Data Augmentation for Bayesian Deep Learning

Yuexi Wang,

Joint work with *Nick Polson and Vadim Sokolov*

Booth School of Business
University of Chicago

June 27th, ISBA 2022

Bayesian Deep Learning

Empirical success of deep learning amply illustrated.

Theoretical developments

- approximability
  - Why and when neural networks generalize well?
  - When do deep networks out-perform shallow ones?
  - Which activation functions and with how many layers?
- rate of convergence, including both frequentist and Bayesian point of view

However, training deep learners is still challenging

- high dimensional search space
- non-convex objective function

Issues like local traps, miscalibration and overfitting.

We provide a *Bayesian alternative* that could help.

What is Deep Learning (DL)?

- DL reconstructs high-dimensional input-output mappings.
- Given training data, *inputs* $\boldsymbol{x}_i \in \mathbb{R}^p$ and *outputs* $y_i = f(\boldsymbol{x}_i)$ for $1 \leq i \leq n$, *learn* $f_{\hat{\boldsymbol{\theta}}}$ such that

$$f_{\hat{\boldsymbol{\theta}}}(\boldsymbol{x}) \approx f(\boldsymbol{x}) \quad \text{for} \quad \boldsymbol{x} \notin \{\boldsymbol{x}_i\}_{i=1}^n.$$

- Training neural networks is then positioned as an *optimization problem* for finding values $\widehat{\theta} \in \mathbb{R}^T$ that minimize regularized empirical risk

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\theta} \sum_{i=1}^{n} \ell(y_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) + \phi(\boldsymbol{\theta}) \tag{1}$$

  where $\phi(\boldsymbol{\theta})$ is a penalty over the weights and offset parameters.

In practice, this is most often carried out with some form of stochastic gradient descent (SGD) (see e.g. Polson and Sokolov (2017) for an overview).

Deep Learning Mappings

A deep neural network $f_{\theta}(\boldsymbol{x})$ is an iterative mapping specified by *hierarchical layers of abstraction.*

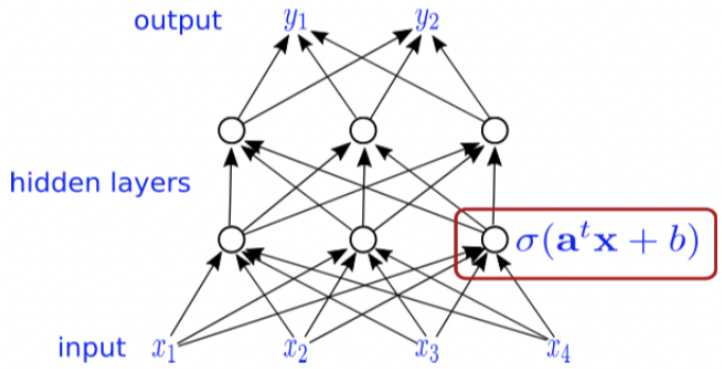


Figure 1: Representation as a direct graph of a network with two hidden layers $L = 2$ and width vector $\mathbf{p} = (4, 3, 3, 2)$.

With $L \in \mathbb{N}$ we denote the *number of hidden layers* and with $p_l \in \mathbb{N}$ the number of neurons at the l^{th} layer.

Setting $p_0 = p$ and $p_{L+1} = 1$, we denote with $\boldsymbol{p} = (p_0, \ldots, p_{L+1})' \in \mathbb{N}^{L+2}$ the vector of neuron counts for the entire network.

We denote the parameters with $\theta = \{(W_0, b_0), (W_1, b_1), \ldots, (W_L, b_L)\}$

Regularization DL

It is commonly agreed that generalizability of neural networks can be improved with **regularization.** (Goodfellow et al., 2016)

- Regularization through ReLU activation.
- Dropout regularization (Srivastava et al., 2014) samples from (and averages over) thinned networks obtained by *randomly* dropping out nodes together with their connections.
- While motivated as stochastic regularization, dropout can be regarded as ℓ_2 *regularization* obtained by margining out dropout noise (Wager, 2014).
- Conceptually related to Bayesian model averaging (ℓ_0 penalization).

How does Bayesian framework come into play?

Duality between Bayesian Simulation and Regularization

We can specify different amount of regularization $\{\lambda_l, \phi_l(\cdot)\}$ for each layer.

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f_\theta(x_i)) + \sum_{l=0}^{L} \lambda_l \phi_l(W_l, b_l). \tag{2}$$

Solving the problem is equivalent to find the *maximum a posteriori* (MAP) estimate to the following posterior

$$p(\boldsymbol{\theta} \mid \boldsymbol{y}) = p(\boldsymbol{y} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}) / p(\boldsymbol{y}),$$

where

$$p(\boldsymbol{y}|\boldsymbol{\theta}) \propto \exp\Big\{ - \sum_{i=1}^{n} \ell\big(y_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\big)\Big\}, \quad p(\boldsymbol{\theta}) \propto \exp\Big\{ - \sum_{l=0}^{L} \lambda_l \phi_l(W_l, b_l)\Big\}.$$

Here $p(\boldsymbol{\theta})$ can be interpreted as a prior probability distribution and the log-prior as the regularization penalty.

From Optimization to Sampling

By exploiting the duality, we can convert the composite functions

$$\begin{aligned} \boldsymbol{y} &= f_0(Z_1 W_0 + b_0), \\ Z_l &= f_l(Z_{l+1} W_l + b_l), \quad l = 1, \ldots, L, \\ Z_{L+1} &= \boldsymbol{x}, \end{aligned} \tag{3}$$

to their stochastic resemble as

$$\begin{aligned} \boldsymbol{y} \mid Z_1 &\sim p(\boldsymbol{y} \mid Z_1), \\ Z_l &\sim N(f_l(W_l Z_{l+1} + b_l), \tau_l^2), \quad l = 1, 2, \ldots, L, \\ Z_{L+1} &= \boldsymbol{x}. \end{aligned}$$

Now the hidden variables $Z = (Z_1, \ldots, Z_L)'$ can be viewed as data augmentation variables and be updated via sampling.

A Stochastic Top Layer

For the ease of computation, we only replace the top layer of DNNs with a stochastic layer.

We implement the solutions with iterative search.

1. DA-update for the top layer W_0, b_0

$$p(W_0, b_0 \mid Z_1^{(t)}, \boldsymbol{y}) \propto \exp\left\{-\frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f_{\boldsymbol{\theta}}(x_i) \mid \boldsymbol{B}^{(t)}) + \lambda_0 \phi_0(W_0, b_0)\right\}$$

2. SGD-update for the deep architecture $\boldsymbol{B}$

$$\boldsymbol{B}^{(t+1)} = \arg\min_{\boldsymbol{B}} \frac{1}{n}\sum_{i=1}^{n} \ell\big(y_i, f_{\boldsymbol{\theta}}(x_i) \mid (W_0, b_0)^{(t+1)}\big) + \sum_{l=1}^{L} \lambda_l \phi_l(W_l, b_l)$$

3. Sample $Z_1^{(t+1)}$ from a normal distribution $\mathcal{N}(\mu_z^{(t)}, \sigma_z^{(t)})$ (determined jointly by $\{\boldsymbol{\theta}^{(t)}, \boldsymbol{x}, \boldsymbol{y}\}$).

DA for Common Activation Functions

Data augmentation tricks allow us to express the likelihood as an expectation of a weighted L^2-norm or mixture of normals

$$\exp\Big\{-\ell\big(\boldsymbol{y}, f_{\boldsymbol{\theta}}(\boldsymbol{x})\big)\Big\} = \int_0^\infty \exp\Big(-Q\big(\boldsymbol{y} \mid f_{\boldsymbol{\theta}}(\boldsymbol{x}), \omega\big)\Big) p(\omega)\mathrm{d}\,\omega,$$

where $p(\omega)$ is the prior on the auxiliary variables ω and $Q(\cdot)$ is designed to be a quadratic form.

Commonly used activation functions such as ReLU, logit, lasso and check can be expressed in the form (Polson et al., 2013)

$$\exp(-\max(1-x,0)) = E_\omega\left\{\frac{1}{\sqrt{2\pi\omega}}\exp\Big(-\frac{1}{2\omega}(x-1-\omega)^2\Big)\right\}, \quad \text{where } \omega \sim \mathcal{GIG}(1,0,0),$$

$$\exp(-\log(1+e^x)) = E_\omega\left\{\exp(-\frac{1}{2}\omega x^2)\right\}, \quad \text{where } \omega \sim \mathcal{PG}(1,0),$$

$$\exp(-|x|) = E_\omega\left\{\frac{1}{\sqrt{2\pi\omega}}\exp\Big(-\frac{1}{2\omega}x^2\Big)\right\}, \quad \text{where } \omega \sim \mathcal{E}\big(\frac{1}{2}\big).$$

$\mathcal{GIG}$– the Generalized Inverse Gaussian distribution
$\mathcal{PG}$ – Pólya Gamma
$\mathcal{E}$ – exponential distribution.

Solutions to Data Augmentation

Two approaches to handle DA (Geyer, 1996)

- missing data methods like Expectation-Maximization (EM)
  - ☺ based on a probabilistic *approximation* of the objective function
  - ☹ less concerned with exploring Θ
- stochastic search methods like MCMC
  - ☺ visiting the entire range of Θ
  - ☹ less tied to the properties of the function

EM Algorithms

Consider constructing a surrogate optimization problem (Lange et al., 2000) which has the same solution to original problem

$$H(\boldsymbol{\theta}) = \mathrm{E}_{\boldsymbol{\omega}|\boldsymbol{\theta}}\Big[\exp\Big(-\frac{1}{n}\sum_{i=1}^{n} Q\big(y_i \mid f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{\omega}\big) - \sum_{l=0}^{L} \lambda_l \phi_l(W_l, b_l)\Big)\Big],$$

which is a concave function to be maximized.

Using Jensen's inequality we construct

$$G\big(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)}\big) = -\mathrm{E}_{\boldsymbol{\omega}|\boldsymbol{\theta}^{(t)}}\left[\frac{1}{n}\sum_{i=1}^{n} Q\big(y_i \mid f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{\omega}\big) + \sum_{l=0}^{L} \lambda_l \phi_l(W_l, b_l)\right],$$

and the minorization is satisfied as

$$\log H(\boldsymbol{\theta}) \geq G\big(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)}\big).$$

Maximizing $G\big(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)}\big)$ with respect to $\boldsymbol{\theta}$ drives $H(\boldsymbol{\theta})$ uphill.

EM in Action: Logistic Regression

Adopting the logistic regression model from Polson and Scott (2013), we focus on the penalization of W_0 as

$$\hat{W}_0 = \arg\min_{W_0} \left[\frac{1}{n} \sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i f_{\boldsymbol{B}}(\boldsymbol{x}_i) W_0\right)\right) + \phi(W_0 \mid \tau) \right],$$

The outcomes y_i are coded as ± 1, and τ is assumed fixed.

For likelihood function ℓ and regularization penalty ϕ, we assume

$$p(y_i \mid \sigma) \propto \int_0^\infty \frac{\sqrt{\omega}_i}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{\omega_i}{2\sigma^2}\left(y_i f_{\boldsymbol{B}}(\boldsymbol{x}_i) W_0 - \frac{1}{2\omega_i}\right)^2\right\} p(\omega_i) \mathrm{d}\,\omega_j,$$

$$p(W_0 \mid \tau) = \int_0^\infty \frac{\sqrt{\lambda}}{\sqrt{2\pi}\tau} \exp\left\{-\frac{\lambda}{2\tau^2}(W_0 - \mu_W - \kappa_W \lambda^{-1})^2\right\} p(\lambda) \mathrm{d}\,\lambda,$$

where λ follows a Pólya distribution prior $P(\lambda)$.

The deterministic updates of $\{W_0, \boldsymbol{\omega}, \boldsymbol{\lambda}\}$ generate a sequence of estimates that converges to a stationary point of posterior

MCMC and J-copies

A sequence can be simulated using Gibbs conditionals,

$$p\big(\boldsymbol{\theta}^{(t)} \mid \boldsymbol{\omega}^{(t)}, \boldsymbol{y}\big) \propto \exp\Big(- Q(\boldsymbol{y} \mid f_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{\omega}^{(t)})\Big) p(\boldsymbol{\theta}),$$

$$p\big(\boldsymbol{\omega}^{(t+1)} \mid \boldsymbol{\theta}^{(t)}, \boldsymbol{y}\big) \propto \exp\Big(- Q(\boldsymbol{y} \mid f_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{x}), \boldsymbol{\omega})\Big) p(\boldsymbol{\omega}).$$

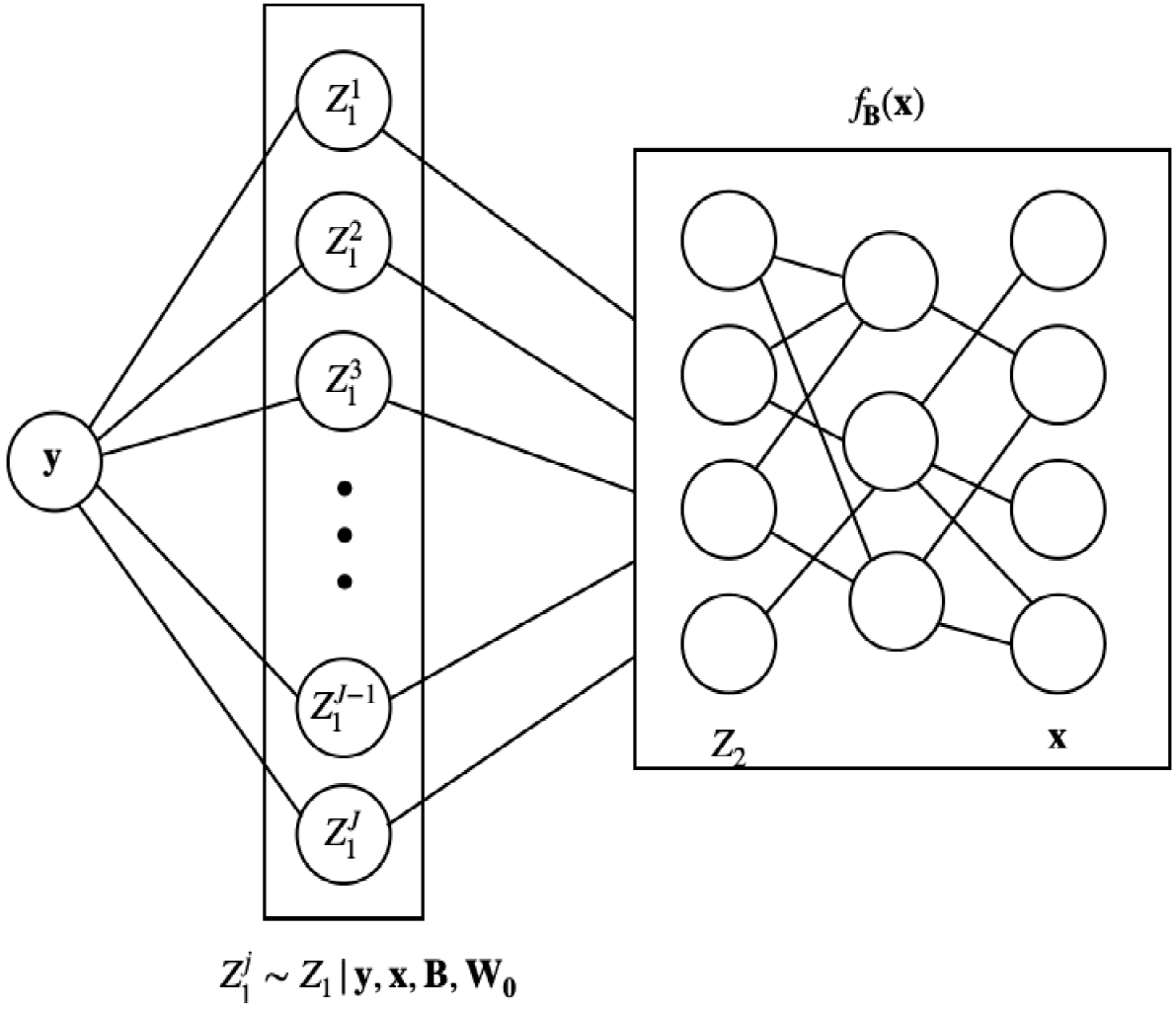
Inspired by the simulated annealing algorithm (Metropolis et al., 1953), we add a scaling factor J to help avoiding the trapping attraction of local maxima

$$\pi_J(\boldsymbol{\theta}) = \exp\left\{-Jf(\boldsymbol{\theta})\right\} / Z_J \quad \text{for} \quad \boldsymbol{\theta} \in \boldsymbol{\Theta} \tag{4}$$

where $Z_J = \int_{\boldsymbol{\Theta}} \exp\left\{-Jf(\boldsymbol{\theta})\right\} d\boldsymbol{\theta}$ is an appropriate normalizing constant.

MCMC and J-copies

To simulate the posterior mode without evaluating the likelihood directly (Jacquier et al., 2007), we sample J independent copies of hidden variable Z_1.



MCMC in Action: Gaussian Regression

We consider the regression model as

$$y_i = z_{1,i} W_0 + b_0 + \epsilon_{0,i}, \qquad \text{where } y_i \in (-\infty, \infty),\ \epsilon_{0,i} \overset{i.i.d}{\sim} \mathcal{N}(0, \tau_0^2),$$

$$z_{1,i} = f_{\boldsymbol{B}}(x_i) + \epsilon_{z,i}, \qquad \text{where } \epsilon_{z,i} \overset{i.i.d}{\sim} \mathcal{N}(0, \tau_z^2).$$

The posterior updates are given by

$$\hat{W}_0 = \mathrm{Cov}(Z_1, \boldsymbol{y})/\mathrm{Var}(Z_1),$$

$$\hat{b}_0 = \bar{\boldsymbol{y}} - W_0 \bar{Z}_1$$

$$p(Z_1 \mid \boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta}) = C_z \exp\left\{ -\frac{1}{2\tau_0^2} \left\| \boldsymbol{y} - Z_1 W_0 - b_0 \right\|^2 - \frac{1}{2\tau_z^2} \left\| Z_1 - f_{\boldsymbol{B}}(\boldsymbol{x}) \right\|^2 \right\},$$

where $\bar{\boldsymbol{y}} = \frac{1}{n} \sum_{i=1}^{n} y_i$ and C_z is a normalizing constant.

MCMC in Action: SVM

Polson and Scott (2011) and Mallick et al. (2005) write the support vector machine model as

$$\boldsymbol{y} = Z_1 W_0 + \boldsymbol{\lambda} + \sqrt{\boldsymbol{\lambda}}\epsilon_0, \text{ where } \boldsymbol{\lambda} \sim p(\boldsymbol{\lambda}),$$

where $p(\boldsymbol{\lambda})$ follows a flat uniform prior.

By incorporating the augmentation variable $\boldsymbol{\lambda}$, the ReLU deep learning model can be written as

$$y_i = z_{1,i} W_0 + \lambda_i + \sqrt{\lambda_i}\epsilon_{0,i}, \text{ where } y_i \in \{-1, 1\}, \epsilon_{0,i} \overset{i.i.d}{\sim} \mathcal{N}(0, \tau_0^2),$$
$$z_{1,i} = f_{\boldsymbol{B}}(\boldsymbol{x}_i) + \epsilon_{z,i}, \text{ where } \epsilon_{z,i} \overset{i.i.d}{\sim} \mathcal{N}(0, \tau_z^2).$$

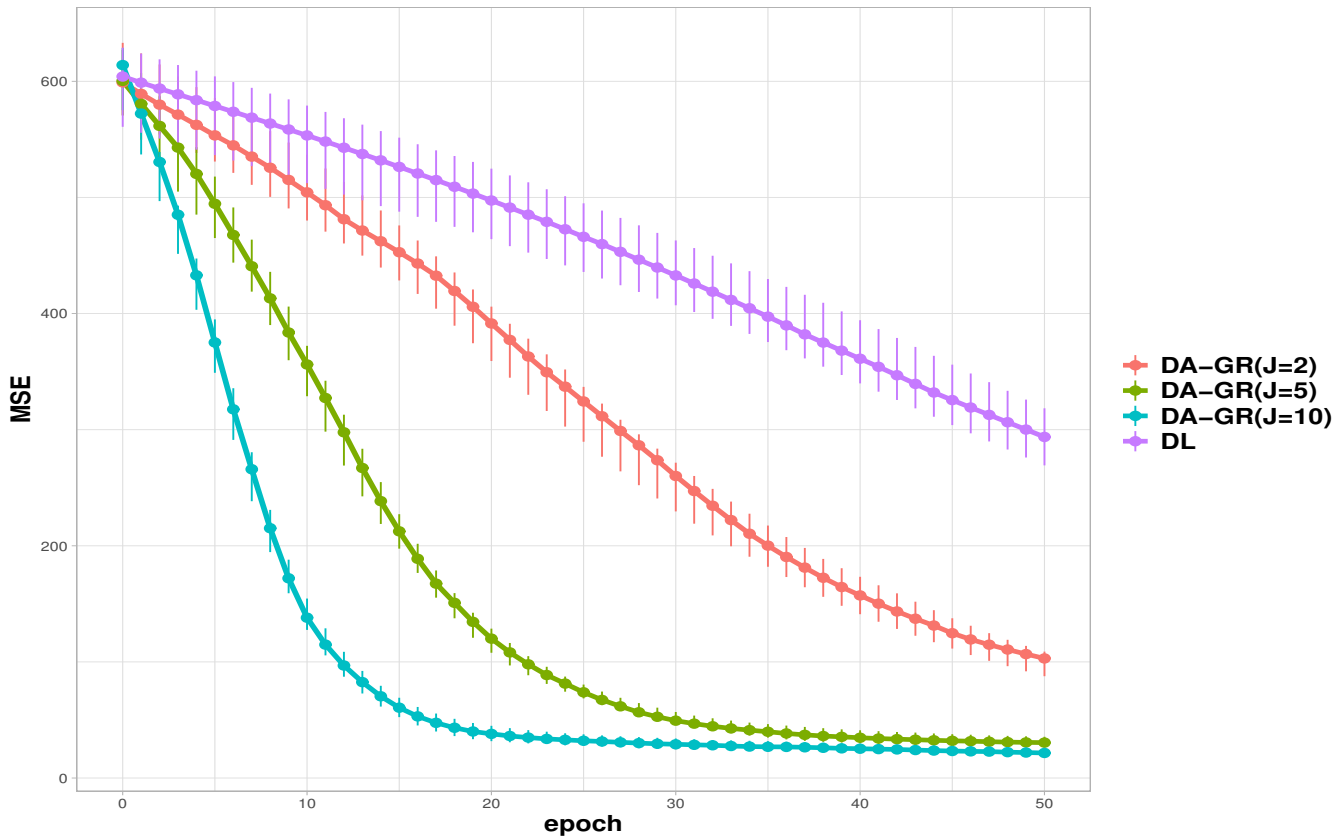
In order to generate the latent variables, we use conditional Gibbs sampling as

$$\lambda_i^{-1} \mid W_0, y_i, z_{1,i} \sim \mathcal{IG}(|1 - y_i z_{1,i} W_0|^{-1}, \tau_0^{-2})$$
$$W_0 \mid \boldsymbol{y}, Z_1, \lambda \sim \mathcal{N}(\mu_w, \sigma_w^2)$$
$$Z_1 \mid \boldsymbol{y}, \boldsymbol{x}, W_0, \boldsymbol{B} \sim \mathcal{N}(\mu_z, \sigma_z^2)$$

Empirical Analysis

Boston Housing Dataset

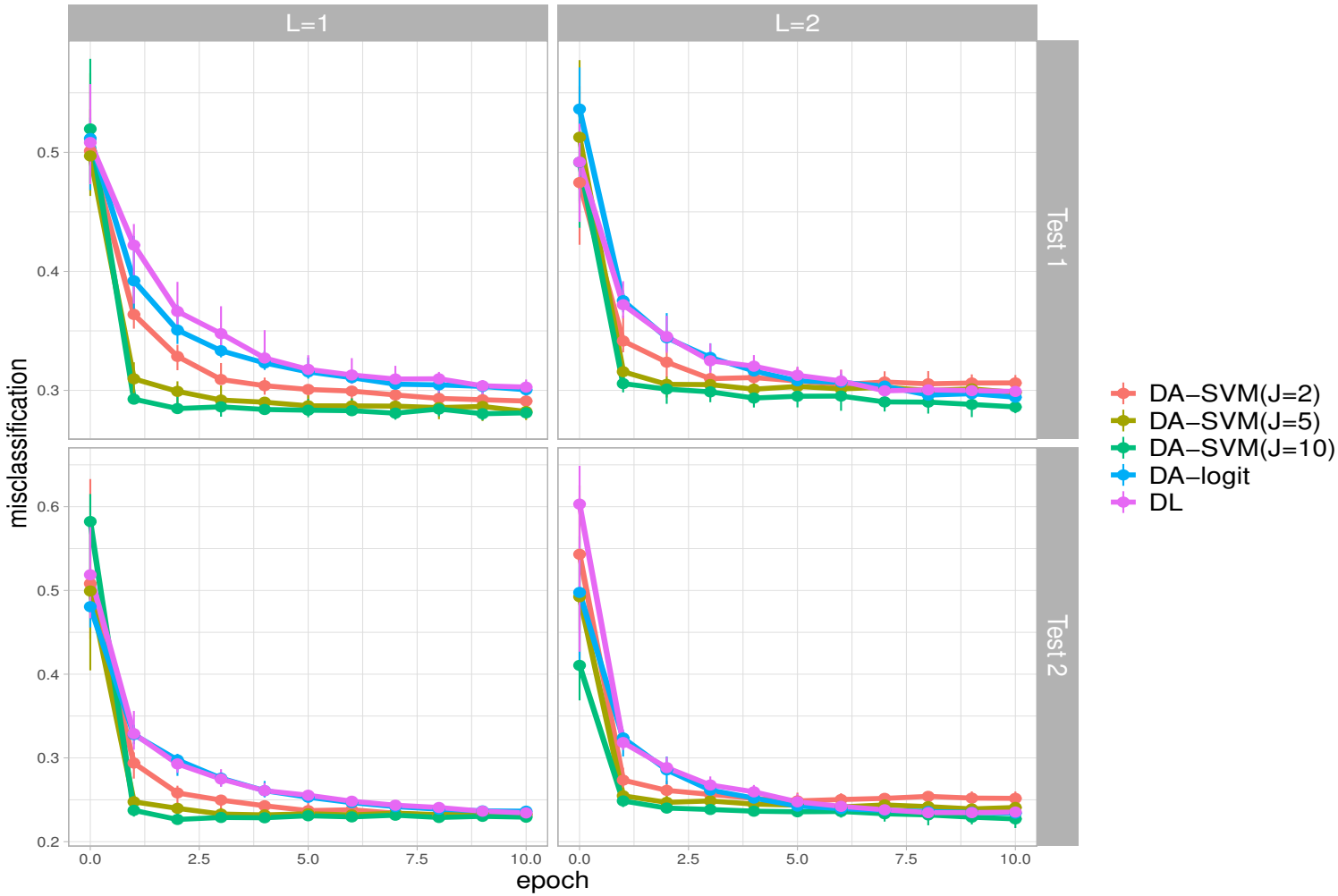
Boston Housing dataset[1] has $n = 506$ observations with 13 features. We adopt a one-layer ReLU networks with 64 units and set the dropout rate to be 0.5.



[1] https://archive.ics.uci.edu/ml/machine-learning-databases/housing/

Wine Quality Dataset

The Wine Quality Data Set [2] contains 4 898 observations with 11 features. Two types of binary classifications are considered.



[2] P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis, 'Wine Quality Data Set', UCI Machine Learning Repository.

Concluding Remarks

- We proposed a new approach to update the neural network hidden nodes via sampling.
- The objective function can be rewritten in weighted least squares form using data augmentation, which can be implemented at scale with accelerated linear algebra methods.
- We provide motivating examples for commonly used activation functions. Substantial performance gains are observed in empirical analysis.

Thank you!

Reference: Wang, Y., Polson, N. G., & Sokolov, V. O. (2019). Scalable Data Augmentation for Deep Learning.