Weight Uncertainty in Neural Networks

by Antoine Logeais

April 2020

Content

Introduction

What purpose ?

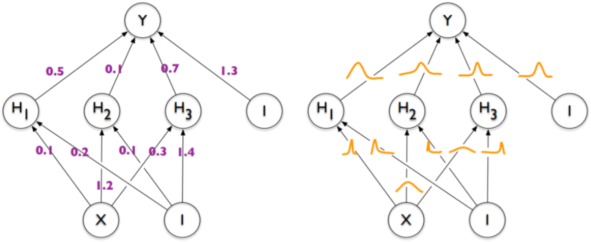
To drive the exploration-exploitation trade-off in reinforcement learning and avoid over adjustment

How to do it?

Using variational Bayesian learning to introduce uncertainty in the weights of the network to develop an algorithm called Bayes by Backprop

A probability distribution on the weights

All weights in our neural networks are represented by probability distributions over possible values, rather than having a single fixed value as is the norm. Unlike other ensemble methods, this method typically only doubles the number of parameters yet trains an infinite ensemble



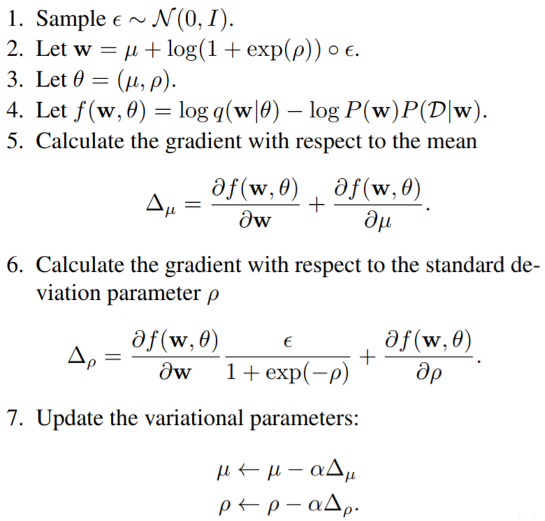
Classical Backpropagation Bayes by Backprop

Gaussian variational posterior

- 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 parameterise the standard deviation pointwise as $\sigma = log(1 + exp(p))$
- Thus the transform from a sample of parameter-free noise and the variational posterior parameters that yields a posterior sample of the weights w is : $w = t(\theta, \epsilon) = \mu + log(1 + exp(p)) \circ \epsilon$
  where $\circ$ is pointwise multiplication.

Gaussian variational posterior

Each step of optimisation 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

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

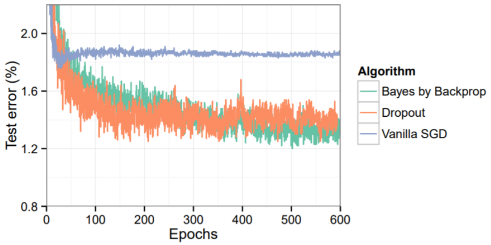
7. Update the variational parameters:

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

Results

Classification on MNIST

Method	# Units/Layer	# Weights	Test Error
SGD, no regularisation (Simard et al., 2003)	800	1.3m	1.6%
SGD, dropout (Hinton et al., 2012)			$\approx$ 1.3%
SGD, dropconnect (Wan et al., 2013)	800	1.3m	**1.2%***
SGD	400	500k	1.83%
	800	1.3m	1.84%
	1200	2.4m	1.88%
SGD, dropout	400	500k	1.51%
	800	1.3m	1.33%
	1200	2.4m	1.36%
Bayes by Backprop, Gaussian	400	500k	1.82%
	800	1.3m	1.99%
	1200	2.4m	2.04%
Bayes by Backprop, Scale mixture	400	500k	1.36%
	800	1.3m	1.34%
	1200	2.4m	**1.32%**



Results

Bandits on Mushroom Task

Contextual bandits are simple reinforcement learning problems without persistent state. At each step an agent is presented with a context x and a choice of one of K possible actions a. Different actions yield different unknown rewards r. The agent must pick the action that yields the highest expected reward. The context is assumed to be presented independent of any previous actions, rewards or contexts.

Here an agent expects to receive a reward of 5 for eating an edible mushroom, but an expected reward of -15 for eating a poisonous mushroom.

Results

Bandits on Mushroom Task

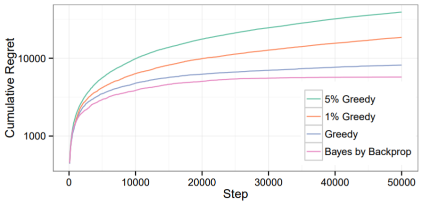


Figure 6. Comparison of cumulative regret of various agents on the mushroom bandit task, averaged over five runs. Lower is better.

This compares a Bayes by Backprop agent with three ϵ-greedy agents, for values of ϵ of 0 % (pure greedy), 1%, and 5%. An ϵ of 5% appears to over-explore, whereas a purely greedy agent does poorly at the beginning, greedily electing to eat nothing, but then does much better once it has seen enough data.

The Bayes by Backprop agent explores from the beginning, both eating and ignoring mushrooms and quickly converges on eating and non-eating with an almost perfect rate

Regression curves

We generated training data from the curve:

$$y = x + 0.3sin(2(x+\epsilon)) + 0.3sin(4(x+\epsilon)) + \epsilon$$

where $N(0, 0.02)$

This figure shows two examples of fitting a neural network to these data, minimising a conditional Gaussian loss

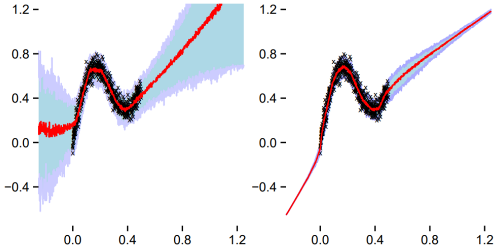


Figure 5. Regression of noisy data with interquatile ranges. Black crosses are training samples. Red lines are median predictions. Blue/purple region is interquartile range. Left: Bayes by Backprop neural network, Right: standard neural network.

Conclusion

- Bayes by Backprop is comparable to dropout on MNIST classification
- On contextual bandits, we showed how Bayes by Backprop can automatically learn how to trade-off exploration and exploitation
- On a non-linear regression problem the uncertainty introduced allows the network to make more reasonable predictions about unseen data
- All of the operations used are readily implemented on a GPU