Bayesian and Maximum Likelihood Neural Networks

Finn Årup Nielsen

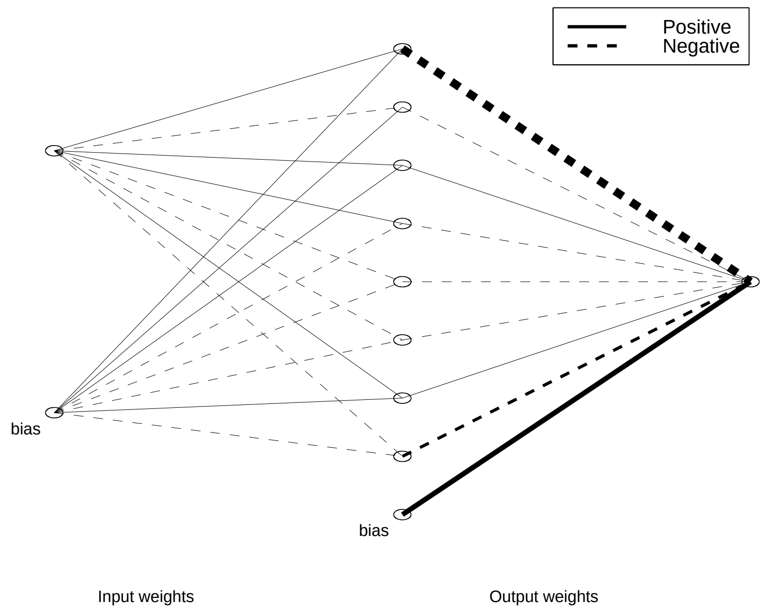
Section for Digital Signal Processing
Department of Mathematical Modelling
Technical University of Denmark
DK-2800 Lyngby, Denmark

Email: fn@imm.dtu.dk
WWW: http://www.imm.dtu.dk/~fn

OVERVIEW

- Artificial neural networks
- Maximum likelihood, MAP, MPL neural networks
- Bayesian neural networks
  - MCMC Bayesian neural networks
    - Hybrid Monte Carlo
- `lyngby` matlab toolbox ML/MLP/MAP neural network
- Radford Neal's "flexible Bayesian models" (`fbm`).
- Comparison between `fbm` and `lyngby`

NEURAL NETWORKS



"Forward equation" or "network equation" for two-layer feedforward neural network:

$$\mathbf{Y} = [\mathrm{tanh}\,([\mathbf{X}\ \mathbf{1}]\,\mathbf{V})\ \ \mathbf{1}]\,\mathbf{W} \tag{1}$$

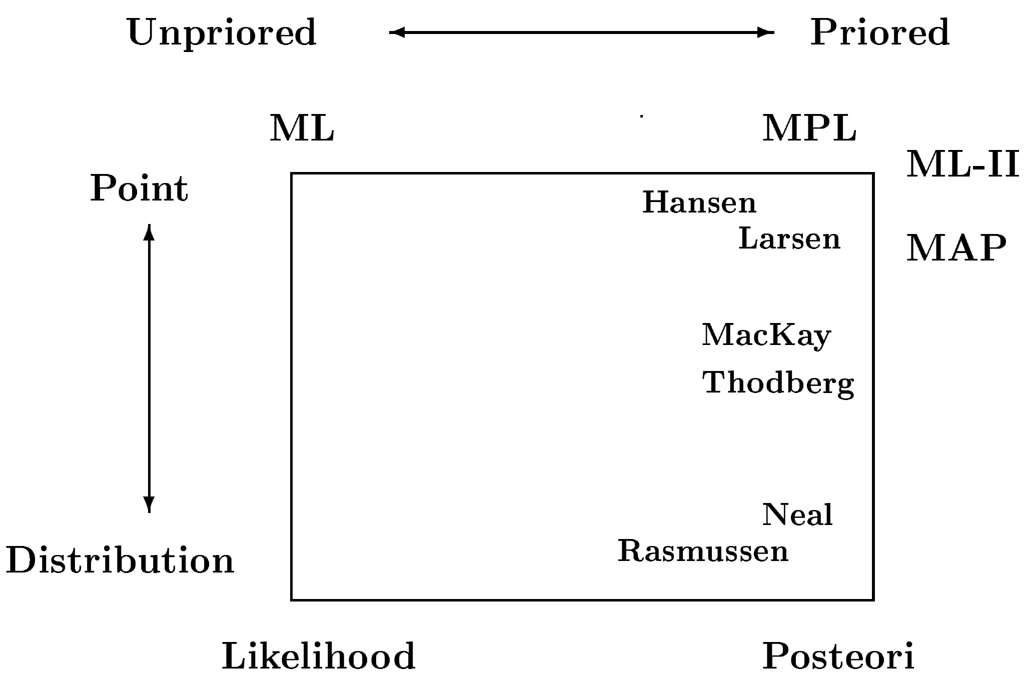
- $\mathbf{X}$: Input as a datamatrix
- $\mathbf{Y}$: Output
- $\mathbf{V}$: Input weights (parameters)
- $\mathbf{W}$: Output weights (parameters)
- $\mathbf{1}$: Bias units
- tanh: The nonlinearity: a sigmoidal function (hyperbolic tangent)

... NEURAL NETWORK

Neural networks uses:

- Regression, no modification of the outputs, linear output, Gaussian distribution $]-\infty;+\infty[$
- Binary (binary classification), tanh on output, binomial distribution. $]-1;+1[$
- Classification, softmax function on outputs [Bridle, 1990], multinomial distribution. $]0;+1[$
- Hybrid: regression and classification.

Types of estimation in neural network:



. . . NEURAL NETWORK

Modelling the regression output:

$\mathbf{y}$ is the output of the neural network — hopefully the "truth". ϵ is additive noise. $\mathbf{t}$ is the target, i.e. the empirical output:

$$\mathbf{t} = \mathbf{y} + \boldsymbol{\epsilon} \tag{2}$$

Gaussian additive noise:

$$p(\mathbf{t}|\mathbf{x};\mathbf{u}) = \frac{1}{\sqrt{(2\pi)^{n_o}|\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}|}} \exp\left(-\frac{1}{2}(\mathbf{y}-\mathbf{t})^{\mathsf{T}}\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}^{-1}(\mathbf{y}-\mathbf{t})\right) \tag{3}$$

$$p(t|\mathbf{x},\mathbf{u}) = \sqrt{\frac{\tau}{2\pi}} \exp\left(-\frac{\tau}{2}(y-t)^2\right) \tag{4}$$

Typical prior / penalization:

$$p(\mathbf{u}) = \frac{1}{\sqrt{(2\pi)^{n_u}|\boldsymbol{\Sigma}_{\mathbf{u}}|}} \exp\left(-\frac{1}{2}\mathbf{u}^{\mathsf{T}}\boldsymbol{\Sigma}_{\mathbf{u}}^{-1}\mathbf{u}\right) \tag{5}$$

ML OR MPL/MAP NEURAL NETWORK

Maximum likelihood neural network (ML-NN)

- Cost function: $\propto -\log p(\mathbf{t}|\mathbf{x};\mathbf{u})$
- Calculate derivatives: First and second order
- Optimize network with: gradient descent ("back-propagation": First order derivative only depend on values in the proceeding layer), pseudo-Gauss-Newton, Gauss-Newton, Levenberg-Marquardt

Maximum penalized likelihood / Maximum a posteriori neural network

- Cost function: $\propto -\log p(\mathbf{t}|\mathbf{x};\mathbf{u}) - \log p(\mathbf{u})$
- Calculate derivatives: First and second order
- Optimize network
- Optimize the hyperparameters in $\log p(\mathbf{u})$ By cross-validation or analytic estimates (of the generalization error), e.g., Moody's GPE [Moody, 1992] (Akaike's FPE for regularized models)
  - Gaussian prior $\rightarrow$ weight decay
  - Individual priors $\{0, \inf\}$ ("implicit prior" $p(\mathbf{u}) \propto \exp(-\log(|\mathbf{u}|))$ [Sporring, 1997]) $\rightarrow$ pruning

BAYESIAN NEURAL NETWORK

- Gaussian approximations to posterior
  - David J. C. MacKay, Hans Henrik Thodberg [Thodberg, 1996]
  - "The evidence framework" [MacKay, 1992]
  - Type II maximum likelihood (ML-II) [Berger, 1985].
- Markov chain Monte Carlo
  - Radford Neal [Neal, 1996], Carl Edward Rasmussen [Rasmussen, 1996]
  - Integration over weights and hyperparameters. Gibbs sampling for hyperparameters
- "Sequential Monte Carlo" for on-line learning (?) [de Freitas and Niranjan, 1998].

MCMC IN BAYESIAN NEURAL NETWORKS

Posterior / costfunction

- Mirror modes: $2^{(n_h)} n_h!$
  - Sign switch between hidden and output layer weights
  - Permutation of hidden units
- Complex nonlinear: Hyperbolic tangent

"Optimization" / Integration

- Gibbs sampling not possible for weights because the posterior is complex
- Rejection sampling not good: Weight space is high dimensional
- MCMC possible: However, "stiff" valleys in posteriori makes it slow.
- MCMC with momentum ("Hybrid Monte Carlo")
- Simulated annealing (tempering) on MCMC parameters: no speed-up [Neal, 1996, page 65]
- Ordered overrelaxation [Neal, 1995] [Neal, 1998b]

MCMC WITH MOMENTUM

Reviewed in [Neal, 1993, Chapter 5] and [Neal, 1996, section 3.1]

Stochastic dynamics + Metropolis = Hybrid Monte Carlo [Duane et al., 1987]

Stochastic dynamics

- $E(\mathbf{u})$ = potential energy $\propto \log(\text{posterior}(\mathbf{u}))$
- $K(\mathbf{z})$ = kinetic energy, fictious momentum associated with $\mathbf{u}$.
- $H(\mathbf{u}, \mathbf{z}) = E(\mathbf{u}) + K(\mathbf{z})$ = Total energy = Hamiltonian
- $p(\mathbf{u}, \mathbf{z}) = 1/Z_H \exp(-H) = \text{posterior}(\mathbf{u})\, p(\mathbf{z})$, "canonical distribution"
- Sampling in a one dimensional path in phase space $(\mathbf{u}, \mathbf{z})$

. . . MCMC WITH MOMENTUM

Leapfrog

Stochastic dynamics + discretization = "leapfrog"

1. Half step for kinetic energy
2. Full step for potential energy
3. Half step for kinetic energy

Hybrid Monte Carlo

1. Stochastic transition, sample new momenta
2. Dynamical transition
   (a) Perform L leapfrog steps of size ϵ
   (b) Negate the momenta
   $$(\mathbf{u}^*, \mathbf{z}^*) = (\hat{\mathbf{u}}(\epsilon L), -\hat{\mathbf{z}}(\epsilon L)) \tag{6}$$
   (c) "Metropolis reject". Eliminate the bias from the discretization approximation.
   $$\min\left[1, \exp(-(H(\mathbf{u}^*, \mathbf{z}^*) - H(\mathbf{u}, \mathbf{z}))\right] \tag{7}$$

HYBRID MONTE CARLO VARIANTS

Hybrid Monte Carlo variants reviewed [Neal, 1993, section 5.2] and [Neal, 1996, section 3.5]

- Non-leapfrog discretization [Creutz and Gocksch, 1989]
- Partial gradients [Neal, 1996, section 3.5.1].
  - Each leapfrog step uses only the gradient computed from a part on the data set.
  - Should be faster to compute. However, lower acceptance rate, thus no advantage.
- Windows [Neal, 1994] [Neal, 1996, section 3.5.2]
- Persistence [Horowitz, 1991] [Neal, 1996, section 3.5.3].
  - Partial remembering the momenta across Metropolis steps (and using only a single leapfrog step).
  - Used in [Rasmussen, 1996]

DSP IMM DTU

LYNGBY

- `lyngby`: Matlab toolbox for mainly functional neuroimaging
- Available at `http://hendrix.imm.dtu.dk/software`
- Contains maximum likelihood neural network for regression, classification and binary outputs
- Optimization with: gradient descent, pseudo-Gauss-Newton, Levenberg-Marqaurdt, ...
- Pruning, weight decay (Grid search).
- Simple (single-blocked) cross validation

```
>> [V, W, Evalidation] = ...
    lyngby_nn_qmain(Xtrain, Ttrain, ...
      'GenOptim', 'Pruning', ...
      'Validation', 'Singleblocked', ...
      'HiddenUnits', 8, ...
      'Reg', 0.001);

>> Ytest = lyngby_nn_qforward(Xtest, V, W);
>> Etest = lyngby_nn_qerror(Ttest, Ytest);
```

RADFORD NEAL'S "FLEXIBLE BAYESIAN MODELS"

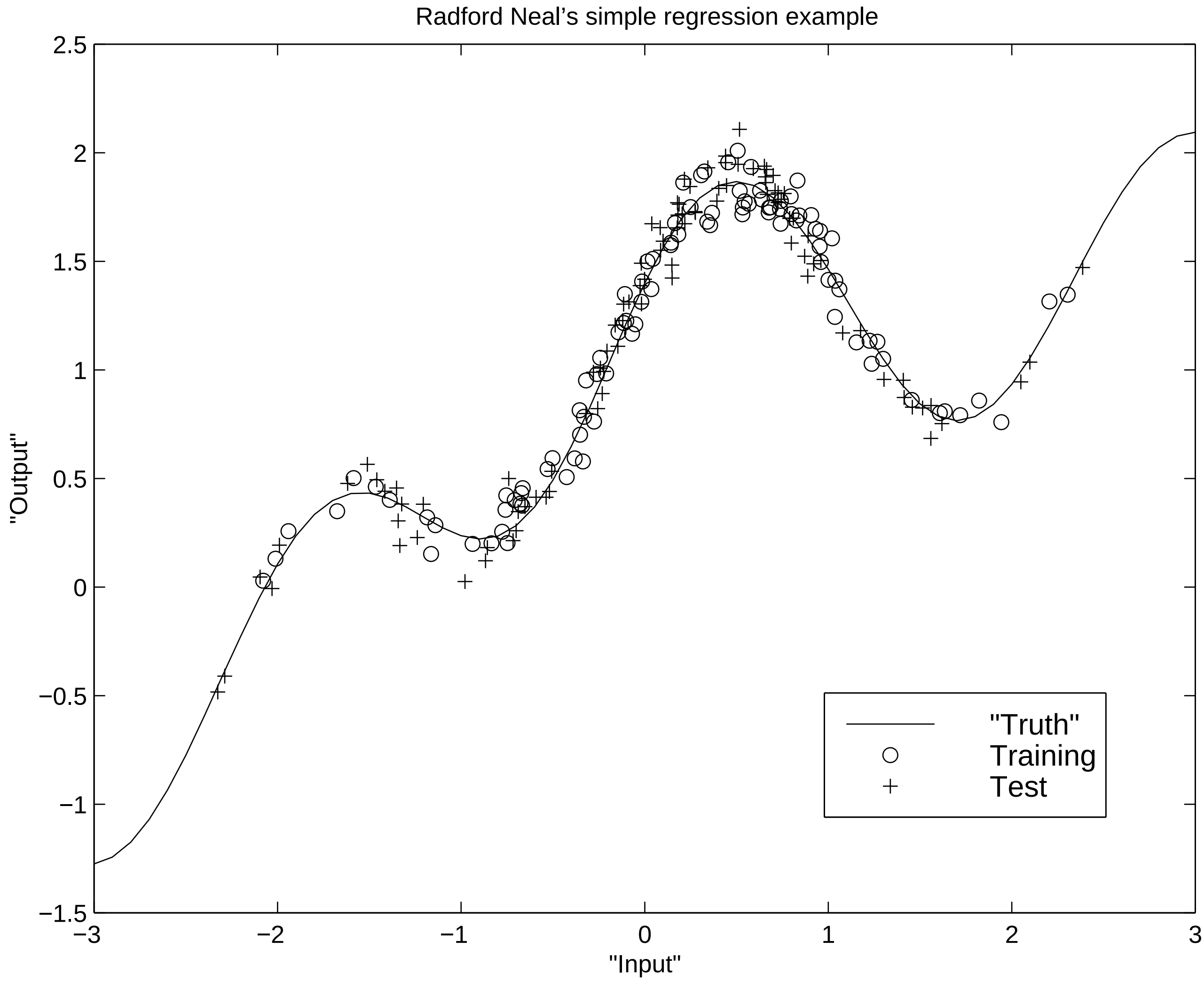
- `fbm`: "Flexible Bayesian models" — Unix programs with source [Neal, 1998a].
- Available at `http://www.cs.toronto.ca/~radford`
- Contains Bayesian neural network, Gaussian process, mixture models
- More general feedforward architecture: Only forward network is required — no derivatives.
- Simple priors: Gaussians and Gamma densities.
- Stochastic integration with: Gibbs sampling (`heatbath`), hybrid Monte Carlo (`hybrid`)

```
> net-spec rlog.net 1 8 1 / - 0.05:0.5 0.05 - x0.05:0.5 - 100
> model-spec rlog.net real 0.05:0.5
> data-spec rlog.net 1 1 / rdata@1:100 . rdata@101:200 .

> net-gen rlog.net fix 0.5
> mc-spec rlog.net repeat 10 sample-noise heatbath hybrid 100:10 0.2
> net-mc rlog.net 1

> mc-spec rlog.net sample-sigmas heatbath hybrid 1000:10 0.4
> net-mc rlog.net 100
```

RADFORD NEAL'S SIMPLE REGRESSION EXAMPLE



$$y = 0.3 + 0.4x + 0.5\sin(2.7x) + 1.1/(1 + x^2); \qquad (8)$$

- From Neal's software [Neal, 1998a]
- 100 training examples, 100 test examples
- Noise on data: $\sigma_\epsilon = 0.1$

PRIORING EXAMPLES ..., (1/2)

Bayesian neural network

Parameter	Parameter	Parameterprior	Hyperparameterprior
Input weights	$\mathbf{v}$	$\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \tau_{\mathbf{v}}\mathbf{I})$	$\tau_{\mathbf{v}} \sim \mathcal{G}(400, 0.5)$
Output weights	$\mathbf{w}$	$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \tau_{\mathbf{w}}\mathbf{I})$	$\tau_{\mathbf{w}} \sim \mathcal{G}(400, 0.5)$
Input bias	$\mathbf{v_0}$	$\mathbf{v_0} \sim \mathcal{N}(\mathbf{0}, \tau_{\mathbf{v_0}}\mathbf{I})$	$\tau_{\mathbf{v_0}} \sim \mathcal{G}(400, 0.5)$
Output bias	w_0	$w_0 \sim \mathcal{N}(0, 100^2)$	
Output noise	ϵ	$\epsilon \sim \mathcal{N}(0, \tau_\epsilon)$	$\tau_\epsilon \sim \mathcal{G}(400, 0.5)$

Table 1: Example on prioring from [Neal, 1998a] in the simple regression example. (See the file `doc/Ex-netgp-r.html`)

Point estimating (MPL) neural network

Parameter	Parameter	Parameterprior	Hyperparameterprior
All weights	$\mathbf{u}$	$\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \tau_{\mathbf{u}}\mathbf{I}) + \mathcal{N}(\mathbf{0}, \boldsymbol{\tau}_{\mathbf{u}}\mathbf{I})$	$\tau_{\mathbf{u}} \in \{\ldots 0.001\ 0.01\ 0.1 \ldots\}$, $\boldsymbol{\tau}_{\mathbf{u}} \in \{0, \infty\}^{n_u}$
Input bias	$\mathbf{v_0}$	$\mathbf{v_0} \sim \mathcal{F}(\boldsymbol{\delta})$	
Output bias	w_0	$w_0 \sim \mathcal{F}(\delta)$	
Output noise	ϵ	$\epsilon \sim \mathcal{N}(0, 1)$	

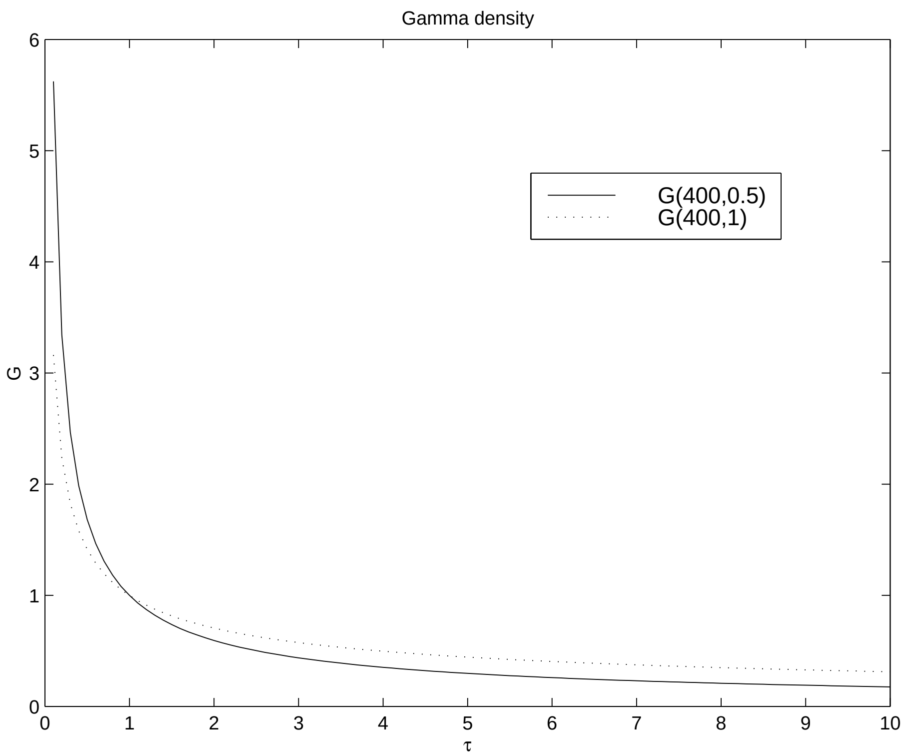
Table 2: Example on prioring in a MPL-NN with the `lyngby` toolbox. - Pruning and common weight decay parameter on input and output weights with grid search on weight decay parameter. - No prior on input and output biases. - No modelling of output noise (assumed Gaussian)

. . . PRIORING EXAMPLES, (2/2)

Parameter	Parameterprior	Hyperparameterprior	Hyper^{2}parameterprior
$\mathbf{v}$	$\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \tau_{\mathbf{v}}\mathbf{I})$	$\tau_v \sim \mathcal{G}(\mu_{\tau_{\mathbf{v}}}, 0.5)$	$\mu_{\tau_{\mathbf{v}}} \sim \mathcal{G}(400, 1)$
$\mathbf{w}$	$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \tau_{\mathbf{w}}\mathbf{I})$	$\tau_w \sim \mathcal{G}(\mu_{\tau_{\mathbf{w}}}, 0.5)$	$\mu_{\tau_{\mathbf{w}}} \sim \mathcal{G}(400, 1)$
$\mathbf{v_0}$	$\mathbf{v_0} \sim \mathcal{N}(\mathbf{0}, \tau_{\mathbf{v}_0}\mathbf{I})$	$\tau_{\mathbf{v}_0} \sim \mathcal{G}(400, 0.5)$	
w_0	$w_0 \sim \mathcal{N}(0, 1000^2)$		

Table 3: Example on prioring from [Rasmussen, 1996].

$$\tau \sim \mathcal{G}(\mu, \alpha) \propto \tau^{\alpha/2-1} \exp(-\tau\alpha/2\mu) \qquad (9)$$



WEIGHTS EVOLUTION FOR MCMC-BNN

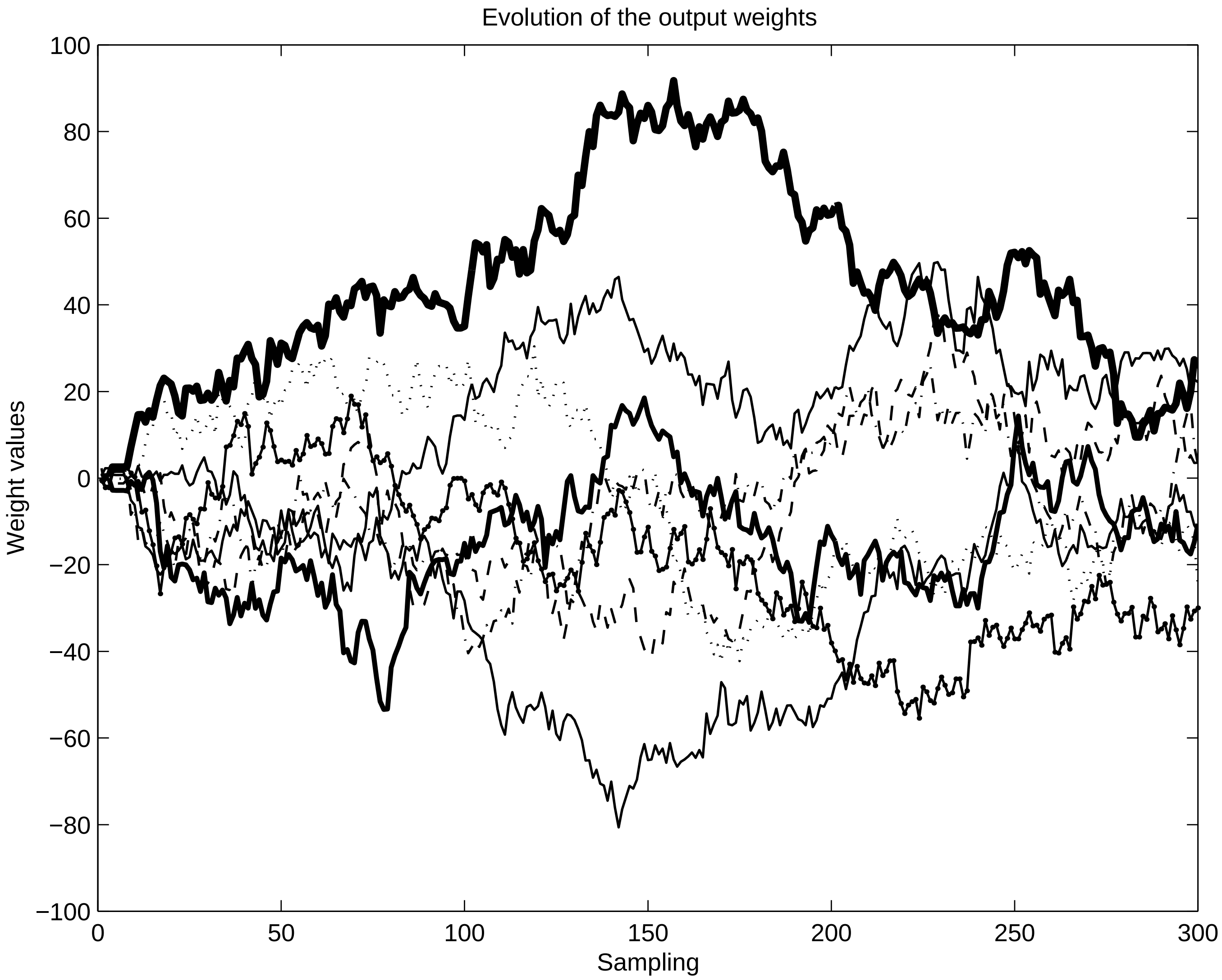


Figure 1: Sampling path of output weights for Neal's simple regression example for a 1–8–1 MCMC-BNN.

- Weights do not convergence to a single value
- After 300 steps: not all modes explored. E.g. the thick weight has not been negative

WEIGHT AUTOCORRELATION COEFFICIENT

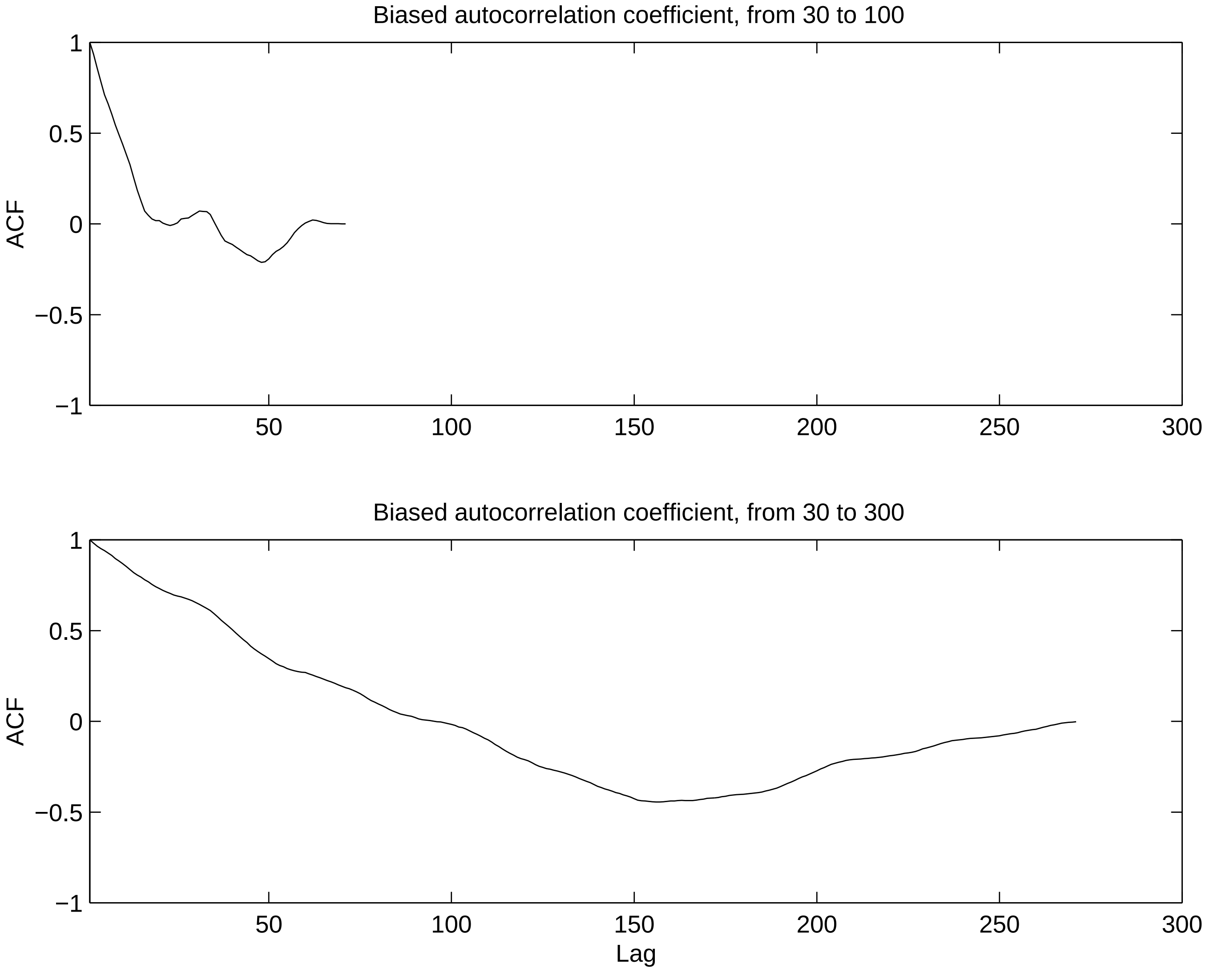


Figure 2: Biased autocorrelation coefficient of one of the weights in Neal's simple regression example.

- The autocorrelation continues to decrease (as the weights explore new space)
- Convergence might be of minor importance: The neural network already predicts quite well with samples from 30 to 100.

WEIGHTS AND HYPERPARAMETERS

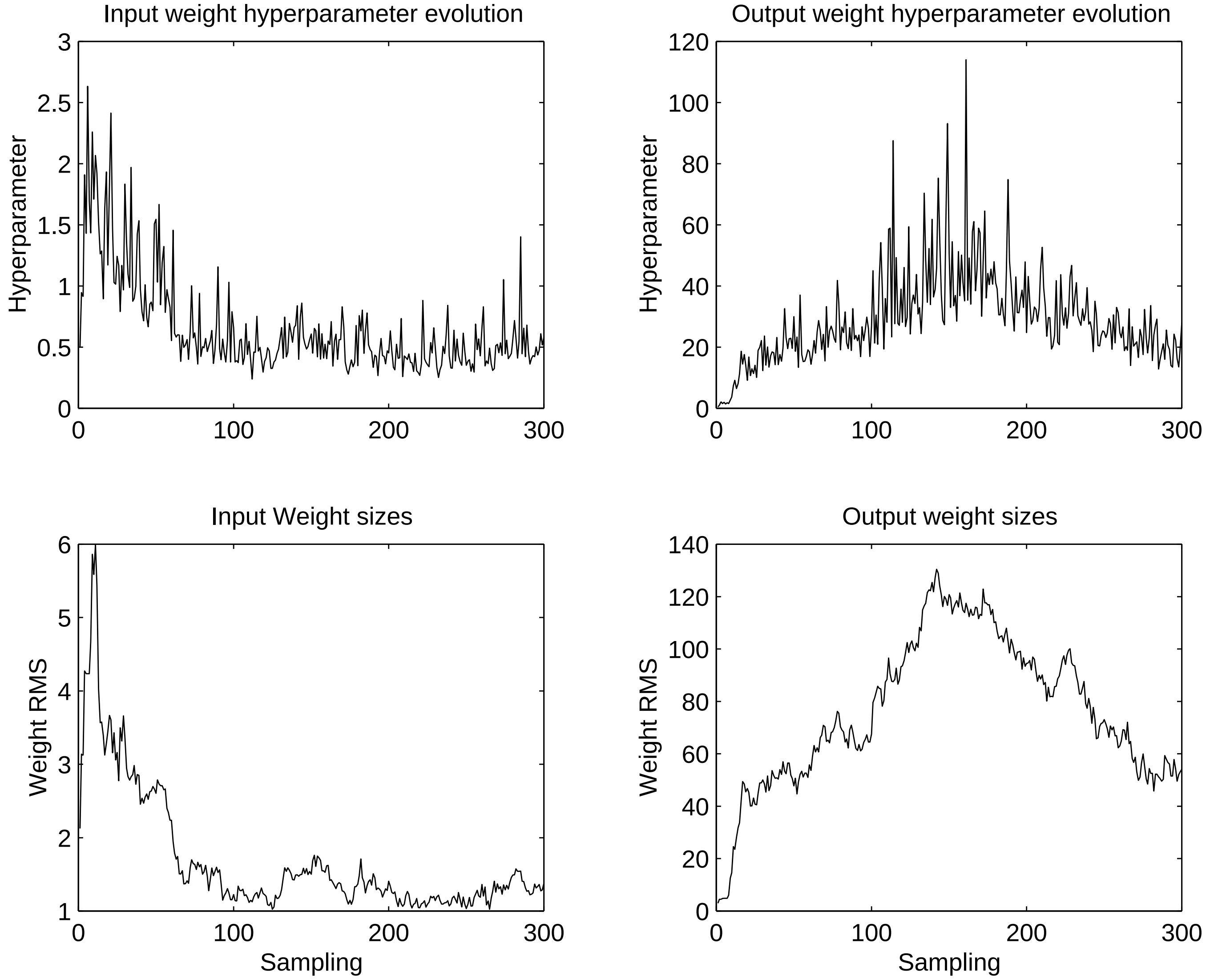


Figure 3: Hyperparameter and size of weights.

LEARNING CURVE

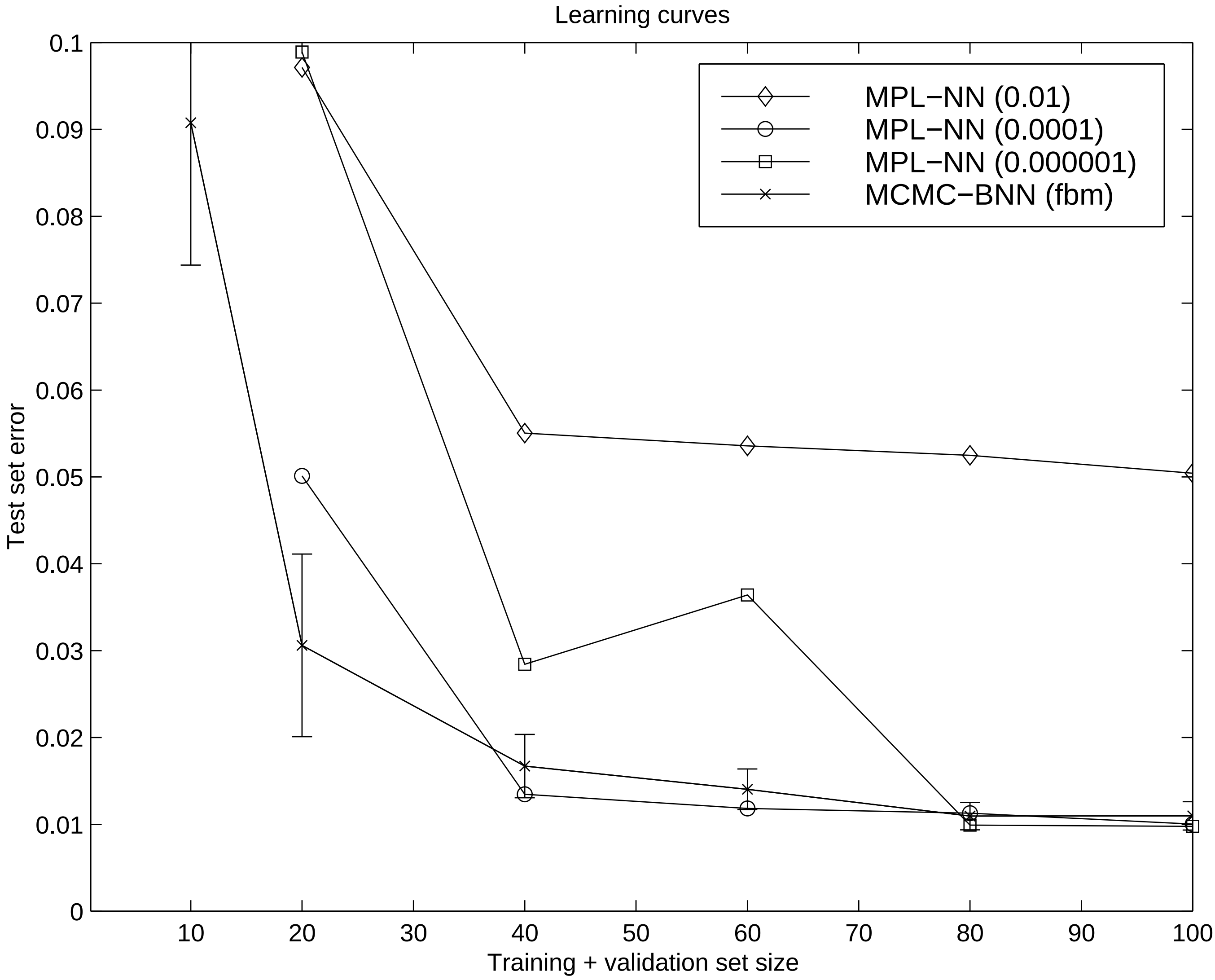


Figure 4: Learning curve.

- (Almost) equal performance on large data sets (consistent with Rasmussen-Hansen density modeling)
- MCMC-BNN better when there is few data sets (consistent with Rasmussen-Hansen density modeling)
- Performance of MPL-NN critically dependent on hyperparameter.

References

[Berger, 1985] Berger, J. O. (1985). *Statistical Decision Theory and Bayesian Analysis*. Springer-Verlag, New York.

[Bridle, 1990] Bridle, J. S. (1990). Probalistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In Fogelman Soulié, F. and Hérault, J., editors, *Neurocomputing: Algorithms, Architectures and Application*, pages 227–236. Springer-Verlag, New York.

[Creutz and Gocksch, 1989] Creutz, M. and Gocksch, A. (1989). Higher-order hybrid Monte Carlo algorithms. *Physical Review Letters*, 63:9–12.

[de Freitas and Niranjan, 1998] de Freitas, J. F. G. and Niranjan, M. (1998). Sequential Monte Carlo methods for optimization of neural network models. Technical Report CUED/F-INFENG/TR 328, Cambridge University Engineering Department, Trumpington Street, Cambridge, England.

[Duane et al., 1987] Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987). Hybrid Monte Carlo. *Physics Leters B*, 195(2):216–222.

[Horowitz, 1991] Horowitz, A. M. (1991). A generalized guided Monte Carlo algorithm. *Physics Letters B.*, 268:247–252.

[MacKay, 1992] MacKay, D. J. C. (1992). The evidence framework for backpropagation networks. *Neural Computation*, 4(5):720–736.

[Moody, 1992] Moody, J. E. (1992). The *effective* number of parameters: An analysis of generalization and regularization in nonlinear learning systems. In Moody, J. E., Hanson, S. J., and Lippmann, R. P., editors, *Advances in Neural Information Processing Systems: Proceedings of the 1991 Conference*, pages 847–854, San Mateo, CA. Morgan Kaufmann Publishers. NIPS-4.

[Neal, 1993] Neal, R. M. (1993). Probabilistic inference using markov chain monte carlo methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto.

[Neal, 1994] Neal, R. M. (1994). An improved acceptance procedure for the hybrid Monte Carlo algorithm. *Journal of Computational Physics*, 111:194–203.

[Neal, 1995] Neal, R. M. (1995). Suppressing random walks in Markov chain Monte Carlo using ordered overralaxation. Technical Report 9508, Department of Statistics, University of Toronto.

[Neal, 1996] Neal, R. M. (1996). *Bayesian Learning for Neural Networks*. Number 118 in Lecture Notes in Statistics. Springer, New York, USA.

[Neal, 1998a] Neal, R. M. (1998a). Flexible Bayesian models. `http://www.cs.toronto.ca/~radford`. "Software that implements Flexible Bayesian Models based on Neural networks, Gaussian processes, and mixtures and that demonstrates Markov chain Monte Carlo methods".

[Neal, 1998b] Neal, R. M. (1998b). Suppressing random walks in Markov chain Monte Carlo using ordered overrelaxation. In Jordan, M. I., editor, *Learning in Graphical Models*, pages 205–225. Kluwer Academic Publishers, Dordrecht.

[Rasmussen, 1996] Rasmussen, C. E. (1996). A practical Monte Carlo implementation of Bayesian learning. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems: Proceedings of the 1995 Conference*, Cambridge, Massachusetts. MIT Press. NIPS-8.

[Rossky et al., 1978] Rossky, P. J., Doll, J. D., and Friedman, H. L. (1978). Brownian dynamics as smart Monte Carlo simulation. *Journal of Chemical Physics*, 69:4628–4633.

[Sporring, 1997] Sporring, J. (1997). A prior of saliency based pruning algorithms. Technical Report DIKU-97/8, Department of Computer Science, University of Copenhagen, Copenhagen, Denmark.

[Thodberg, 1996] Thodberg, H. H. (1996). A review of Bayesian neural networks with an application to near infrared spectroscopy. *IEEE Transactions on Neural Networks*, 7(1):56–72.