Bayesian Neural Networks

Rhys Murray
Arian Pentza
Liyi Zhang

Talk Outline

1. Why should we care?
2. Brief review and derivation of background info
    a. Vanilla Neural Network review
    b. Bayes' Rule
3. Bayesian Neural Networks overview
4. Variational Inference
    a. KL Divergence and some key results
    b. ELBO objective
5. Pseudocode
6. Code demo + results
7. Future plans

Primary References

Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D.. (2015). Weight uncertainty in neural networks. *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37 (ICML'15).* JMLR.org, 1613–1622.

Blei, D., Kucukelbir, A., & McAuliffe, J. (2017). Variational Inference: A Review for Statisticians *Journal of the American Statistical Association, 112*(518), 859–877.

Kingma, D.P., & Welling, M. (2014). Auto-Encoding Variational Bayes. *CoRR, abs/1312.6114*.

Why are we interested?

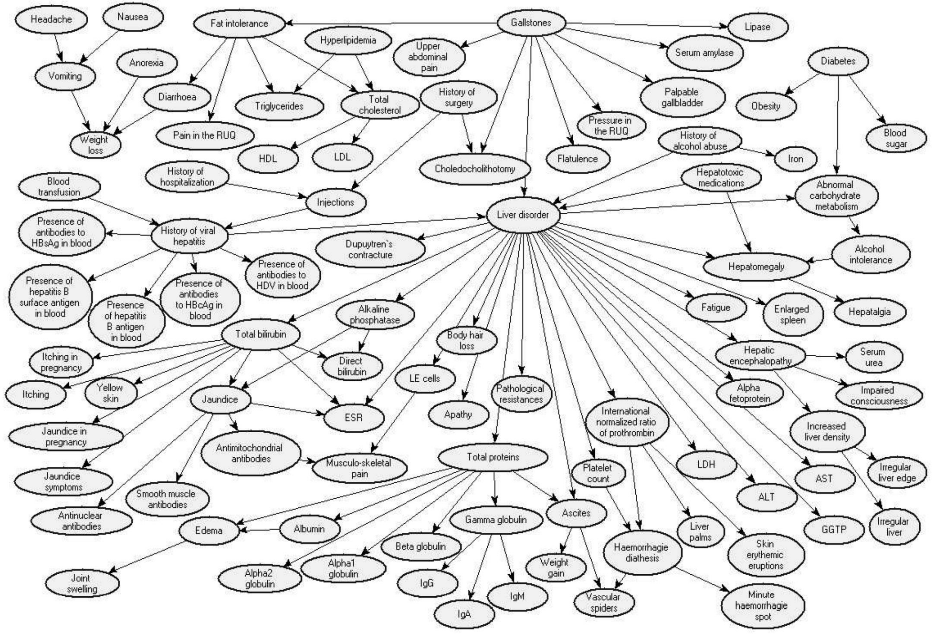
- Rhys
  - Interested in novel quantitative models with applications to finance
- Liyi
  - Been working on both Bayesian statistics and neural networks. Interested in seeing examples how they come together.
- Arian
  - Interested in Artificial Intelligence and what the future will hold with such powerful technology!

Why a Bayesian approach?

- Easier to estimate uncertainty than in standard DL
  - Standard DL weights are point estimates
  - Bayesian NN automatically encapsulates uncertainty in estimation
  - Models ‘know when they don’t know’
- A natural marginalization effect, or ensemble learning
- More robust to overfitting
  - ‘Occam’ factor
- Less data hungry than standard DL techniques
- Comparable predictive performance to standard without significant increase in computational complexity
  - However, training is sensitive to factors such as initialization

Potential Applications

- Good where:
  - Even one mistake is costly
  - Data collection is impractical
  - Data is noisy
  - One class is oversampled
- Fields of Interest:
  - Medical Diagnosis
  - Financial asset forecasting
  - Fraud detection
  - Computer vision



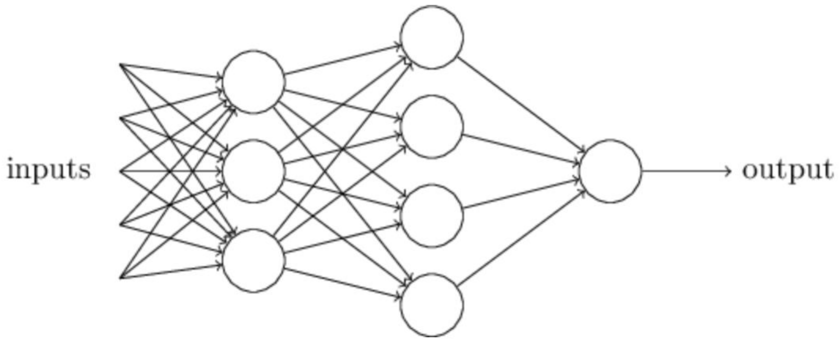
Onisko, Druzdel & Wasyluk (1999) - A Bayesian Network Model for Diagnosis of Liver Disorders

Current State of BNN

- Still not very prevalent in industry
- Improvements in variational inference
  - Decreases in computational complexity
- Significant research interest
  - DL community is interested in more robust, randomness-injected models
  - E.g. NIPS workshop: Bayesian Deep Learning
- Increasing community support
  - Open source software packages
  - Implemented in TensorFlow-Probability

Neural Networks Overview

- Simple mathematical model somewhat inspired by 'brain'
  - A series of neurons that output based off of its inputs, weights, and bias
  - An activation function to add non-linearity (ReLU, sigmoid, etc.)
- Each neuron has:
  - A set of inputs (x_1, x_2, ... x_n)
  - A set of weights for each input (w_1, w_2, ... w_n)
  - A bias (b)



Source: http://neuralnetworksanddeeplearning.com/chap1.html

Training Neural Networks

1. Initialize a random set of weights and biases
2. Generate a prediction
3. Define a loss function to measure prediction error
    a. Depends on the class of problem
4. Calculate gradient and step in the opposite direction (gradient descent)
    a. The size of this step is called the learning rate
5. Repeat until we find a minimum for the loss function

Backpropagation: Basic Outline

- Efficiently calculates gradient of loss function
- Want to calculate $\partial C/\partial w^l_{jk}$ and $\partial C/\partial b^l_j$
  - w^l_{jk}: weight for the connection in the kth neuron in the (l -1)th layer to the jth neuron in the lth layer
  - b^l_j: bias of the jth neuron in the lth layer
- Instead calculate $\partial C/\partial z^l_j$ and relate it to above quantities
  - z^l_j : weighted input to the jth neuron in the lth layer
  - This measures the ‘error’ of a given neuron
  - When $\partial C/\partial z^l_j$ is small, neuron is close to optimal

Backpropagation: Terminology

- z^l_j : weighted input to the jth neuron in the lth layer
- σ : activation function
- a^l_j: activation input to the jth neuron in the lth layer
  - Just $\sigma(z^l_j)$
- δ^l_j: the error $(\partial C/\partial z^l_j)$
- w^l_{jk}: weight for the connection in the kth neuron in the (l -1)th layer to the jth neuron in the lth layer

Backpropagation: Main Equations

$$\delta_j^L = \sum_k \frac{\partial C}{\partial a_k^L} \frac{\partial a_k^L}{\partial z_j^L}, = \sum_k \frac{\partial C}{\partial a_k^L} \sigma'(z_j^l)$$

Backpropagation: Main Equations

$$\delta_j^L = \sum_k \frac{\partial C}{\partial a_k^L} \frac{\partial a_k^L}{\partial z_j^L}, = \sum_k \frac{\partial C}{\partial a_k^L} \sigma'(z_j^l)$$

$$\delta_j^l = \sum_k w_{kj}^{l+1} \delta_k^{l+1} \sigma'(z_j^l).$$

Backpropagation: Main Equations

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \frac{\partial C}{\partial z^l_j} \quad \Longleftrightarrow \quad \frac{\partial C}{\partial w} = a_{\rm in} \delta_{\rm out},$$

Backpropagation: Main Equations

$$\frac{\partial C}{\partial w} = a_{\rm in}\delta_{\rm out}\,,$$

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j\,.$$

Backpropagation: Pseudocode

1. Set the initial input values
2. Feed the input values forward through all layers
3. Compute the error in the last layer (formula 1)
4. Calculate the error of previous layers from the last iteratively (formula 2)
    a. This is the ‘backpropagation’
5. Obtain the gradient from formulas 3-4

Loss Function

- Regression problems: MSE
- Classification problems: cross entropy loss
  - Given number of classes C, number of training examples N
  - Assume true labels y as one-hot C-vector: [0,...,0,1,0,...,0]
  - Apply softmax function to last layer of neural network so that prediction y_hat is a C-1 simplex
  - Loss function:

$$-\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c} \log(\hat{y_{i,c}})$$

Bayesian View

- Prior. Initial ‘belief’ about parameters: $p(\theta)$
- Likelihood. Data probability given parameters: $p(D|\theta)$
- Marginal. How is the data distributed: $p(D)$
- Posterior. Parameter distribution with knowledge of data: $p(\theta|D)$
- Average over all possible parameters to get a distribution for new inputs

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)\ p(\theta)}{p(\mathcal{D})}$$

$$p(y^\star|x^\star, \mathcal{D}) = \int p(y^\star|x^\star, \theta)\ p(\theta|\mathcal{D}) d\theta.$$

NN as Maximum Likelihood

- Can view a neural network as a likelihood model:
  - P(y|x, w)
  - Given some input x, we assign probabilities to each possible output y $\in$ Y using weights w
  - Y $=$ $\mathbb{R}$ for regression problems
  - Y $=$ set of classes for classification
- Standard NN provides point estimates for weights using MLE

$$\begin{aligned}\mathbf{w}^{\mathrm{MLE}} &= \arg\max_{\mathbf{w}} \log P(\mathcal{D}|\mathbf{w}) \\ &= \arg\max_{\mathbf{w}} \sum_i \log P(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w}).\end{aligned}$$

BNN Overview

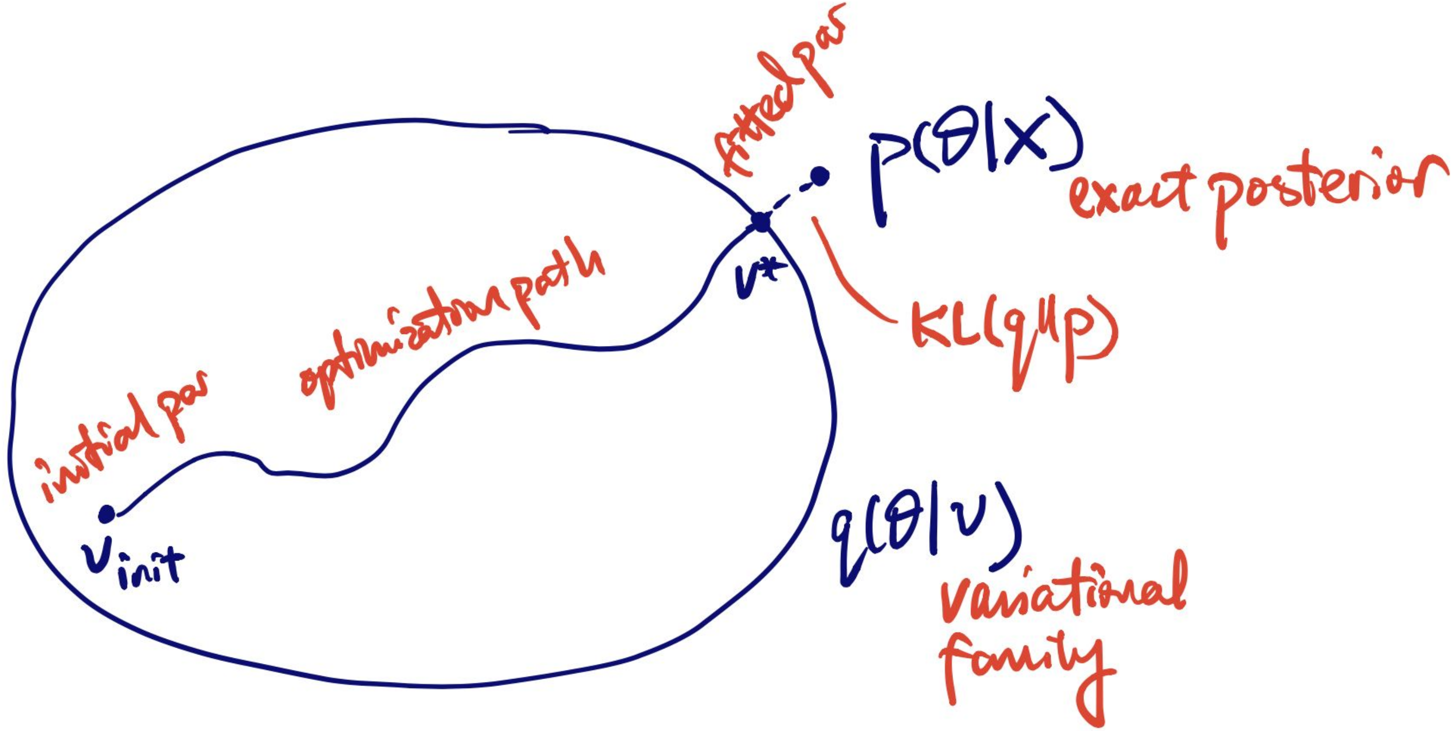
- NN as likelihood
- Prior distribution on the weights
- Find posterior for the weights given training data
- Cannot derive analytical forms for the posterior
- Instead:
    - Approximate inference with Variational Inference

Variational Inference

Turn inference problem to optimization problem

- Assume a family of distributions for the posterior
- Vary the parameters of the distribution to minimize distance between this distribution and the true posterior
    - Distance described by the KL-Divergence

Variational Inference



Kullback-Leibler (KL) divergence

- Measures how ‘different’ two probability distributions are
  - Bayesian: Measures the distance from approximate dist. to true posterior
- General entropy for probability distribution
  - KL is just the expected log difference
- Intuition: Amount of information lost when using Q instead of P
- Asymmetric so not a distance metric

$$h[f] = \mathrm{E}[-\ln(f(x))] = -\int_{\mathbb{X}} f(x) \ln(f(x))\, dx.$$

$$D_{KL}(P \| Q) = \mathbb{E}_{x \sim P}\left[\log \frac{P(X)}{Q(X)}\right]$$

Properties of KL Divergence

- Non-negative
  - The best we can do is lose 0 ‘bits’ of information
  - Can’t gain
- Only zero iff $P = Q$
  - Lose nothing if our approximation is exact
  - Losing nothing means our approximation is exact
- Minimization amounts to finding better approximations of P

Non-negativity of KL Divergence

Fact about ln: $\forall x > 0, ln(x) \leq x - 1$

Non-negativity of KL Divergence

Fact about ln: $\forall x > 0, ln(x) \leq x - 1$

$$-D_{KL}(p||q) = -\int p(x)\ln(\frac{p(x)}{q(x)})dx = \int p(x)\ln(\frac{q(x)}{p(x)})dx$$

Non-negativity of KL Divergence

Fact about ln: $\forall x > 0, ln(x) \leq x - 1$

$$-D_{KL}(p||q) = -\int p(x)\ln(\frac{p(x)}{q(x)})dx = \int p(x)\ln(\frac{q(x)}{p(x)})dx$$

$$\leq \int p(x)(\frac{q(x)}{p(x)} - 1)dx = \int q(x) - p(x)dx = 1 - 1 = 0$$

When does KL equal 0?

Jensen's Inequality: $\phi(\mathbb{E}[X]) \leq \mathbb{E}[\phi(X)]$ for convex ϕ

When does KL equal 0?

Jensen's Inequality: $\phi(\mathbb{E}[X]) \leq \mathbb{E}[\phi(X)]$ for convex ϕ

Equality holds iff ϕ is linear or p = q

When does KL equal 0?

Jensen's Inequality: $\phi(\mathbb{E}[X]) \leq \mathbb{E}[\phi(X)]$ for convex ϕ

Equality holds iff ϕ is linear or p = q

$$D_{KL}(p||q) = \int -p(x)\log(\frac{q(x)}{p(x)})dx \geq -\log(\int p(x)\frac{q(x)}{p(x)}) = -\log(1) = 0$$

When does KL equal 0?

Jensen's Inequality: $\phi(\mathbb{E}[X]) \leq \mathbb{E}[\phi(X)]$ for convex ϕ

Equality holds iff ϕ is linear or p = q

$$D_{KL}(p||q) = \int -p(x)\log(\frac{q(x)}{p(x)})dx \geq -\log(\int p(x)\frac{q(x)}{p(x)}) = -\log(1) = 0$$

If KL is 0, we have equality in Jensen's inequality and so

When does KL equal 0?

Jensen's Inequality: $\phi(\mathbb{E}[X]) \leq \mathbb{E}[\phi(X)]$ for convex ϕ

Equality holds iff ϕ is linear or p = q

$$D_{KL}(p||q) = \int -p(x)\log(\frac{q(x)}{p(x)})dx \geq -\log(\int p(x)\frac{q(x)}{p(x)}) = -\log(1) = 0$$

If KL is 0, we have equality in Jensen's inequality and so

$$\log(\frac{q(x)}{p(x)}) = 0 \iff p(x) = q(x)$$

ELBO Equation (Blei, et al. 2018)

- Directly minimizing KL is not computable
- z: parameters of our model (latent variables)
- x: observed data
- Dependent on p(x), our evidence
  - This generally has no closed form
  - Intractable to compute
- Instead maximize the ELBO equation
- Equal to negative KL + log p(x) which is constant with respect to q
- Maximizing this equation is equivalent to minimizing KL
- All quantities are effectively computable

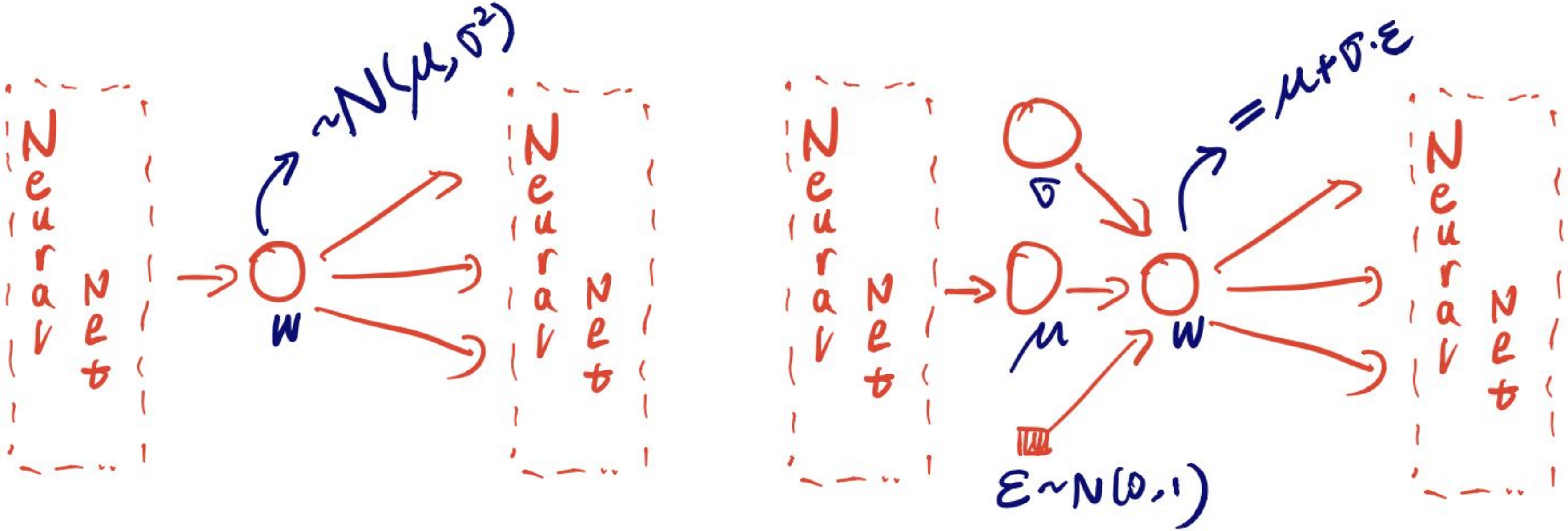
$$p(\mathbf{z} \mid \mathbf{x}) = \frac{p(\mathbf{z}, \mathbf{x})}{p(\mathbf{x})}. \qquad p(\mathbf{x}) = \int p(\mathbf{z}, \mathbf{x})\, d\mathbf{z}.$$

$$\mathrm{KL}(q(\mathbf{z}) \| p(\mathbf{z} \mid \mathbf{x})) = \mathbb{E}[\log q(\mathbf{z})] - \mathbb{E}[\log p(\mathbf{z}, \mathbf{x})] + \log p(\mathbf{x}).$$

$$\mathrm{ELBO}(q) = \mathbb{E}[\log p(\mathbf{z}, \mathbf{x})] - \mathbb{E}[\log q(\mathbf{z})].$$

Blei et al. (2017) - Variational Inference: A Review for Statisticians

The ‘Reparameterization Trick’



Undifferentiable

Differentiable

Kingma & Welling (2014) - Auto-encoding Variational Bayes
Rezende, et al. (2014) - Stochastic Backpropagation and Approximate Inference in Deep Generative Models

Algorithm 1: Bayesian Neural Network based on Variational Inference

0. Initialize variational parameters $\nu = (\mu, \rho)$. Choose step size α.

1. **for** i = *1 to N* **do**

    a. Sample from standard normal

    $$\epsilon_i \sim \mathcal{N}(0, I)$$

    b. Compute weights

    $$w_i = \mu + log(1 + \exp\rho) * \epsilon_i$$

end

2. Compute Monte Carlo loss

$$\mathcal{L} = \sum_{i=1}^{N} \log(q(w_i|\nu)) - \log(p(w_i)) - \log(P(\mathcal{D}|w_i))$$

3. Update variational parameters

$$\mu = \mu - \alpha\nabla_\mu\mathcal{L}$$

$$\rho = \rho - \alpha\nabla_\rho\mathcal{L}$$

Results: Test Accuracy

Bayes-1200-1200-10	Bayes-100-100-10	FC-1200-1200-10	FC-100-100-10
98.18%	97.90%	98.15%	97.74%

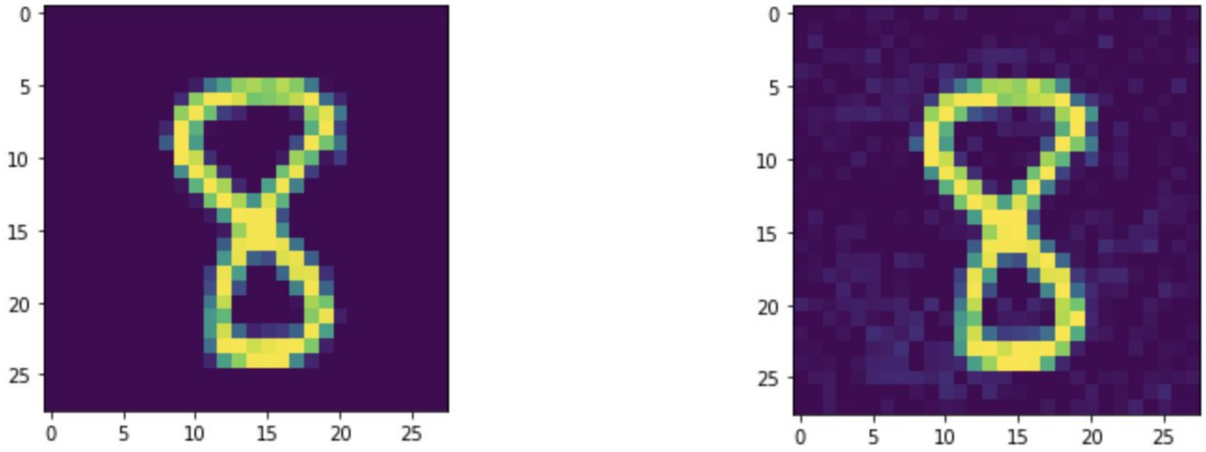
Test accuracies of Bayesian and (non-Bayesian) feed forward neural networks on the MNIST dataset.
x-x-10 refers to a network with two hidden layers, each of x hidden units.

Results: Model Pruning

Removed 0%	Removed 50%	Removed 75%	Removed 95%	Removed 98%
98.18%	98.18%	98.17%	97.89%	96.72%
98.15%	97.53%	92.87%	47.55%	38.20%

Test accuracies with different proportions of weights set to 0.
First row: Bayesian neural net (1200-1200-10). Second row: vanilla neural net of the same structure.

Results: Input Perturbation



Adversarially perturb input data using FC-1200-1200-10. Left: the network predicts as 8. Right: the network predicts as 3.

Szegedy et al. (2014) - Intriguing properties of neural networks

Results: Input Perturbation

Bayes-100-100-10	FC-1200-1200-10-ReLU	FC-100-100-10-ReLU	FC-100-100-10-Sigmoid
96.33% (97.79%)	0.00% (98.12%)	83.49% (97.65%)	89.91% (97.73%)
FC-10	FC-10 ($\lambda = 1$)	FC-10 ($\lambda = 1e - 2$)	FC-10 ($\lambda = 1e - 4$)
53.21% (93.01%)	82.57% (88.92%)	75.23% (92.71%)	52.29% (93.04%)

Accuracy of Bayesian and non-Bayesian neural networks on the perturbed dataset. In parenthesis is their accuracies on the original validation set.

Future Plans

Liyi Zhang

- Applying to graduate programs (mostly Stats/CS PhD); planning to work more on deep probabilistic modeling and approximate inference

Rhys Murray

- S&T right after graduation, planning to go back to a graduate program in Stats or pure math

Arian Pentza

- Planning to head into the web-dev industry, so if any of you guys need help making a website just hit me up

References

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, & Daan Wierstra. 2015. Weight uncertainty in neural networks. *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37 (ICML'15).* JMLR.org, 1613–1622.

Blei, D., Kucukelbir, A., & McAuliffe, J. (2017). Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association, 112*(518), 859–877.

Kingma, D.P., & Welling, M. (2014). Auto-Encoding Variational Bayes. *CoRR, abs/1312.6114*.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I.J., & Fergus, R. (2014). Intriguing properties of neural networks. *CoRR, abs/1312.6199*.

Ranganath, R., Gerrish, S. & Blei, D.. (2014). Black Box Variational Inference. *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, in PMLR* 33:814-822

Rezende, D.J., Mohamed, S., & Wierstra, D. (2014). Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *ICML*.

Wilson, A. (2020). The Case for Bayesian Deep Learning. *ArXiv, abs/2001.10995*.

Shridhar, K., Laumann, F., Maurin, A.L., Olsen, M., & Liwicki, M. (2018). Bayesian Convolutional Neural Networks with Variational Inference.

Gelman, A., Carlin, J., Stern, H., Dunson, D., Vehtari, A., & Rubin, D. (2014). Bayesian Data Analysis.

Q & A