(1, \ldots, N)$$

λ? **How to differentiate?**

Doubly stochastic variational inference

How to estimate the data term?

$$\sum_{i=1}^{N} \mathbb{E}_{q(w|\lambda)} \log p(y^i|x^i, w) - KL(q(w|\lambda)||p(w)) \to \max_{\lambda}$$

$$\frac{\partial}{\partial \lambda}?$$

sample mini-batch of the training objects

sample weights from q

$$\sum_{j=1}^{m} \log p(y^{i_j}|x^{i_j}, \hat{w}_j), \qquad \hat{w}_j \sim q(w|\lambda) \qquad i_j \sim \text{Unif}(1, \ldots, N)$$

λ? **How to differentiate?**

Gradient of expectation

$$\frac{\partial}{\partial \lambda} \mathbb{E}_{q(w|\lambda)} \log p(y^i|x^i, w) = \frac{\partial}{\partial \lambda} \int q(w|\lambda) \log p(y^i|x^i, w) dw =$$

$$= \int \left(\frac{\partial}{\partial \lambda} q(w|\lambda) \right) \log p(y^i|x^i, w) dw$$

Not an expectation over q anymore!

Most popular solution: **reparametrization trick**

Reparametrization trick

As an example, consider 1-dim normal approximate posterior:

$$q(w|\lambda) = \mathcal{N}(\mu, \sigma^2), \quad \lambda = \{\mu, \sigma\}$$

Reparametrization trick

As an example, consider 1-dim normal approximate posterior:

$$q(w|\lambda) = \mathcal{N}(\mu, \sigma^2), \quad \lambda = \{\mu, \sigma\}$$

Property of normal distribution:

$$w \sim \mathcal{N}(\mu, \sigma^2) \quad \Leftrightarrow \quad w = \mu + \sigma\epsilon, \;\; \epsilon \sim \mathcal{N}(0, 1)$$

Reparametrization trick

As an example, consider 1-dim normal approximate posterior:

$$q(w|\lambda) = \mathcal{N}(\mu, \sigma^2), \quad \lambda = \{\mu, \sigma\}$$

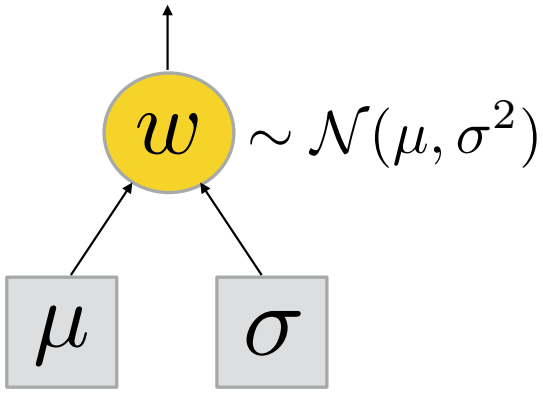
Property of normal distribution:

$$w \sim \underbrace{\mathcal{N}(\mu, \sigma^2)}_{\text{distribution conditioned on parameters}} \quad \Leftrightarrow \quad w = \mu + \sigma\epsilon, \;\; \epsilon \sim \underbrace{\mathcal{N}(0, 1)}_{\text{unconditioned distribution}}$$

Diederik P Kingma, Max Welling . Auto-Encoding Variational Bayes. ICLR 2013
Image inspired by https://jaan.io/what-is-variational-autoencoder-vae-tutorial/

Reparametrization trick

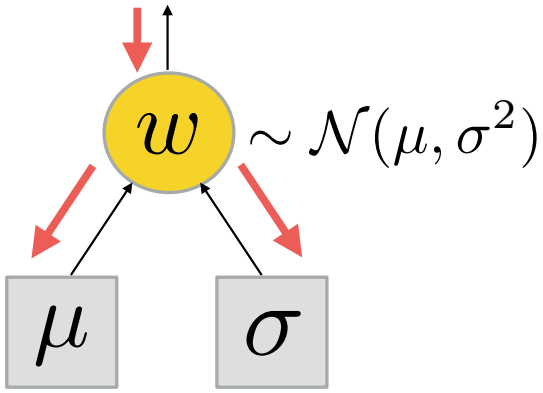
$$w \sim \mathcal{N}(\mu, \sigma^2) \quad \Leftrightarrow \quad w = \mu + \sigma\epsilon, \;\; \epsilon \sim \mathcal{N}(0, 1)$$



Diederik P Kingma, Max Welling . Auto-Encoding Variational Bayes. ICLR 2013
Image inspired by https://jaan.io/what-is-variational-autoencoder-vae-tutorial/

Reparametrization trick

$$w \sim \mathcal{N}(\mu, \sigma^2) \quad \Leftrightarrow \quad w = \mu + \sigma\epsilon, \;\; \epsilon \sim \mathcal{N}(0, 1)$$

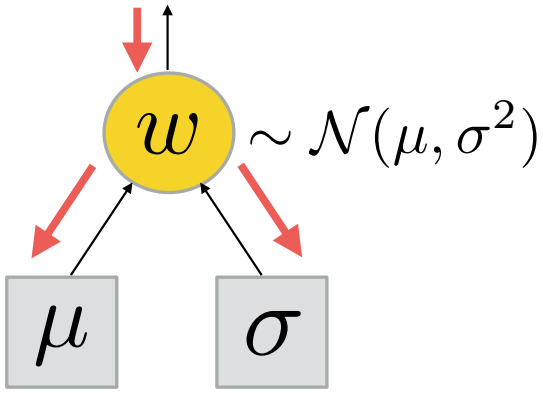


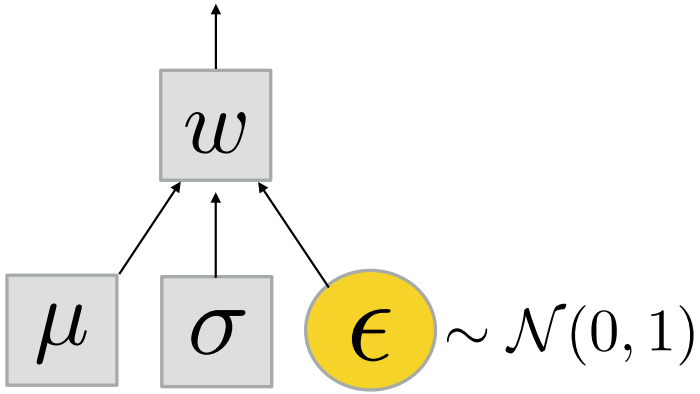
Gradients propagate through randomness

Diederik P Kingma, Max Welling . Auto-Encoding Variational Bayes. ICLR 2013
Image inspired by https://jaan.io/what-is-variational-autoencoder-vae-tutorial/

Reparametrization trick

$$w \sim \mathcal{N}(\mu, \sigma^2) \quad \Leftrightarrow \quad w = \mu + \sigma\epsilon, \;\; \epsilon \sim \mathcal{N}(0, 1)$$



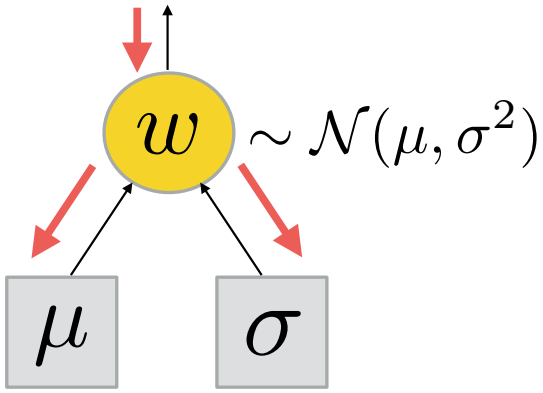


Gradients propagate through randomness

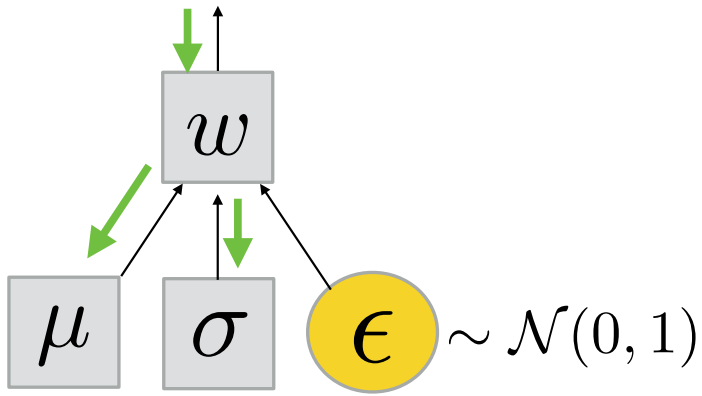
Diederik P Kingma, Max Welling . Auto-Encoding Variational Bayes. ICLR 2013
Image inspired by https://jaan.io/what-is-variational-autoencoder-vae-tutorial/

Reparametrization trick

$$w \sim \mathcal{N}(\mu, \sigma^2) \quad \Leftrightarrow \quad w = \mu + \sigma\epsilon, \;\; \epsilon \sim \mathcal{N}(0, 1)$$



Gradients propagate through randomness



Gradients propagate only through deterministic nodes

Diederik P Kingma, Max Welling . Auto-Encoding Variational Bayes. ICLR 2013
Image inspired by https://jaan.io/what-is-variational-autoencoder-vae-tutorial/

Reparametrization trick: general form

$$w \sim \mathcal{N}(\mu, \sigma^2) \quad \Leftrightarrow \quad w = \mu + \sigma\epsilon, \quad \epsilon \sim \mathcal{N}(0, 1)$$

$$w \sim q(w|\lambda) \quad \Leftrightarrow \quad w = f(\lambda, \epsilon), \quad \epsilon \sim p(\epsilon)$$

distribution conditioned on parameters

dependency on parameters

unconditioned distribution

Reparametrization trick

$$\underbrace{w \sim q(w|\lambda)}_{\text{distribution conditioned on parameters}} \quad \Leftrightarrow \quad \underbrace{w = f(\lambda, \epsilon),}_{\text{dependency on parameters}} \quad \underbrace{\epsilon \sim p(\epsilon)}_{\text{unconditioned distribution}}$$

$$\nabla_\lambda \mathbb{E}_{q(w|\lambda)} \log p(y^i|x^i, w) = \nabla_\lambda \mathbb{E}_{p(\epsilon)} \log p(y^i|x^i, w = f(\lambda, \epsilon)) =$$

$$= \mathbb{E}_{p(\epsilon)} \nabla_\lambda \log p(y^i|x^i, w = f(\lambda, \epsilon))$$

Reparametrization trick: examples

$$q(w|\lambda) \quad \longrightarrow \quad w = f(\lambda, \epsilon), \;\; p(\epsilon)$$

$q(w\|\lambda)$	$p(\epsilon)$	$f(\lambda, \epsilon)$
$\mathcal{N}\left(\mu, \sigma^2\right)$	$\mathcal{N}(0, 1)$	$w = \sigma\epsilon + \mu$
$\mathcal{G}(1, \beta)$	$\mathcal{G}(1, 1)$	$w = \beta\epsilon$
$\mathcal{E}(\lambda)$	$\mathcal{U}(0, 1)$	$w = -\frac{\log \epsilon}{\lambda}$
$\mathcal{N}(\mu, \Sigma)$	$\mathcal{N}(0, I)$	$w = A\epsilon + \mu, \;\; \text{where } AA^T = \Sigma$

Reparametrization trick: examples

$$q(w|\lambda) \quad \longrightarrow \quad w = f(\lambda, \epsilon), \;\; p(\epsilon)$$

$q(w\|\lambda)$	$p(\epsilon)$	$f(\lambda, \epsilon)$
$\mathcal{N}\left(\mu, \sigma^2\right)$	$\mathcal{N}(0,1)$	$w = \sigma\epsilon + \mu$
$\mathcal{G}(1, \beta)$	$\mathcal{G}(1,1)$	$w = \beta\epsilon$
$\mathcal{E}(\lambda)$	$\mathcal{U}(0,1)$	$w = -\frac{\log \epsilon}{\lambda}$
$\mathcal{N}(\mu, \Sigma)$	$\mathcal{N}(0, I)$	$w = A\epsilon + \mu, \;\; \text{where } AA^T = \Sigma$

noise on weights!

Training: putting everything together

How to estimate the data term?

$$\sum_{i=1}^{N} \underbrace{\mathbb{E}_{q(w|\lambda)} \log p(y^i|x^i, w)}_{\text{Data term}} - \underbrace{KL(q(w|\lambda)||p(w))}_{\text{Regularizer}} \to \max_{\lambda}$$

sample mini-batch of the training objects

sample weights from q using reparametrization trick

$$\sum_{j=1}^{m} \log p(y^{i_j}|x^{i_j},\, w = f(\lambda, \epsilon^j)), \quad \epsilon^j \sim p(\epsilon) \quad i_j \sim \text{Unif}(1, \ldots, N)$$

Training: putting everything together

Deterministic neural network:

$$w^{new} = w^{old} + \eta \frac{\partial}{\partial w} \sum_{j=1}^{m} \log p(y^{i_j} | x^{i_j}, w^{old}) \qquad i_j \sim \text{Unif}(1, \ldots, N)$$

Bayesian neural network:

$$\lambda^{new} = \lambda^{old} + \eta \frac{\partial}{\partial \lambda} \sum_{j=1}^{m} \log p(y^{i_j} | x^{i_j}, w = f(\lambda^{old}, \epsilon^j)), \qquad i_j \sim \text{Unif}(1, \ldots, N)$$

$$\epsilon^j \sim p(\epsilon)$$

m — mini-batch size η — learning rate

From general framework to particular method

$$\sum_{i=1}^{N} \mathbb{E}_{q(w|\lambda)} \log p(y^i|x^i, w) - KL(q(w|\lambda)||p(w)) \to \max_{\lambda}$$

Model specification:

- Choose particular prior

Training:

- Choose particular family for approximate posterior
- How to compute the KL-divergence?

Software

- Pyro (based on PyTorch)
- TensorFlow Probability (based on TensorFlow)
- Edward (based on TensorFlow)
- PyMC (based on Theano, pre-release with TensorFlow Probability)
- https://github.com/JavierAntoran/Bayesian-Neural-Networks — PyTorch implementations of popular Bayesian deep learning papers

Summary

- A lot of BNN advantages: regularization, ensembling, uncertainty estimation, …
- To train BNN, one should optimize ELBO using DSVI & RT
- Three steps towards a particular method

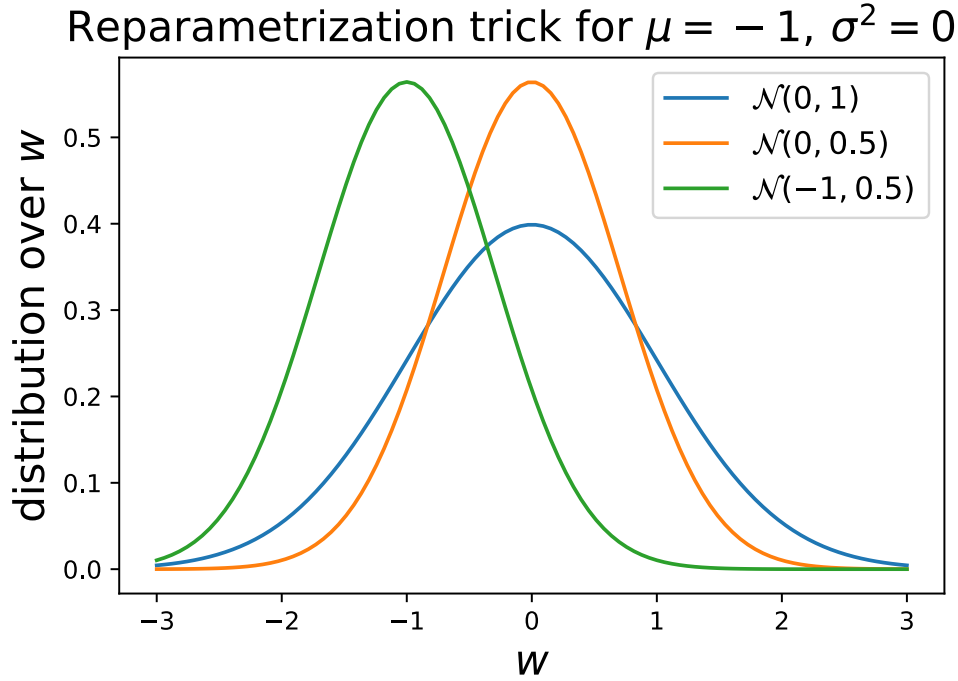
Plan

- Advantages of using Bayesian neural networks
- Training Bayesian neural networks
- Q&A + exercises

Questions

Reparametrization trick: recup

$$w \sim \mathcal{N}(\mu, \sigma^2) \quad \Leftrightarrow \quad w = \mu + \sigma\epsilon, \quad \epsilon \sim \mathcal{N}(0, 1)$$
$$w \sim q(w|\lambda) \quad \Leftrightarrow \quad w = f(\lambda, \epsilon), \quad \epsilon \sim p(\epsilon)$$

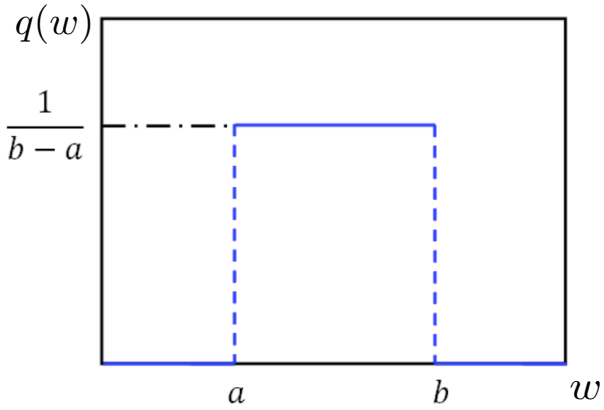


Exercises

Reparametrization trick: $w \sim q(w|\lambda) \quad \Leftrightarrow \quad w = f(\lambda, \epsilon), \qquad \epsilon \sim p(\epsilon)$

- How can we reparametrize uniform distribution?

$$q(w|a,b) = \begin{cases} \frac{1}{b-a}, \; w \in [a,b] \\ 0, \; \text{otherwise} \end{cases}$$



- How can we reparametrize a mixture of normal distributions?

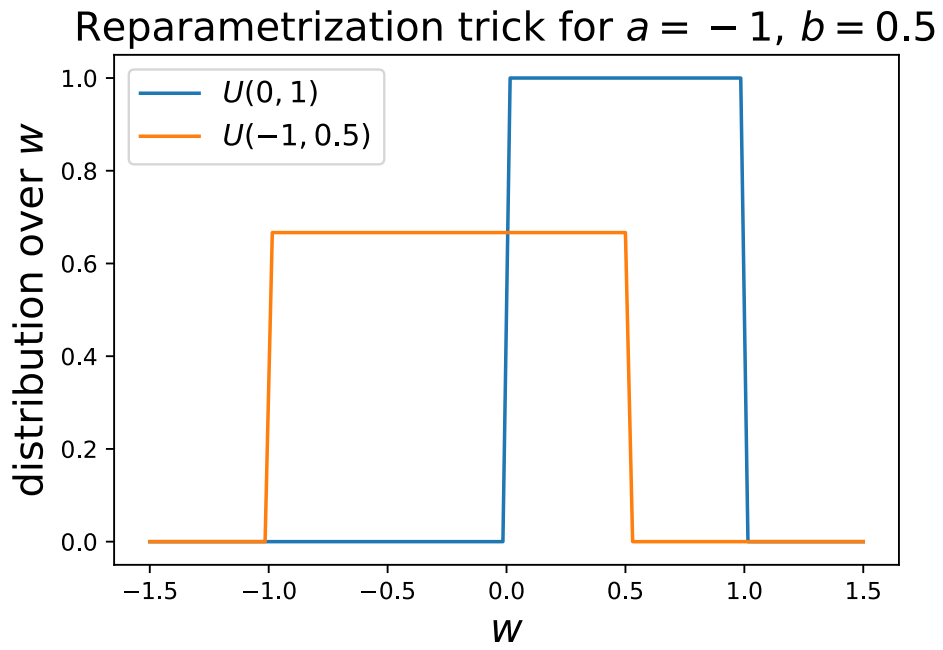
$$q(w|\mu,\sigma) = \sum_{s=1}^{S} \pi_s \mathcal{N}(w|\mu_s,\sigma_s), \quad \sum_{s=1}^{S} \pi_s = 1, \; \pi_s \geqslant 0 \qquad w \in \mathbb{R}$$

(π is fixed)

Reparametrizing uniform distribution

$$q(w|a,b) = \begin{cases} \frac{1}{b-a}, \ w \in [a,b] \\ 0, \ \text{otherwise} \end{cases}$$

$$w \sim q(w|a,b) \Leftrightarrow$$
$$w = (1-t)a + tb, \ t \sim U(0,1)$$



Reparametrizing a mixture of normal distributions

$$q(w|\mu,\sigma) = \sum_{s=1}^{S} \pi_s \mathcal{N}(w|\mu_s,\sigma_s), \quad \sum_{s=1}^{S} \pi_s = 1,\ \pi_s \geqslant 0$$

$$\hat{w} \sim q(w|\mu,\sigma) \quad \Leftrightarrow \quad \hat{w} = f(\mu,\sigma,\hat{\epsilon},\hat{z}) = \mu_{\hat{z}} + \hat{\epsilon}\sigma_{\hat{z}}$$
$$\hat{\epsilon} \sim \mathcal{N}(\epsilon|0,1), \quad \hat{z} \sim Cat(\pi_1,\ldots,\pi_S)$$