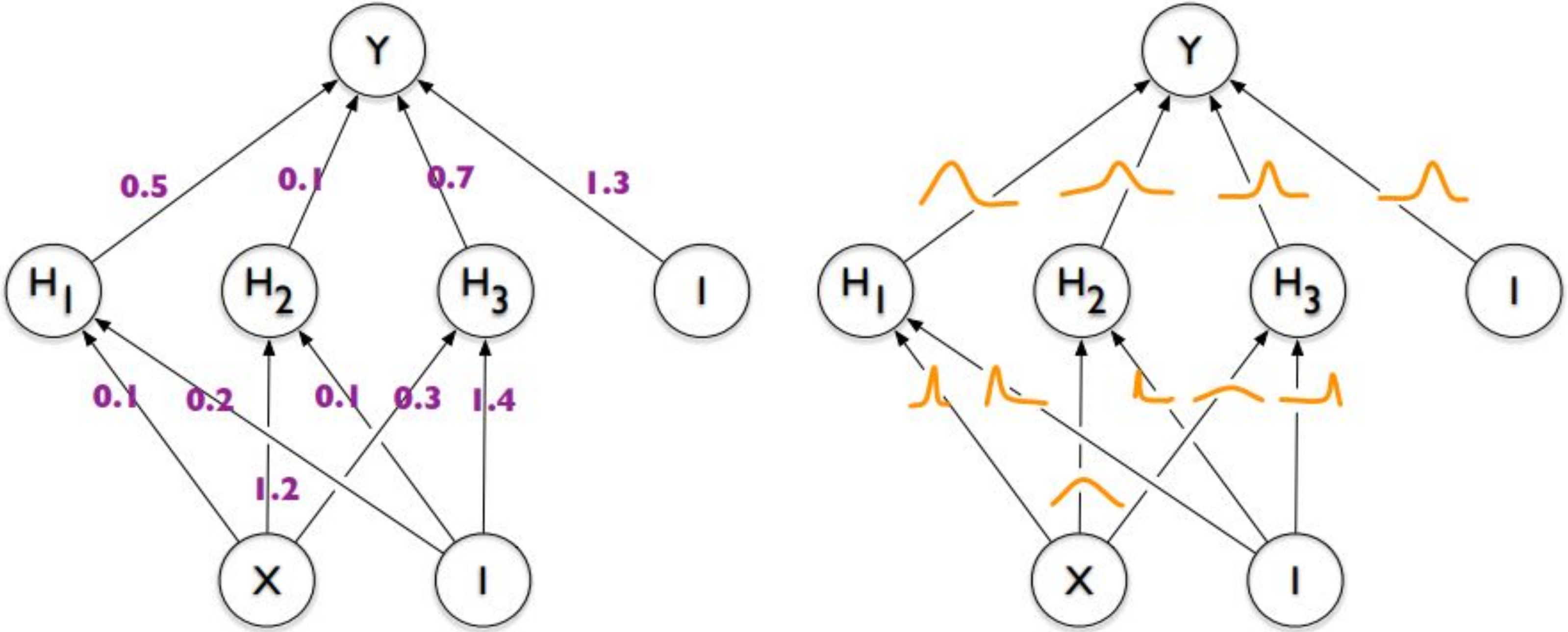
Weight Uncertainty in Neural Networks

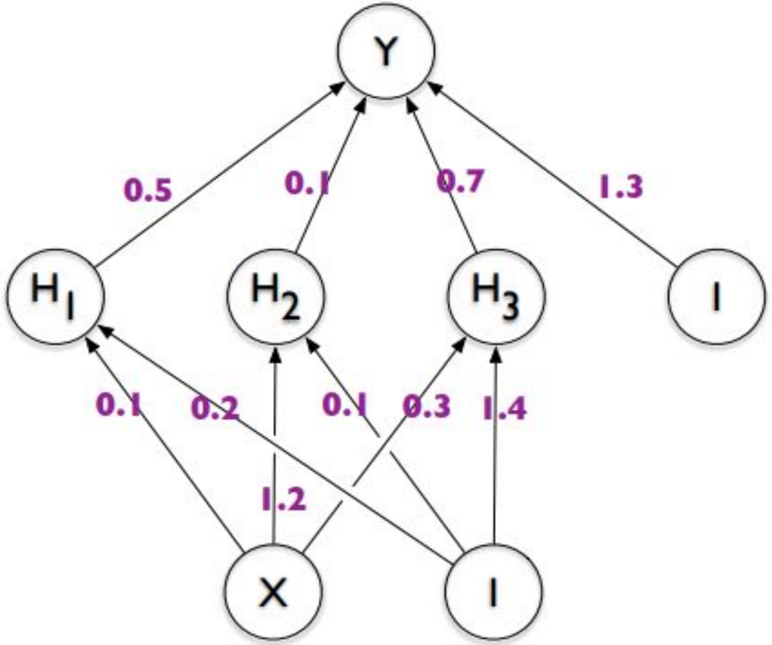
Slides by Cameron Loewen (with a review slide from 696h)

Introduction

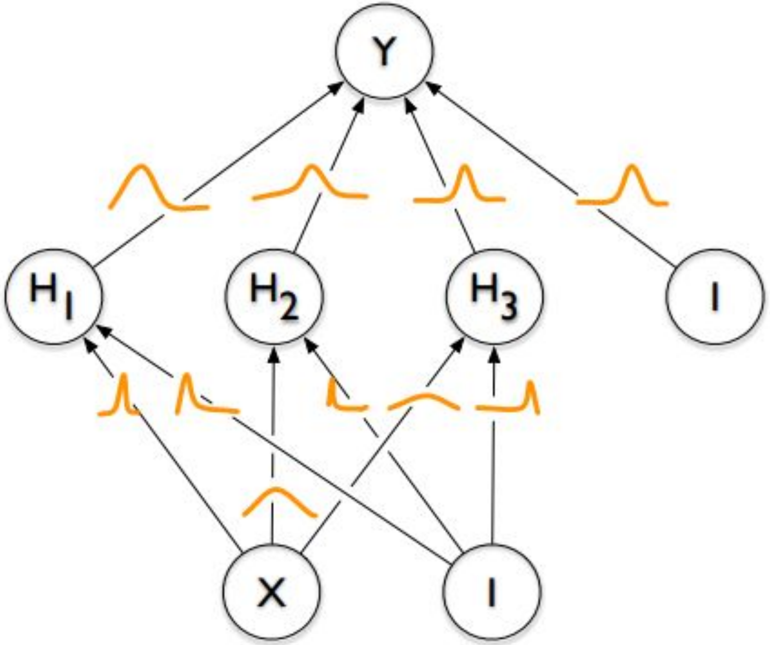


Introduction: Typical Neural Network

- Deterministic weights
- Overly Confident
- Weights determined using MLE



Introduction: Bayesian Network



- Sampled weights
- Can model uncertainty
- Weights determined by learned distribution (more parameters)

Use cases for Bayesian Neural Networks

- A bayesian neural network can be used:
  - Wherever a normal neural network would be used
  - To better quantify uncertainty in classification or regression estimates
  - To have a better model for exploration/reward trade off in bandits

Using Probability Models with a Neural Network

Classifying with a Neural Network

- Assign probability to an output (given x)
  - P(y|x, w), where x is our feature vector

- Theoretical way with Posterior Distribution
  - $$P(\hat{\mathbf{y}}|\hat{\mathbf{x}}) = \mathbb{E}_{P(\mathbf{w}|\mathcal{D})}[P(\hat{\mathbf{y}}|\hat{\mathbf{x}}, \mathbf{w})]$$

Determining Weights

This is an optimization problem solved with backpropagation

MLE

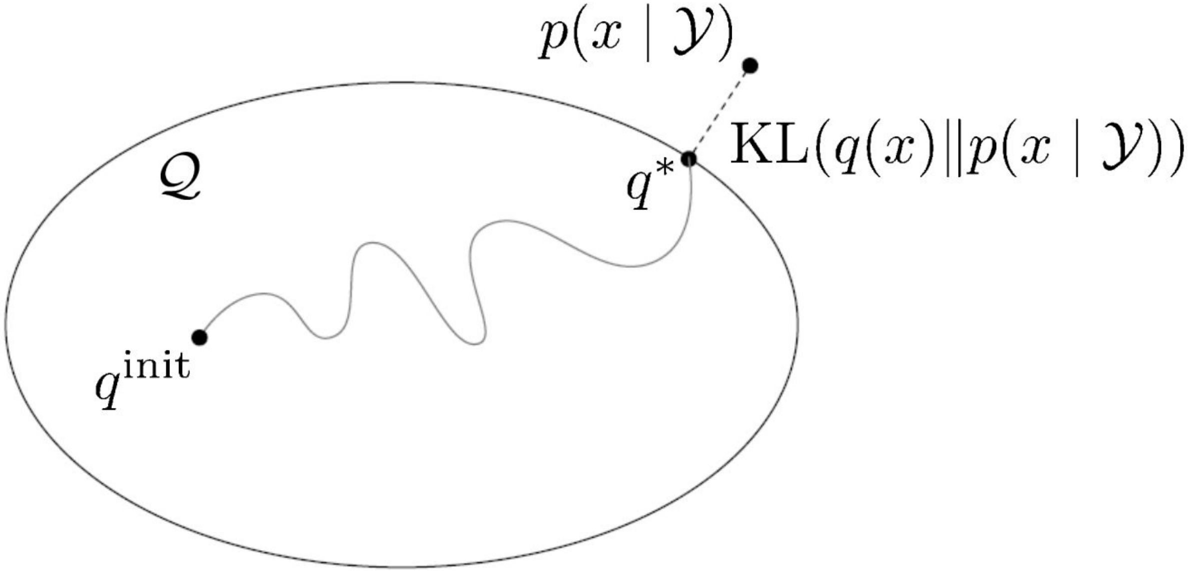
$$\mathbf{w}^{\text{MLE}} = \arg\max_{\mathbf{w}} \log P(\mathcal{D}|\mathbf{w})$$
$$= \arg\max_{\mathbf{w}} \sum_i \log P(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w})$$

MAP

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

Being Bayesian By Backpropagation

Recall: Variational Inference



Minimize KL between $q(x)$ and posterior $p(x \mid \mathcal{Y})$.

Variational Lower Bound (Step 1)

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

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

Reparameterization (Intro)

- Say we want to represent $X \sim N(\mu, \sigma^2)$

- Introduce noise $\boldsymbol{\varepsilon} \sim N(0, 1)$

- Use deterministic function **f(ε)**:

$$\mathbf{f(\varepsilon) = \mu + \sigma * \varepsilon}$$

Proposition (Step 2)

Definitions

Let $q(\boldsymbol{\varepsilon})$ be the probability density of $\boldsymbol{\varepsilon}$

Let $w = t(\theta, \boldsymbol{\varepsilon})$, where t is a deterministic function

Given that $q(\boldsymbol{\varepsilon})d\boldsymbol{\varepsilon} = q(w|\theta)d\theta$

Proposition (Step 2)

For a function f:

$$\frac{\partial}{\partial \theta}\mathbb{E}_{q(\mathbf{w}|\theta)}[f(\mathbf{w},\theta)] = \mathbb{E}_{q(\epsilon)}\left[\frac{\partial f(\mathbf{w},\theta)}{\partial \mathbf{w}}\frac{\partial \mathbf{w}}{\partial \theta} + \frac{\partial f(\mathbf{w},\theta)}{\partial \theta}\right]$$

This shows that optimizing some expectation of a function can be done as an expectation.

Thus, we can use Monte Carlo Methods

Proposition (Step 2)

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

$\square$

Combining Steps 1 and 2

(1) $$\theta^\star = \arg\min_\theta \int q(\mathbf{w}|\theta) \log \frac{q(\mathbf{w}|\theta)}{P(\mathbf{w})P(\mathcal{D}|\mathbf{w})} \mathrm{d}\mathbf{w}$$

(2) $$\frac{\partial}{\partial \theta}\mathbb{E}_{q(\mathbf{w}|\theta)}[f(\mathbf{w},\theta)] = \mathbb{E}_{q(\epsilon)}\left[\frac{\partial f(\mathbf{w},\theta)}{\partial \mathbf{w}}\frac{\partial \mathbf{w}}{\partial \theta} + \frac{\partial f(\mathbf{w},\theta)}{\partial \theta}\right]$$

If we change out the integral for expectation, and set f(w, θ) to be the inside of the expectation, (1) becomes the left hand side of (2)

Combining Steps 1 and 2

(1) $$\theta^\star = \arg\min_\theta \int q(\mathbf{w}|\theta) \log \frac{q(\mathbf{w}|\theta)}{P(\mathbf{w})P(\mathcal{D}|\mathbf{w})} d\mathbf{w}$$

(2) $$\frac{\partial}{\partial\theta}\mathbb{E}_{q(\mathbf{w}|\theta)}[f(\mathbf{w},\theta)] = \mathbb{E}_{q(\epsilon)}\left[\frac{\partial f(\mathbf{w},\theta)}{\partial \mathbf{w}}\frac{\partial \mathbf{w}}{\partial\theta} + \frac{\partial f(\mathbf{w},\theta)}{\partial\theta}\right]$$

If we change out the integral for expectation, and set f(w, θ) to be the inside of the expectation, (1) becomes the left hand side of (2)

Our Cost Function

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

Algorithm uses unbiased estimates of this cost function to learn a distribution over the weights

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

Variational Posterior: Diagonal Gaussian

Algorithm using Diagonal Gaussian

Reparameterization step

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

Algorithm using Diagonal Gaussian

Backpropagation Step

5. Calculate the gradient with respect to the mean

$$\Delta_\mu = \frac{\partial f(\mathbf{w},\theta)}{\partial \mathbf{w}} + \frac{\partial f(\mathbf{w},\theta)}{\partial \mu}. \tag{3}$$

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

$$\Delta_\rho = \frac{\partial f(\mathbf{w},\theta)}{\partial \mathbf{w}} \frac{\epsilon}{1+\exp(-\rho)} + \frac{\partial f(\mathbf{w},\theta)}{\partial \rho}. \tag{4}$$

7. Update the variational parameters:

$$\mu \leftarrow \mu - \alpha\Delta_\mu \tag{5}$$

$$\rho \leftarrow \rho - \alpha\Delta_\rho. \tag{6}$$

Prior:
Scale Mixture

Prior Probability Function

- Resembles a spike and slab
- One Gaussian has high variance
- Other has low variance

$$P(\mathbf{w}) = \prod_j \pi \mathcal{N}(\mathbf{w}_j | 0, \sigma_1^2) + (1 - \pi)\mathcal{N}(\mathbf{w}_j | 0, \sigma_2^2)$$

Do we Optimize the Prior

- Experimentation had worse results when optimizing the prior
- This practice is known as empirical bayes
- The reasons for the results could be:
  - There are far fewer prior parameters to optimize (relative to posterior)
  - Poor initial parameters are difficult to move away from (weird local extremas)

Tangent:
Contextual Bandits

Using our Network for RL

Changes:

- X is now our context
- New parameter a, for action
- P(r | x, a, w) instead of P(y | x, w)

Intuition:

- Uncertainty captured better
- Allows intuitive basis for exploration/ exploitation balance

Sampling Differences

Thompson:

1. Sample new parameters
2. Pick action with highest expected reward
3. Update model, repeat

Adapted:

1. Sample weights from variational posterior
2. Receive context x
3. Pick action a that maximizes? the expected reward
4. Receive reward r
5. Update variational parameters θ and repeat

The paper says minimizes, which seems to be a typo, thoughts?

Results: Finally

First Results: MNIST with Bayesian Neural Network

The results are on the following slide

They show:

- Stochastic Gradient Descent and variations
- Two bayes networks (different priors)
- Varying Layers and weights/hyperparameters

Table 1. Classification Error Rates on MNIST. ⋆ indicates result used an ensemble of 5 networks.

Method	**# Units/Layer**	**# Weights**	**Test Error**
SGD, no regularisation (Simard et al., 2003)	800	1.3m	1.6%
SGD, dropout (Hinton et al., 2012)			≈ 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%**

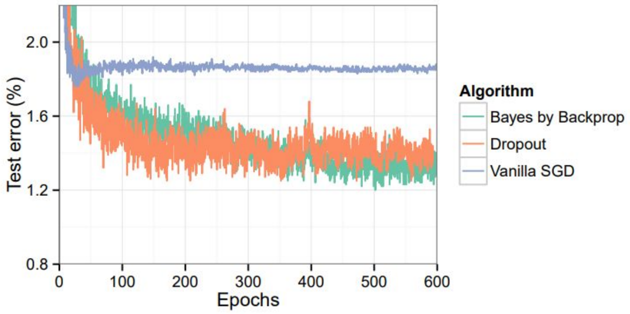


Figure 2. Test error on MNIST as training progresses.

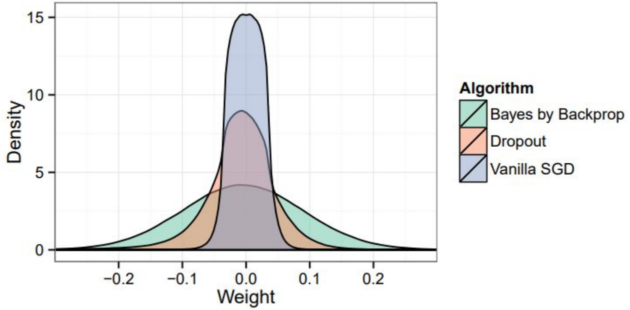


Figure 3. Histogram of the trained weights of the neural network, for Dropout, plain SGD, and samples from Bayes by Backprop.

MNIST: Compression Results

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%

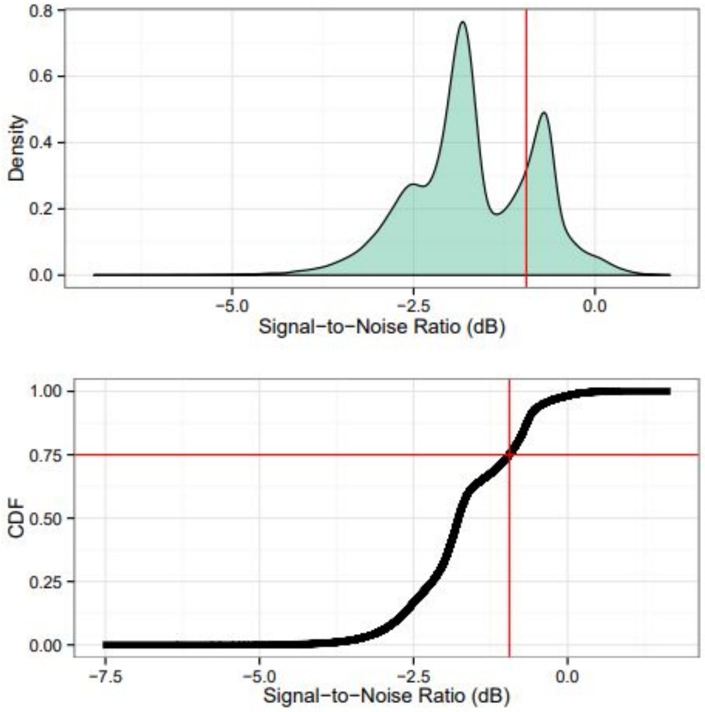


Figure 4. Density and CDF of the Signal-to-Noise ratio over all weights in the network. The red line denotes the 75% cut-off.

Second Results: Regression

The results are on the following slide

They show:

- Stochastic Gradient Descent and variations
- Two bayes networks (different priors)
- Varying Layers and weights/hyperparameters
- Data generated from sin combination with noise

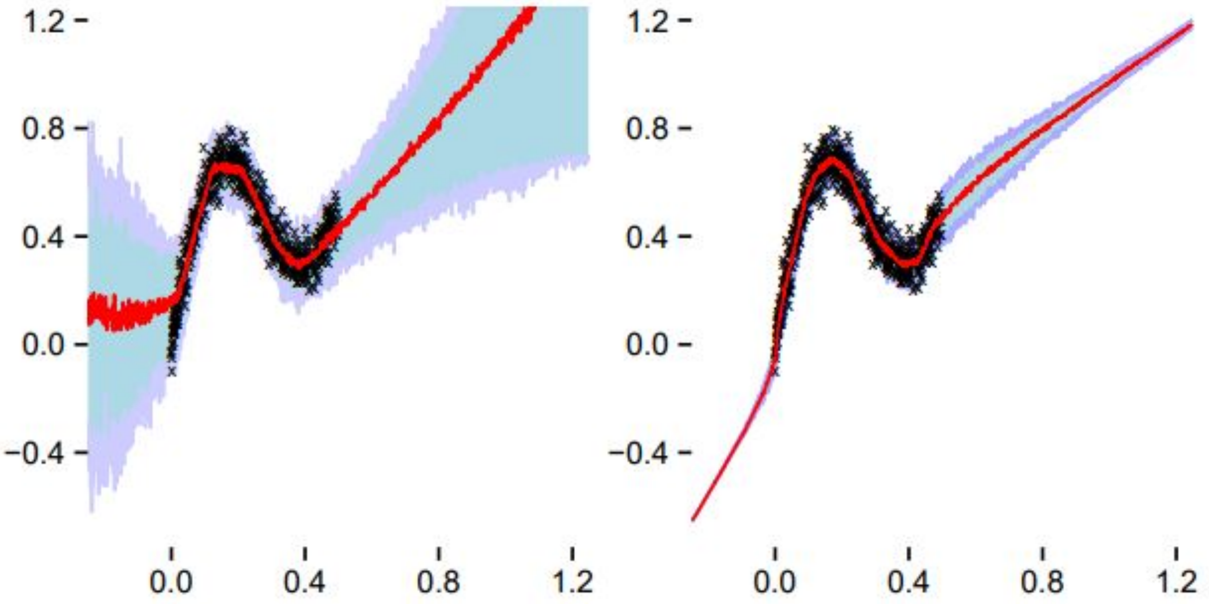


Figure 5. Regression of noisy data with interquatile ranges. Black crosses are training samples. Red lines are median predictions. Blue/purple region is interquartile range. Left: Bayes by Backprop neural network, Right: standard neural network.

Third Results: Mushroom Eating

The results are on the following slide

They show:

- Simple bandit problem with poisonous and nonpoisonous mushrooms
- Bayesian Neural Network with sample size = 2
- Epsilon-Greedy approach to bandit problem

Regret is defined as:
Difference between MAX and Model rewards

That is, someone who always chooses the right action versus someone who is trying to learn the best action

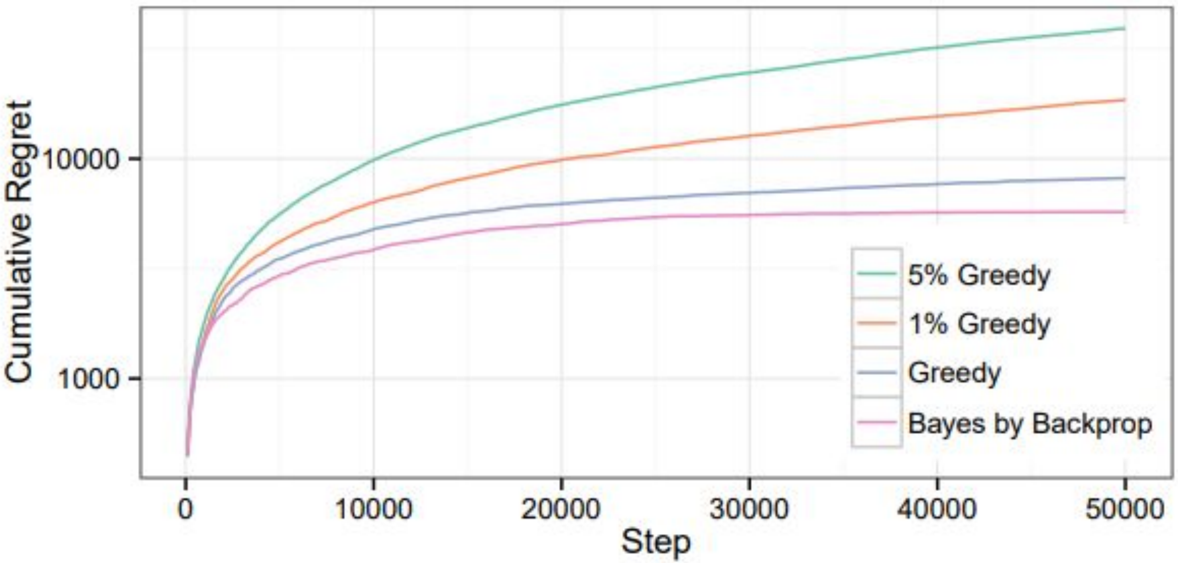


Figure 6. Comparison of cumulative regret of various agents on the mushroom bandit task, averaged over five runs. Lower is better.