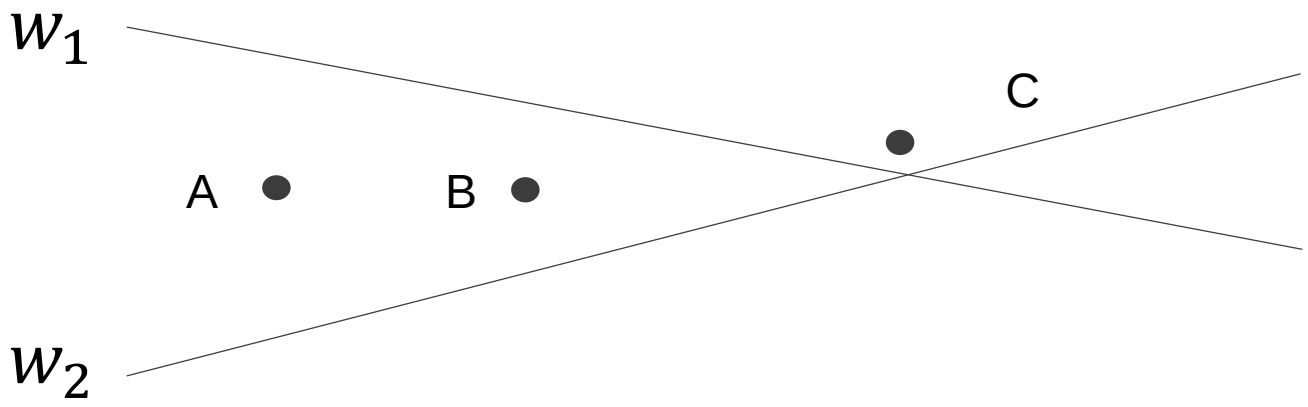
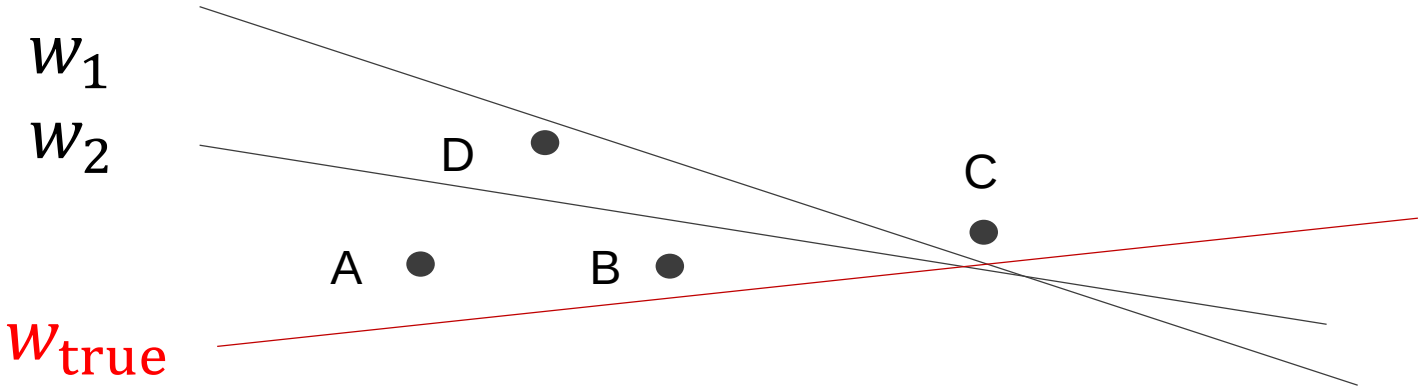
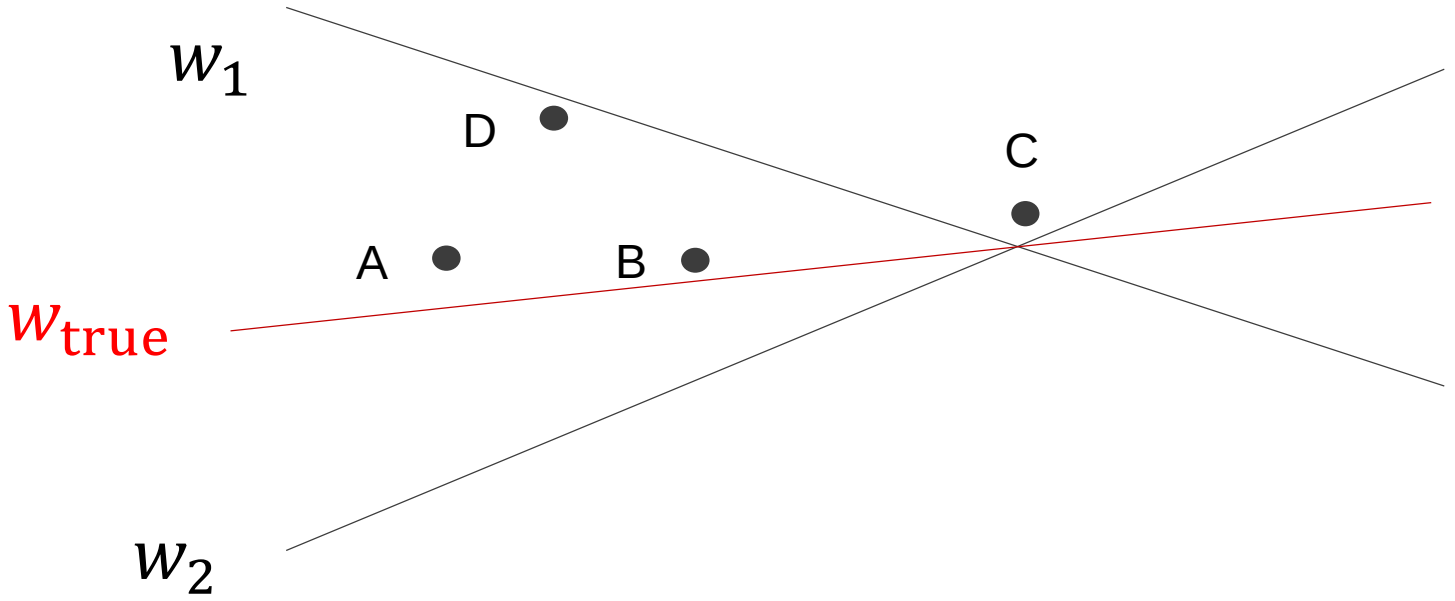
按照BALD获得函数，哪个点能被query?



$$\mathbb{I}[y, \boldsymbol{\omega} | \mathbf{x}, \mathcal{D}_{\text{train}}] = \mathbb{H}[y | \mathbf{x}, \mathcal{D}_{\text{train}}] - \mathbb{E}_{p(\boldsymbol{\omega} | \mathcal{D}_{\text{train}})} \left[\mathbb{H}[y | \mathbf{x}, \boldsymbol{\omega}] \right]$$

$$\approx -\sum_c \left(\frac{1}{T} \sum_t \widehat{p}_c^t \right) \log \left(\frac{1}{T} \sum_t \widehat{p}_c^t \right) + \frac{1}{T} \sum_{c,t} \widehat{p}_c^t \log \widehat{p}_c^t =: \widehat{\mathbb{I}}[y, \boldsymbol{\omega} | \mathbf{x}, \mathcal{D}_{\text{train}}]$$


w_1
w_2
D
C
A
B
w_{true}


w_1
D
C
A
B
w_{true}
w_2

Uncertainty in Deep Learning

Weight Uncertainty in Neural Networks ICML2015

Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks ICML2015

Point Estimates of Neural Networks

The weights can be learnt by maximum likelihood estimation (MLE): given a set of training examples: $\mathcal{D} = (\mathbf{x}_i, \mathbf{y}_i)$ the MLE weights $\mathbf{w}^{\text{MLE}}$ are given by:

$$\mathbf{w}^{\text{MLE}} = \arg\max_{\mathbf{w}} \log P(\mathcal{D}|\mathbf{w}) = \arg\max_{\mathbf{w}} \sum_i \log P(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w})$$

This is typically achieved by gradient descent (e.g., backpropagation), where we assume that $\log P(\mathcal{D}|\mathbf{w})$ is differentiable in $\mathbf{W}$

Regularization can be introduced by placing a prior upon the weights $\mathbf{W}$ and finding the maximum a posteriori (MAP) weights $\mathbf{w}^{\text{MAP}}$:

$$\mathbf{w}^{\text{MAP}} = \arg\max_{\mathbf{w}} \log P(\mathbf{w}|\mathcal{D}) = \arg\max_{\mathbf{w}} \log P(\mathcal{D}|\mathbf{w}) + \log P(\mathbf{w})$$

If $\mathbf{W}$ are given a Gaussian prior, this yields L2 regularization (or weight decay). If $\mathbf{W}$ are given a Laplace prior, then L1 regularization is recovered.

Bayesian inference for neural networks calculates the posterior distribution of the weights given the training data $P(\mathbf{w}|\mathcal{D})$

Predictive distribution of an unknown label $\hat{\mathbf{y}}$ of a test data item $\hat{\mathbf{x}}$ is given by

$$P(\hat{\mathbf{y}}|\hat{\mathbf{x}}) = \mathbb{E}_{P(\mathbf{w}|\mathcal{D})}[P(\hat{\mathbf{y}}|\hat{\mathbf{x}}, \mathbf{w})]$$

Thus taking an expectation under the posterior distribution on weights is equivalent to using an ensemble of an uncountably infinite number of neural networks.

Unfortunately, this is intractable for neural networks of any practical size.

**** suggested finding a variational approximation to the Bayesian posterior distribution on the weights. Variational learning finds the parameters of a distribution on the weights **that minimizes the KL divergence with the true Bayesian posterior on the weights**:

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

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

The cost function is a sum of a **data-dependent part**, which we shall refer to as the **likelihood cost**, and a **prior-dependent part**, which we shall refer to as the **complexity cost**. The cost function embodies a **trade-off** between satisfying the complexity of the data $\mathcal{D}$ and satisfying the simplicity prior $P(\mathbf{w})$.

Unbiased Monte Carlo gradients

Under certain conditions, the derivative of an expectation can be expressed as the expectation of a derivative:

Proposition 1. *Let ϵ be a random variable having a probability density given by $q(\epsilon)$ and let $\mathbf{w} = t(\theta, \epsilon)$ where $t(\theta, \epsilon)$ is a deterministic function. Suppose further that the marginal probability density of $\mathbf{w}$, $q(\mathbf{w}|\theta)$, is such that $q(\epsilon)d\epsilon = q(\mathbf{w}|\theta)d\mathbf{w}$. Then for a function f with derivatives in $\mathbf{w}$:*

$$\begin{aligned}
\frac{\partial}{\partial\theta}\mathbb{E}_{q(\mathbf{w}|\theta)}[f(\mathbf{w},\theta)] &= \frac{\partial}{\partial\theta}\int f(\mathbf{w},\theta)q(\mathbf{w}|\theta)\mathrm{d}\mathbf{w} \\
&= \frac{\partial}{\partial\theta}\int f(\mathbf{w},\theta)q(\epsilon)\mathrm{d}\epsilon \\
&= \mathbb{E}_{q(\epsilon)}\left[\frac{\partial f(\mathbf{w},\theta)}{\partial\mathbf{w}}\frac{\partial\mathbf{w}}{\partial\theta} + \frac{\partial f(\mathbf{w},\theta)}{\partial\theta}\right]
\end{aligned}$$

We apply Proposition 1 to the optimization problem in

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

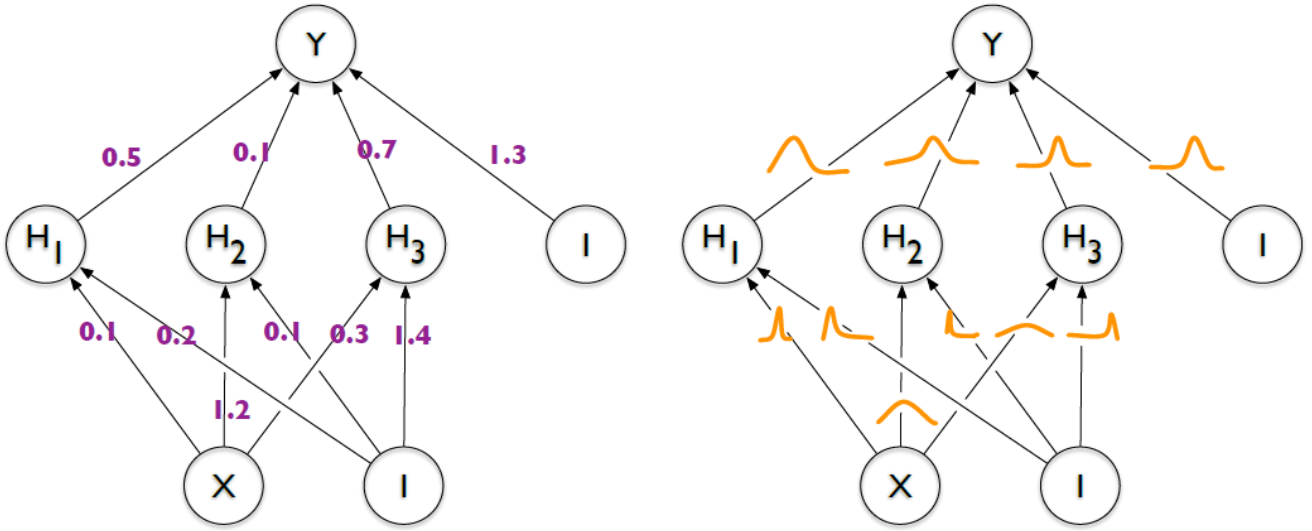
$$\text{let } f(\mathbf{w}, \theta) = \log q(\mathbf{w}|\theta) - \log P(\mathbf{w})P(\mathcal{D}|\mathbf{w})$$

Using Monte Carlo sampling to evaluate the expectations, a backpropagation-like algorithm is obtained for variational Bayesian inferencein neural networks.

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

Gaussian variational posterior

$$p(y|x, \mathbf{X}, \mathbf{Y}) = \int p(y|x, W)p(W|\mathbf{X}, \mathbf{Y})\, dW$$



Suppose that the variational posterior is a diagonal Gaussian distribution, then a sample of the weights **w** can be obtained by sampling a unit Gaussian, shifting it by a mean μ and scaling by a standard deviation σ.

We parameterize the standard deviation pointwise as $\sigma = \log(1 + \exp(\rho))$

The variational posterior parameters are $\theta = (\mu, \rho)$

A posterior sample of the weights $\mathbf{w}$ is $\mathbf{w} = t(\theta, \epsilon) = \mu + \log(1 + \exp(\rho)) \circ \epsilon$

Each step of optimization proceeds as follows:

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

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

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

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

7. Update the variational parameters:

$$\mu \leftarrow \mu - \alpha \Delta_\mu$$
$$\rho \leftarrow \rho - \alpha \Delta_\rho.$$

Scale mixture prior

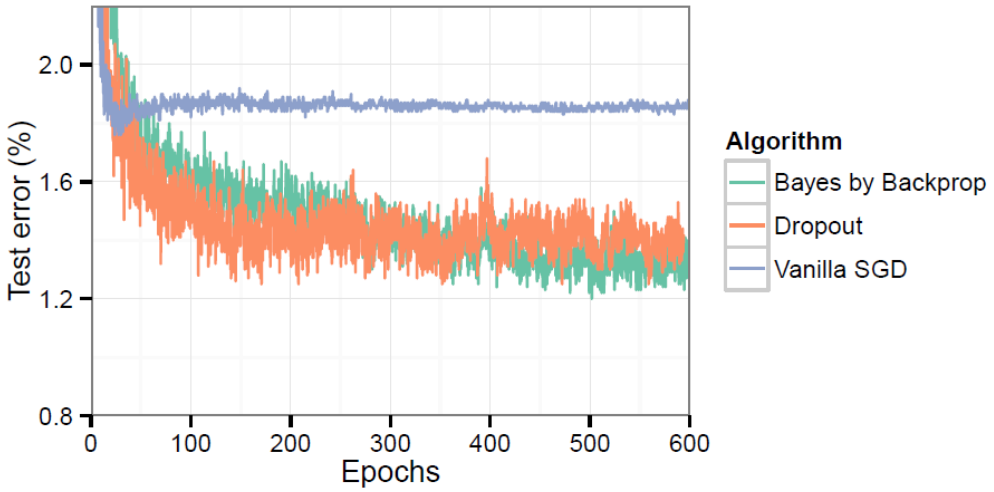


Figure 2. Test error on MNIST as training progresses.

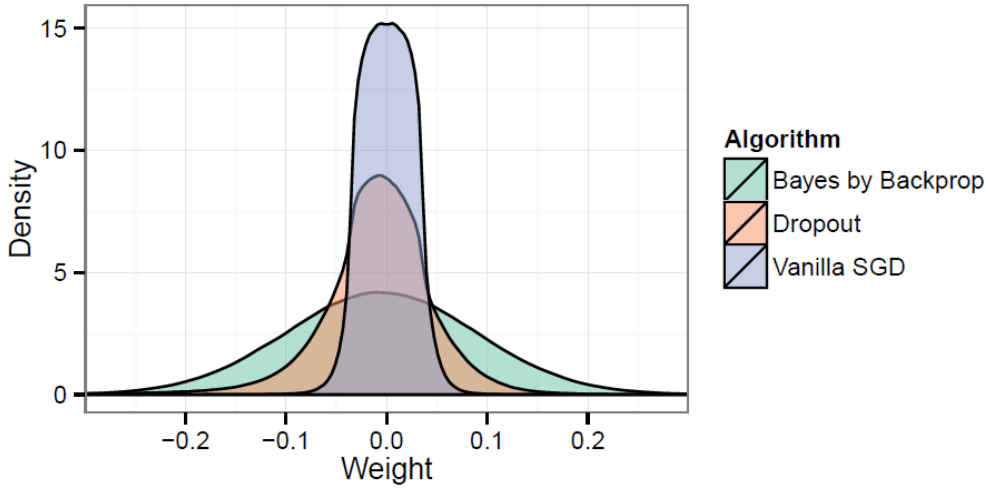


Figure 3. Histogram of the trained weights of the neural network, for Dropout, plain SGD, and samples from Bayes by Backprop.