# Dropout as a Bayesian Approximation: Insights and Applications

Yarin Gal, Zoubin Ghahramani

University of Cambridge
{yg279, zg201}@cam.ac.uk

[GG2015A] [GG2015B]

## In short:

**Dropout neural networks are identical to variational inference in Gaussian processes.**

This gives us...

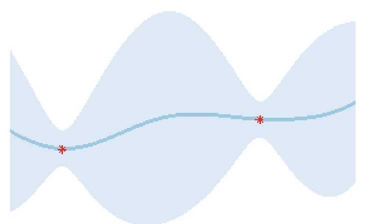
- **Insights** into some of dropout's key properties.
- **Uncertainty in deep learning**.
- Introduce the **Bayesian machinery** into existing deep learning frameworks.
- Straightforward **generalisations** of dropout.

## Background

### What is dropout?

- A technique to avoid over-fitting in multilayer perceptrons (MLPs).
- Given weight matrices $\mathbf{W}_i$ and a bias vector $\mathbf{b}$, sample vectors of Bernoulli random variables $\mathrm{b}_i$ with probabilities $p_i$, to get MLP output:
$$\widehat{y} = \sigma\big(\mathbf{x}(\mathrm{b}_1\mathbf{W}_1) + \mathbf{b}\big)(\mathrm{b}_2\mathbf{W}_2).$$
- Optimisation objective:
$$\mathcal{L}_{\text{dropout}} = \frac{r_1}{2N}\sum_{n=1}^{N}||\mathbf{y}_n - \widehat{\mathbf{y}}_n||_2^2 + r_2\big(||\mathbf{W}_1||_2^2 + ||\mathbf{W}_2||_2^2 + ||\mathbf{b}||_2^2\big).$$
Can easily be generalised to multiple layers and classification.

### Wait, what is a Gaussian process (GP)?



- A powerful tool in statistics, robust to over-fitting.
- Models distributions over functions.
- Supervised/unsupervised, regression/classification.
- Offers uncertainty estimates over the function values (in blue).

- Given training inputs $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N \in \mathbb{R}^{N\times Q}$ and outputs $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^N \in \mathbb{R}^{N\times D}$, estimate a function $\mathbf{y} = \mathbf{f}(\mathbf{x})$ that is **likely to have generated** $\mathbf{Y}$.
- We place a joint Gaussian distribution over all function values:
$$p(\mathbf{Y} \mid \mathbf{X}) = \mathcal{N}(\mathbf{0}, \mathbf{K}(\mathbf{X},\mathbf{X}) + \tau^{-1}\mathbf{I}_N)$$
with precision hyper-parameter $\tau$ and covariance function $\mathbf{K}(\mathbf{X},\mathbf{X})$.

### Ok, what is variational inference?

- Condition the model on a finite set of random variables $\boldsymbol{\omega}$.
- The predictive distribution for a new input point $\mathbf{x}^*$
$$p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) = \int p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega})p(\boldsymbol{\omega}|\mathbf{X},\mathbf{Y})\, \mathrm{d}\boldsymbol{\omega},$$
- The distribution $p(\boldsymbol{\omega}|\mathbf{X},\mathbf{Y})$ cannot be evaluated analytically — define an "easier" approximating *variational* distribution $q(\boldsymbol{\omega})$.
- Minimise the Kullback–Leibler (KL) divergence: $\text{KL}(q(\boldsymbol{\omega}) \mid p(\boldsymbol{\omega}|\mathbf{X},\mathbf{Y}))$.
- Minimising the KL divergence = maximising *log evidence lower bound*,
$$\mathcal{L}_{\text{VI}} := \int q(\boldsymbol{\omega})\log p(\mathbf{Y}|\mathbf{X},\boldsymbol{\omega})\mathrm{d}\boldsymbol{\omega} - \text{KL}(q(\boldsymbol{\omega})||p(\boldsymbol{\omega}))$$
with respect to the variational parameters defining $q(\boldsymbol{\omega})$.

## Proof Sketch

1. Given a GP covariance function
$$\mathbf{K}(\mathbf{x},\mathbf{y}) = \int \mathcal{N}(\mathbf{w}; 0, \mathbf{I}_Q)p(b)\sigma(\mathbf{w}^T\mathbf{x} + b)\sigma(\mathbf{w}^T\mathbf{y} + b)\mathrm{d}\mathbf{w}\mathrm{d}b$$
with some distribution $p(b)$, $\sigma$ element-wise non-linear function (e.g. ReLU/TanH).
2. Approximate with Monte Carlo integration with $K$ terms:
$$\widehat{\mathbf{K}}(\mathbf{x},\mathbf{y}) = \frac{1}{K}\sum_{k=1}^{K}\sigma(\mathbf{w}_k^T\mathbf{x} + b_k)\sigma(\mathbf{w}_k^T\mathbf{y} + b_k)$$
with $\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{I}_Q)$ and $b_k \sim p(b)$. This is a *random covariance function*.
3. The GP predictive distribution is re-parametrised as
$$\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{I}_Q), \quad \mathbf{w}_d \sim \mathcal{N}(0, \mathbf{I}_K), \quad b_k \sim p(b),$$
$$\mathbf{W}_1 = [\mathbf{w}_k]_{k=1}^K, \quad \mathbf{W}_2 = [\mathbf{w}_d]_{d=1}^D, \quad \mathbf{b} = [b_k]_{k=1}^K,$$
$$\boldsymbol{\omega} = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}\}$$
$$p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega}) = \mathcal{N}\left(\mathbf{y}^*; \sqrt{\frac{1}{K}}\sigma(\mathbf{W}_1^T\mathbf{x}^* + \mathbf{b})\mathbf{W}_2, \tau^{-1}\mathbf{I}_N\right)$$
$$p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) = \int p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega})p(\boldsymbol{\omega}|\mathbf{X},\mathbf{Y})\mathrm{d}\boldsymbol{\omega}.$$
4. Use variational distribution $q(\boldsymbol{\omega}) = q(\mathbf{W}_1)q(\mathbf{W}_2)q(\mathbf{b})$ to approximate posterior $p(\boldsymbol{\omega}|\mathbf{X},\mathbf{Y})$:
$$q(\mathbf{W}_1) = \prod_{q=1}^{Q}q(\mathbf{w}_q), \quad q(\mathbf{w}_q) = p_1\mathcal{N}(\mathbf{m}_q, \boldsymbol{\sigma}^2\mathbf{I}_K) + (1-p_1)\mathcal{N}(0, \boldsymbol{\sigma}^2\mathbf{I}_K)$$
with some probability $p_1 \in [0,1]$, scalar $\boldsymbol{\sigma} > 0$ and $\mathbf{M}_1 = [\mathbf{m}_q]_{q=1}^Q \in \mathbb{R}^{K\times D}$ variational parameters. Repeat for $\mathbf{W}_2$.
5. Approximate the log evidence lower bound with Monte Carlo integration with a single sample $\widehat{\boldsymbol{\omega}} \sim q(\boldsymbol{\omega})$:
$$\mathcal{L}_{\text{GP-MC}} = \log p(\mathbf{Y}|\mathbf{X},\widehat{\boldsymbol{\omega}}) - \frac{p_1}{2}||\mathbf{M}_1||_2^2 - \frac{p_2}{2}||\mathbf{M}_2||_2^2 - \frac{1}{2}||\mathbf{m}||_2^2.$$
This is an unbiased estimator of $\mathcal{L}_{\text{VI}}$.
6. For regression we maximise
$$\mathcal{L}_{\text{GP-MC}} \propto -\frac{\gamma\tau}{2}\sum_{n=1}^{N}||\mathbf{y}_n - \widehat{\mathbf{y}}_n||_2^2 - \frac{\gamma p_1}{2}||\mathbf{M}_1||_2^2 - \frac{\gamma p_2}{2}||\mathbf{M}_2||_2^2 - \frac{\gamma}{2}||\mathbf{m}||_2^2$$
recovering dropout objective with appropriate $\gamma$ and model precision $\tau$ for small enough $\boldsymbol{\sigma}$. Can easily be generalised to multiple layers and classification.

## Insights

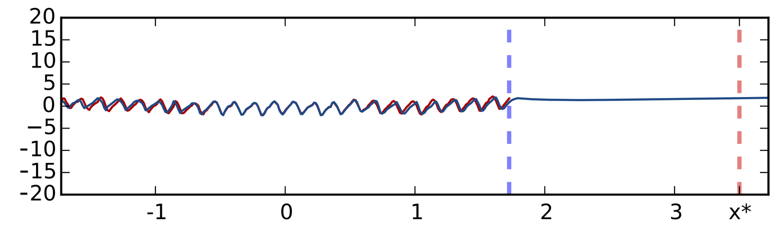
- Alternative explanation to dropout robustness to over-fitting.
- Weight-decay for the dropped-out weights should be scaled by the probability of the weights not to be dropped.
- Dropout extensions such as $\mathbf{m}_i \cdot \mathcal{N}(1,1)$ suggested in [S2014] are alternative approximating distributions.
- At test time should use Monte Carlo integration with $T$ terms
$$p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) \approx \frac{1}{T}\sum_{i=1}^{T}p(\mathbf{y}^*|\mathbf{x}^*, \widehat{\boldsymbol{\omega}}_i)$$
with $\widehat{\boldsymbol{\omega}}_i \sim q(\boldsymbol{\omega})$, named *MC dropout*. Mentioned in [S2014] as model averaging.

## Example Applications

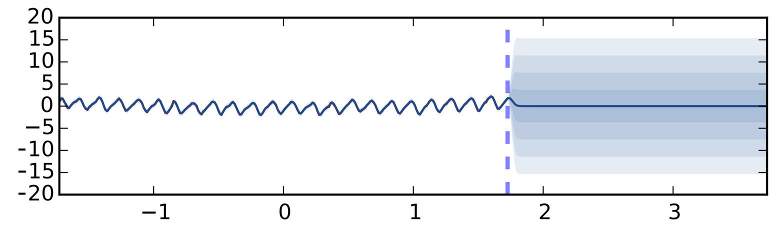
### Model uncertainty

- We can obtain model uncertainty **from existing models**

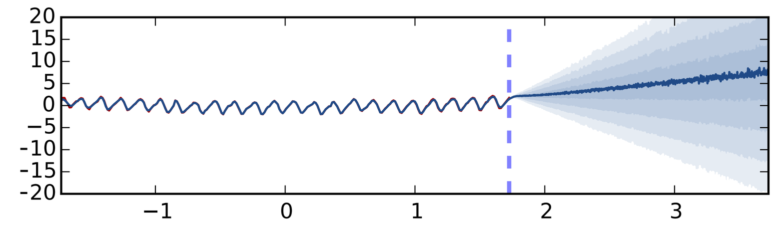
Predictive mean and uncertainties on the Mauna Loa $CO_2$ concentrations dataset:



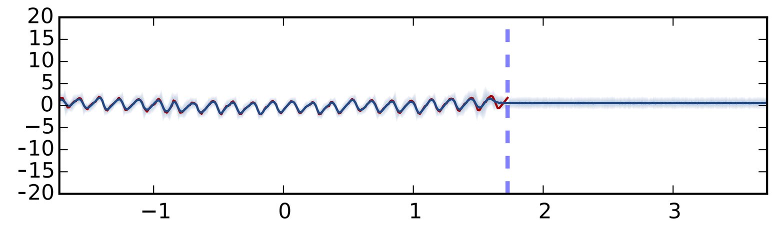
*Standard dropout, ReLU non-linearities*



*Gaussian process, SE covariance function*



*MC dropout, same network as above*



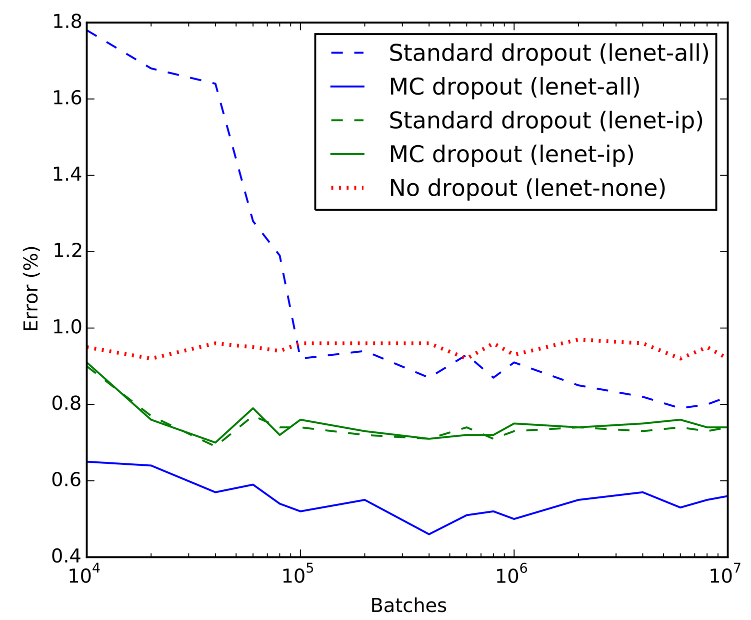
*MC dropout, TanH non-linearities*

Complete treatment with **applications to Deep Reinforcement Learning** given in [GG2015A].

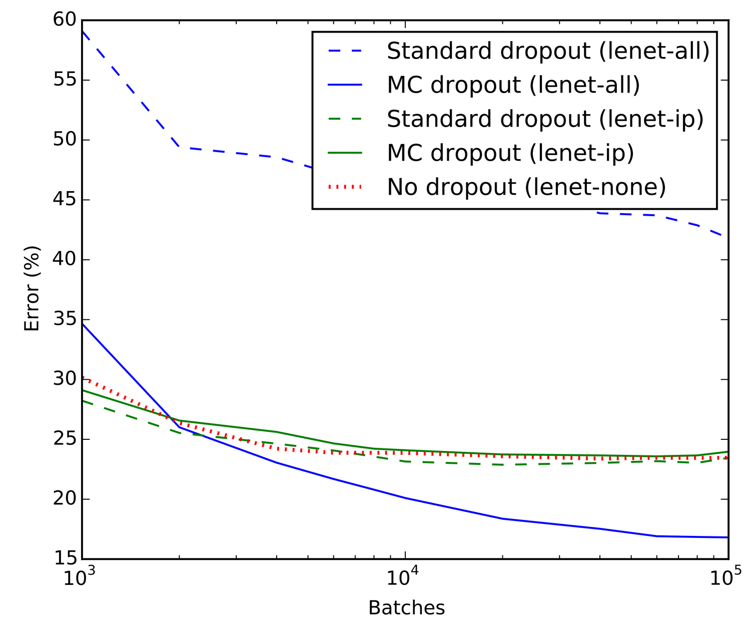
### Bayesian convolutional neural networks (convnets)

- We can implement Bayesian convnets with existing tools in the field.

Test set error *on log scale* for LeNet:




*MNIST*




*CIFAR-10*

*In blue is our Bayesian convnet implementation (lenet-all), in green is dropout applied after the fully connected layer alone (lenet-ip), in red no dropout (lenet-none). Standard dropout shown with a dashed line, MC dropout shown with a solid line.*

Complete treatment with **new state-of-the-art results on CIFAR-10** given in [GG2015B].

### Principled extensions of dropout

- Use of new approximating distributions.
- Also mathematically identical to variational inference in Bayesian neural networks.

**References:**

[GG2015A] Gal, Y, and Ghahramani, Z. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning, 2015.

[GG2015B] Gal, Y, and Ghahramani, Z. Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference, 2015.

[S2014] Srivastava, N, Hinton, G, Krizhevsky, A, Sutskever, I, and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. JMLR, 2014.