Weight Uncertainty in Neural Networks

Google DeepMind (2015)

Uncertainty on Weights

Problem: Plain feedforward neural networks are prone to overfitting.

Solution: Use variational Bayesian learning to introduce uncertainty on the weights in the network. (*Bayes by Backprop*)

Motivation:

- Regularization via a complexity cost on the weights
- Richer representations and predictions from cheap model averaging
- Exploration in simple RL problems

Probabilistic View of Neural Networks

Probabilistic Model: $P(y|x, w) \text{ where } x \in \mathcal{R}^p,\ y \in Y$

Training the weights:

$$w^{MLE} = \arg\max_{w} \underbrace{\sum_i \log P(y_i|x_i, w)}_{\log P(D|w)}$$

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

Black Box Variational Inference for Neural Networks

Variational Model:

$$\theta = \{\mu, \rho\}$$

$$\sigma = \underbrace{\log(1 + \exp(\rho))}_{\text{softplus}}$$

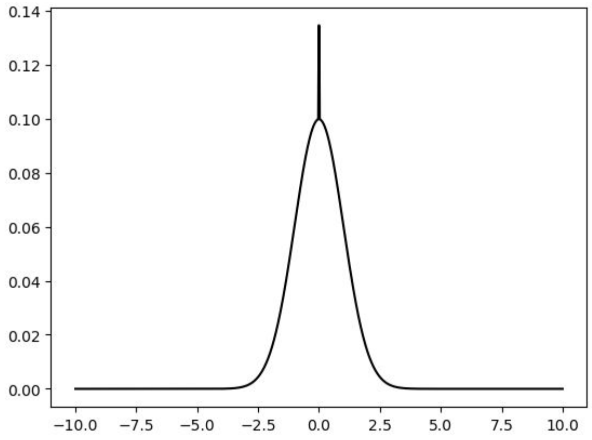
$$q(w;\theta) = \mathcal{N}\left(w; \mu, \sigma^2\right)$$

Objective:

$$\theta^* = \arg\min_\theta \mathrm{KL}\left(q(w;\theta) \,||\, P(w \mid \mathcal{D})\right)$$

$$= \arg\min_\theta \underbrace{\mathrm{KL}\left(q(w;\theta) \,||\, P(w)\right)}_{\text{prior/regularization}} + \underbrace{\mathbb{E}_{q(w;\theta)}\left[-\log P(\mathcal{D} \mid w)\right]}_{\text{data-driven/likelihood}}$$

$$\mathcal{F}(\theta) = \mathbb{E}_{q(w;\theta)}[\underbrace{\log q(w;\theta) - \log P(w) - \log P(\mathcal{D} \mid w)}_{f(w;\theta)}]$$



Black Box Variational Inference for Neural Networks

Reparameterization trick:

$$\begin{aligned}\mathcal{F}(\theta) &= \mathbb{E}_{q(\omega;\theta)}\left[f(w;\theta)\right] \\ &= \mathbb{E}_{p(\epsilon)}\left[f(w;\theta)\right] \\ \nabla_\theta \mathcal{F}(\theta) &= \mathbb{E}_{p(\epsilon)}\left[\nabla_\theta f(w;\theta)\right]\end{aligned}$$

Backpropagation:

$$\begin{aligned}\nabla_\mu \mathcal{F}(\theta) &= \mathbb{E}_{p(\epsilon)}\left[\frac{\partial f(w;\theta)}{\partial w} + \frac{\partial f(w;\theta)}{\partial \mu}\right] \\ \nabla_\rho \mathcal{F}(\theta) &= \mathbb{E}_{p(\epsilon)}\left[\frac{\partial f(w;\theta)}{\partial w}\frac{\epsilon}{1+\exp(-\rho)} + \frac{\partial f(w;\theta)}{\partial \rho}\right] \\ \frac{\partial f(w;\theta)}{\partial w} &\equiv \text{Backpropagation}\end{aligned}$$

Bayes by Backprop

1. Sample $\epsilon \sim \mathcal{N}(0, I)$
2. Let $w = \mu + \log(1 + \exp(\rho)) \circ \epsilon$
3. Let $\theta = (\mu, \rho)$
4. Let $f(w, \theta) = \log q(w|\theta) - \log P(w) - \log P(D|w)$
5. Calculate gradients:

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

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

7. Update the variational parameters:

$$\mu \leftarrow \mu - \alpha\Delta_\mu$$

$$\rho \leftarrow \rho - \alpha\Delta_\rho$$

Results

Our Implementations:

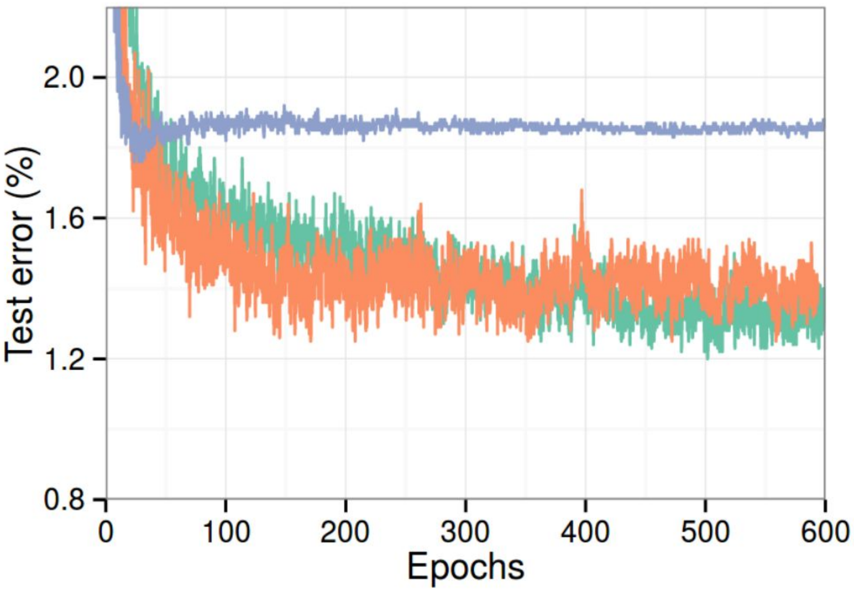
- Homebrewed numpy w/autograd
- PyTorch

Trained Model with ‘Backprop by Bayes’ on three domains:

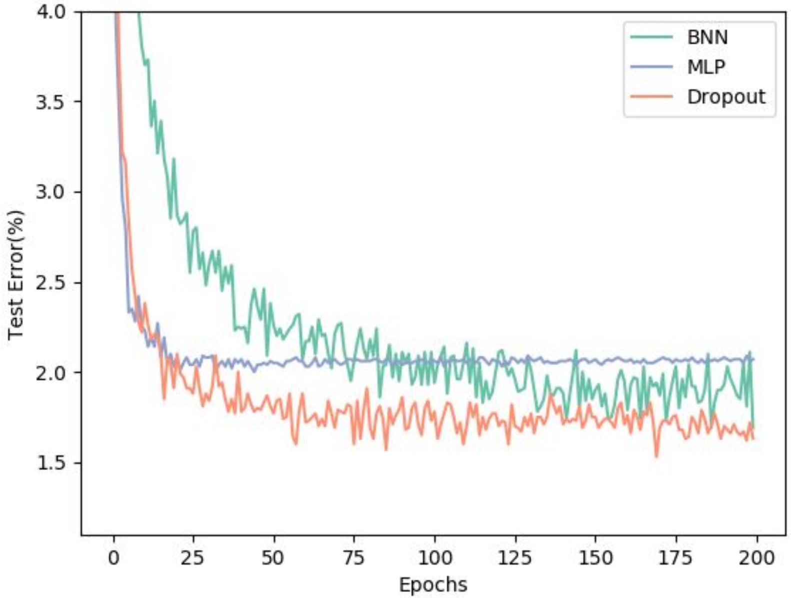
- Classification: MNIST digit recognition
- Regression: 1D synthetic non-linear function
- Contextual Bandits: “mushroom” domain

MNIST - Test Error

Paper (2 Layers of 1200 Units)

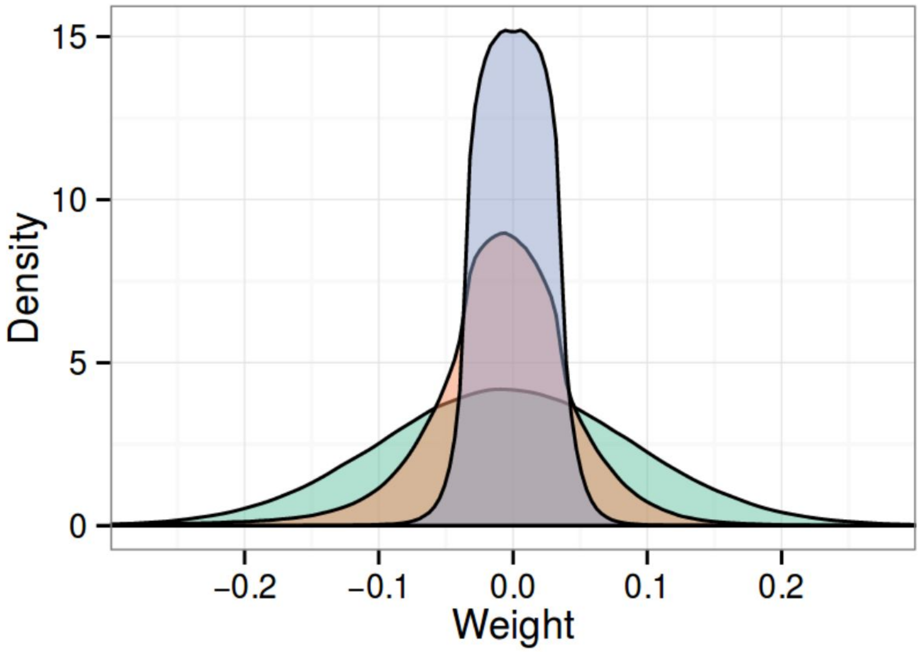


Our Implementation (2 Layers of 400 Units)

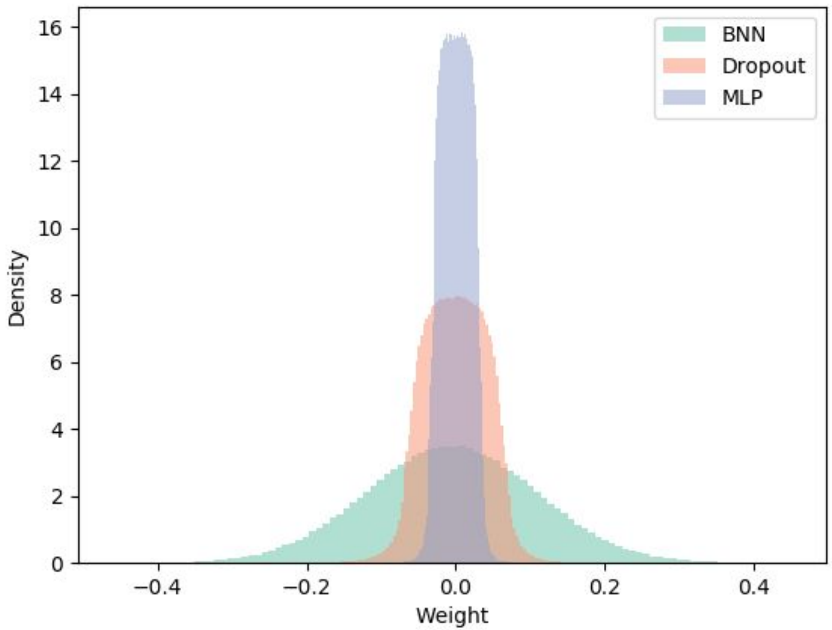


MNIST - Weight Histogram

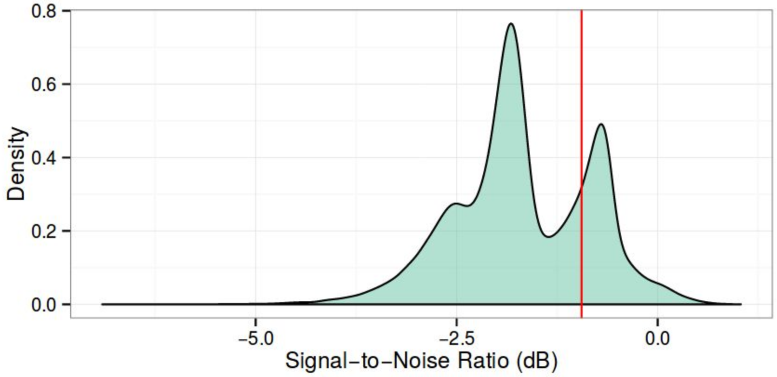
Paper



Our Implementation



MNIST - Weight Pruning



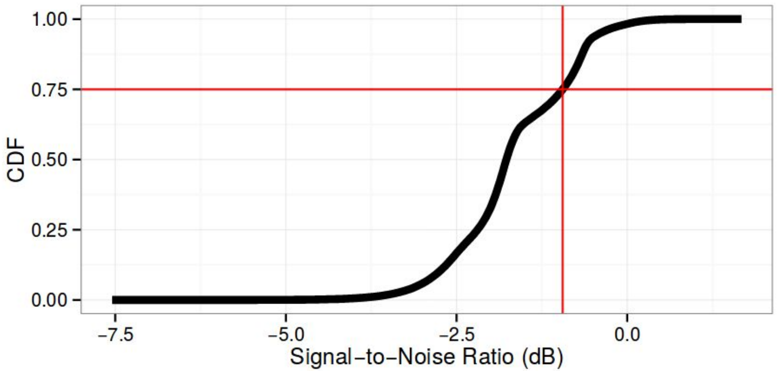
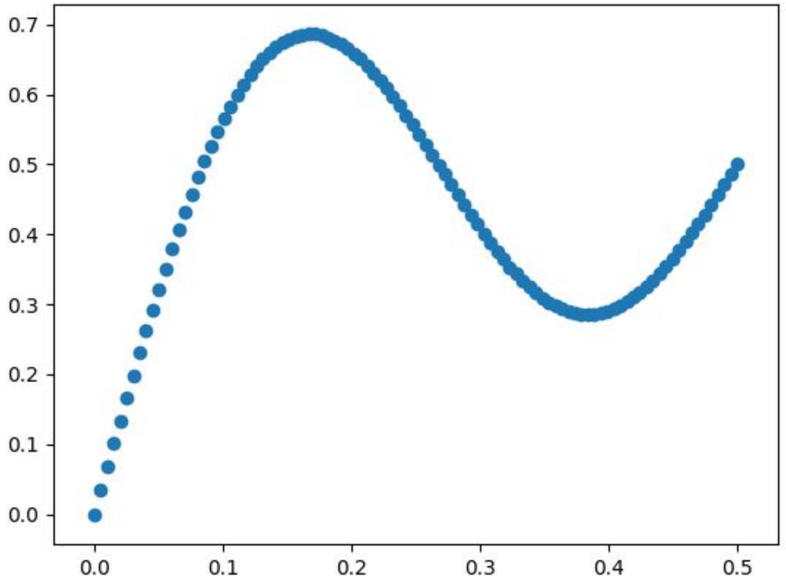


Table 2. Classification Errors after Weight pruning

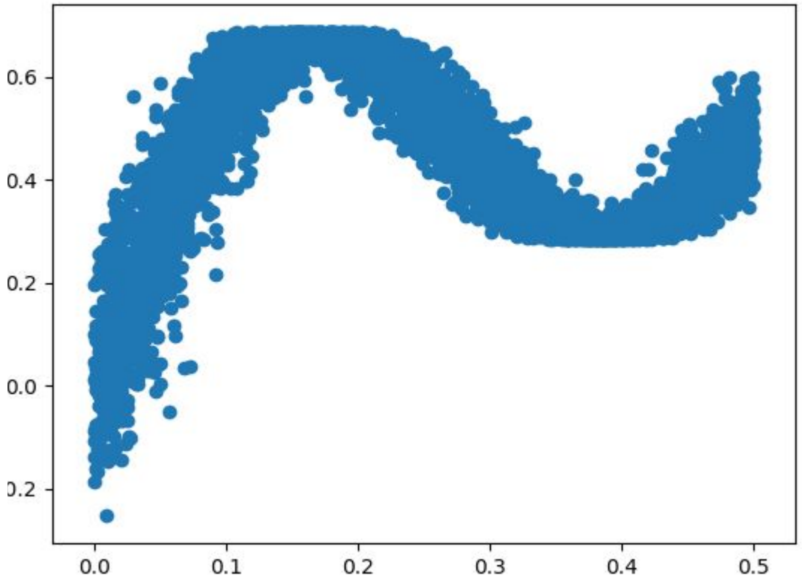
Proportion removed	# Weights	Test Error
0%	2.4m	1.24%
50%	1.2m	1.24%
75%	600k	1.24%
95%	120k	1.29%
98%	48k	1.39%

Regression - Non-Linear Functions

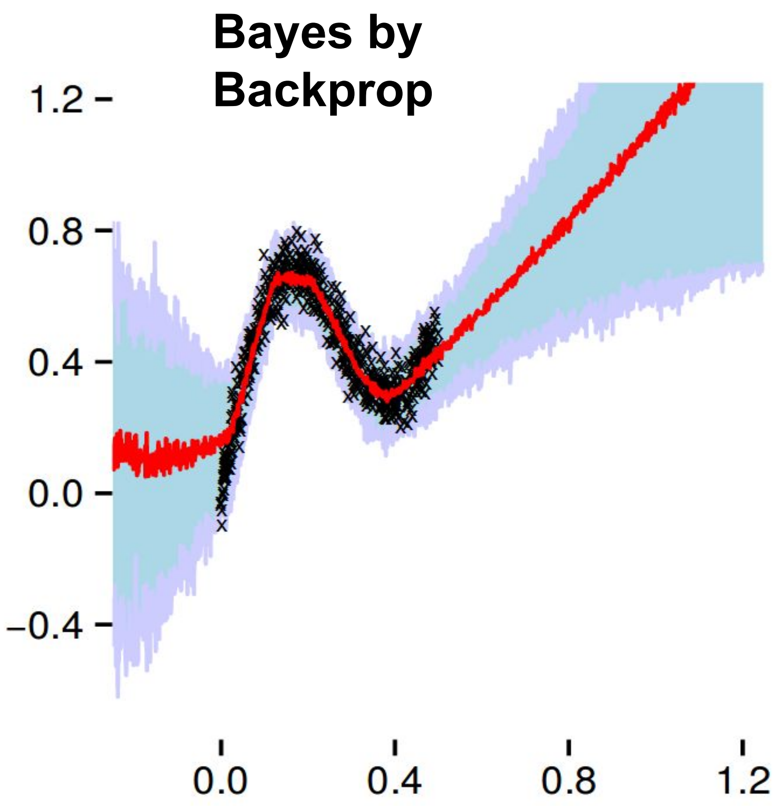
$$y = x + 0.3\sin(2\pi x) + 0.3\sin(4\pi x)$$

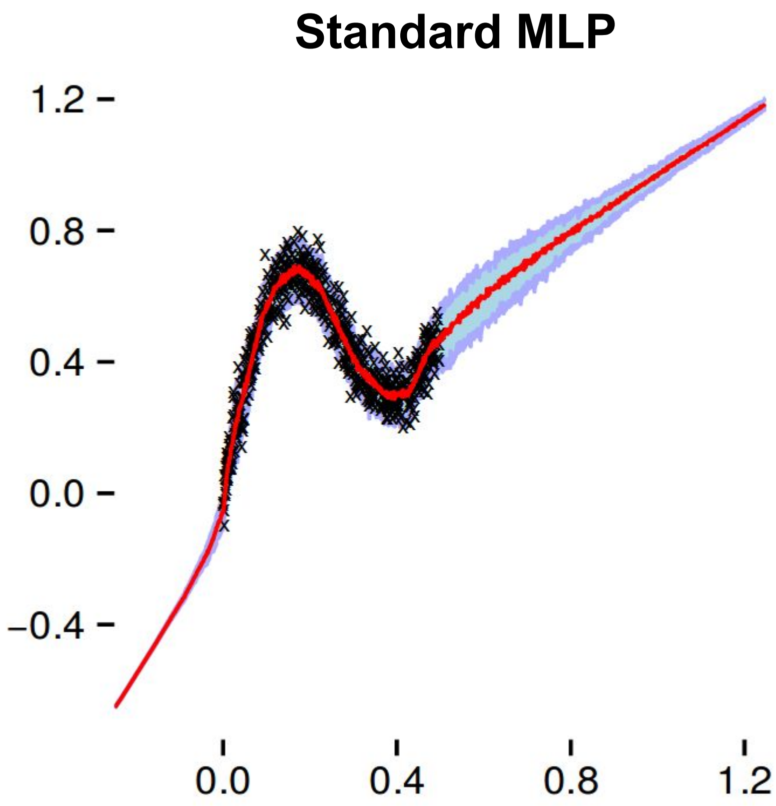


$$y = x + 0.3\sin(2\pi(x+\epsilon)) + 0.3\sin(4\pi(x+\epsilon)) + \epsilon$$
$$\epsilon \sim \mathcal{N}(0, 0.02)$$



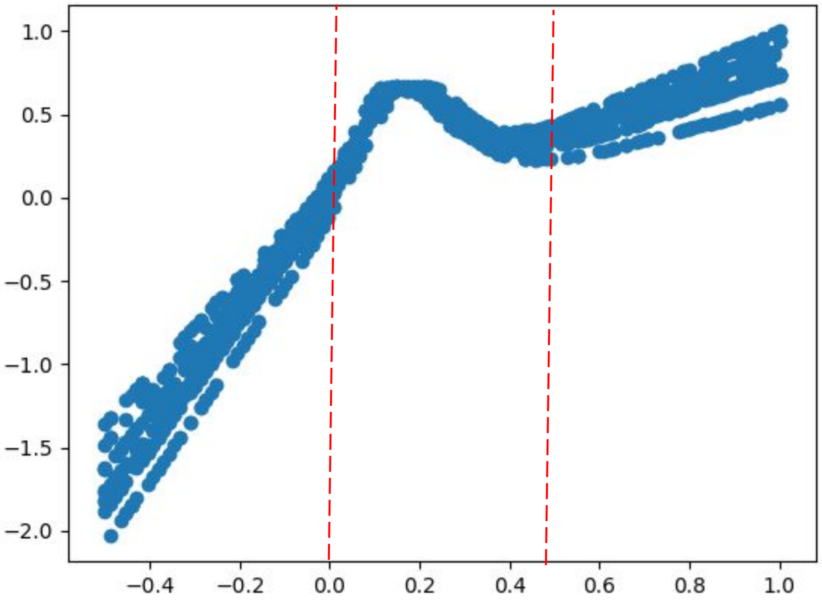
Regression - Uncertainty in New Data



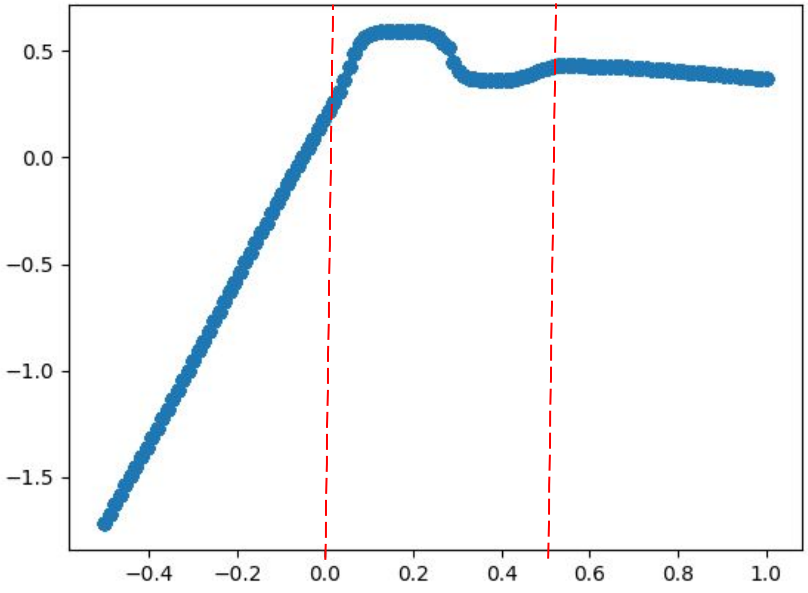


Regression - Uncertainty in New Data

Bayes by Backprop



MLP



Multi-Armed Bandits (Recap)

Objectives:

- Exploitation - Choose “best” bandit;
- Exploration - Discover better bandits;

Approaches:

- Non-Bayesian - Point-estimate of expected rewards, explore w/ ϵ-greedy
- Bayesian (optimal) - Full belief over rewards, optimize long term return.
- Bayesian (heuristic) - e.g. Thompson sampling.

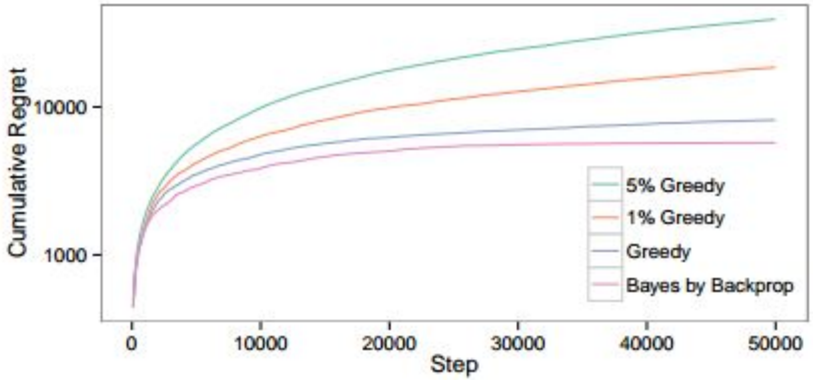
Contextual Bandits - Mushroom Domain

Context: Mushroom features $x \in \{0,1\}^{117}$ (color, size, shape, population, …)

Actions: Eat or Ignore

Rewards:

- Ignore $\Rightarrow$ 0
- Eat + Edible $\Rightarrow$ 5
- Eat + Toxic $\Rightarrow$ Discrete({5, -35})



Conclusions

Contribution: Extension of Backprop training for approx. Bayesian inference.

Pros:

- Performs similarly to non-Bayesian state-of-the-art (SGD + dropout).
- Learns sensible notion of uncertainty.
- Straightforward parallelization.
- Results partially replicated.

Cons:

- Finicky, sensitive to hyperparameters (typical of NNs, BBVI innocent?).
- Substantially slower (more parameters, more samples required).