[Paper review 2]

Bayesian Learning For Neural Networks (Radford M. Neal, 1995)

[Contents]

3. Monte Carlo Implementation

- hybrid MC > simple Metropolis, due to its avoidance of random walk behavior
- evaluate the merits of "dynamical computation", based on "partial gradients" and of "windowed variant" of hybrid MC

Problem with Gaussian approximation

- will not always produce good approximations

  (especially when complex models are used)

- Thus, need a learning that "does not rely" on any assumption concerning the "form of posterior dist'n"

  (MC avoids such assumptions... but slow)

  (Better results can be obtained by using hybrid Monte Carlo!)

[extra] summary of hybrid Monte Carlo (= Hamiltonian Monte Carlo)

- MCMC method that uses "derivatives of pdf"
- efficient transitions
- use approximate Hamiltonian dynamics simulations (based on numerical integration)

  (as a way to avoid "mode collapse" = to search all the space)

(1) Target density

- θ : parameter of interest
- $p(\theta)$: target distribution
- $P(\theta \mid y)$; posterior

(2) Auxiliary Momentum Variable : ρ

joint density : $p(\rho, \theta)=p(\rho \mid \theta) p(\theta)$

$\rho \sim \operatorname{Multi-Normal}(0, M)$

(3) Hamiltonian

$$\begin{aligned} H(\rho, \theta) & =-\log p(\rho, \theta) \\ & =-\log p(\rho \mid \theta)-\log p(\theta) \\ & =T(\rho \mid \theta)+V(\theta) \end{aligned}$$

- potential energy : $T(\rho \mid \theta)=-\log p(\rho \mid \theta)$
- kinetic energy : $V(\theta)=-\log p(\theta)$

(4) Generating Transitions

step 1) momentum(ρ) is drawn

- $\rho \sim \operatorname{Multi-Normal}(0, M)$

step 2) calculate the derivative

- $\dfrac{d\theta}{dt} = +\dfrac{\partial H}{\partial \rho} = +\dfrac{\partial T}{\partial \rho}$
  $\dfrac{d\rho}{dt} = -\dfrac{\partial H}{\partial \theta} = -\dfrac{\partial T}{\partial \theta} - \dfrac{\partial V}{\partial \theta} = -\dfrac{\partial V}{\partial \theta}$

step 3) Leapfrog Integrator

- two-state differential equation
- takes discrete steps, of some small time interval (=ϵ)
  $\rho \leftarrow \rho - \dfrac{\epsilon}{2}\dfrac{\partial V}{\partial \theta}$
- $\theta \leftarrow \theta + \epsilon M^{-1}\rho$
  $\rho \leftarrow \rho - \dfrac{\epsilon}{2}\dfrac{\partial V}{\partial \theta}$
- resulting state = (ρ^*, θ^*)

step 4) Metropolis Acceptance step

- probability of keeping the proposal (ρ^*, θ^*) = $\min(1, \exp(H(\rho,\theta) - H(\rho^*, \theta^*)))$

3-1. The hybrid Monte Carlo algorithm

hybrid MC = (1) Metropolis algorithm + (2) Dynamical simulation

3-1-1. Formulating the problem in terms of energy

sampling from canonical (=Boltzmann) distribution, which derivatives of the pdf can be computed!

suppose we wish to sample from distribution of "Position Variable" (= q)

probability density : $P(q) \propto \exp(-E(q))$, where $E(q)$ is "potential energy function"

(That is, $E(q) = -\log P(q) - logZ$)

To allow the use of "dynamical method", introduce "MOMENTUM VARIABLE" (=p)

- $P(q,p) \propto \exp(-H(q,p))$
- $H(q,p) = E(q) + K(p)$ = potential energy + kinetic energy
- kinetic energy : $K(p) = \sum_{i=1}^{n} \frac{p_i^2}{2m_i}$ (where m = mass)

We can proceed by defining a Markov chain that converges to the canonical distribution of q and p

Then, simply IGNORE p when estimating expectations of functions of q (since q and p are independent)

3-1-2. The stochastic dynamics method

- task of sampling from the canonical distribution for q & p
- sampling at a fixed total energy (=$H(q,p)$)

3 Properties of Hamiltonian dynamics

- 1) H stays constant, as q and p varies

  $$\frac{dq_i}{d\tau} = +\frac{\partial H}{\partial p_i} = \frac{p_i}{m_i}$$
  $$\frac{dp_i}{d\tau} = -\frac{\partial H}{\partial q_i} = -\frac{\partial E}{\partial q_i}$$

- 2) Hamiltonian preserves the volumes

  $$\frac{dH}{d\tau} = \sum_i \left[\frac{\partial H}{\partial q_i}\frac{dq_i}{d\tau} + \frac{\partial H}{\partial p_i}\frac{dp_i}{d\tau}\right] = \sum_i \left[\frac{\partial H}{\partial q_i}\frac{\partial H}{\partial p_i} - \frac{\partial H}{\partial p_i}\frac{\partial H}{\partial q_i}\right] = 0$$

- 3) Dynamics is reversible

These properties imply that "canonical distribution for q and p is invariant(=stationary) w.r.t transitions "

Conclusion : transitions based on Hamiltonian dynamics will "explore the whole region of phase space" with a given value of H

In stochastic dynamics

- update the momentum ($= p$)

  (since q and p are independent, p may be updated without reference to q)
- update of p can change H, thus allow the entire phase space to be explored

Leapfrog method

- Hamiltonian dynamics can not be simulated exactly,
- but can only be approximated by some "discretization" using finite time step
- single iteration calculates approximations to the position & momentum

$$\hat{p}_i\left(\tau + \frac{c}{2}\right) = \hat{p}_i(\tau) - \frac{\epsilon}{2}\frac{\partial E}{\partial q_i}(\hat{q}(\tau))$$
$$\hat{q}_i(\tau + \epsilon) = \hat{q}_i(\tau) + \epsilon\frac{\hat{p}_i\left(\tau + \frac{\epsilon}{2}\right)}{m_i}$$
$$\hat{p}_i(\tau + \epsilon) = \hat{p}_i\left(\tau + \frac{c}{2}\right) - \frac{\epsilon}{2}\frac{\partial E}{\partial q_i}(\hat{q}(\tau + \epsilon))$$

- in the leapfrog discretization, phase space volume is still preserved & dynamics can also still be reversed!
- BUT, H no longer stays exactly constant.... by merging it with "Metropolis Algorithm"

3-1-3. Hybrid Monte Carlo

- merge leapfrog & "Metropolis Algorithm"
- samples points in phase space by means of Markov Chain in which "stochastic" & "dynamical" transitions alternate

  1) stochastic transitions : momentum (=p)is replaced using Gibbs sampling

  2) dynamic transitions : it is only candidate (to be accepted or rejected!)
- notation :

  ϵ : magnitude of the leapfrog step size

  L : number of leapfrog iteration in each

  total number of iteration = $\epsilon \times L$

3-2. An implementation of BNN Learning

Notation

- network : $f(x, \theta)$
- prior for network parameters : $P(\theta, \gamma)$ (where γ is hyperparameter)
- conditional probabilities for the target : $P(y \mid x, \theta, \gamma)$

Our ultimate goal : predict $y^{(n+1)}$

- Posterior :
  $P\left(\theta, \gamma \mid\left(x^{(1)}, y^{(1)}\right), \ldots,\left(x^{(n)}, y^{(n)}\right)\right) \quad \propto \quad P(\gamma) P(\theta \mid \gamma) \prod_{c=1}^{n} P\left(y^{(c)} \mid x^{(c)}, \theta, \gamma\right)$
- Posterior Predictive :
  $$\begin{aligned} & P\left(y^{(n+1)} \mid x^{(n+1)},\left(x^{(1)}, y^{(1)}\right), \ldots,\left(x^{(n)}, y^{(n)}\right)\right) \\ & \quad=\int P\left(y^{(n+1)} \mid x^{(n+1)}, \theta, \gamma\right) P\left(\theta, \gamma \mid\left(x^{(1)}, y^{(1)}\right), \ldots,\left(x^{(n)}, y^{(n)}\right)\right) d \theta d \gamma \end{aligned}$$

For a regression model, single-valued prediction that minimizes squared-error loss :

- $\hat{y}^{(n+1)}=\int f\left(x^{(n+1)}, \theta\right) P\left(\theta, \gamma \mid\left(x^{(1)}, y^{(1)}\right), \ldots,\left(x^{(n)}, y^{(n)}\right)\right) d \theta d \gamma$
- to calculate the integral, use "Monte Carlo approach" (= approximated by the average value)

believes hybrid MC is the most promising MC method (for sampling from the posterior of NN model)

Goal 1) handle a wide range of network architecture

Goal 2) minimize the amount of "tuning"

3-2-1. Gibbs sampling for hyperparameters

2 types of hyperparameters

- 1) term of which the prior distribution of the parameters is expressed
- 2) noise levels in regression models

(1) prior distribution of the parameters

- control the standard deviation for all parameters in a certain group
- $u_i \sim N(0, \sigma_u^2)$

  (express it as "precision".... $\tau_u = \sigma_u^{-2}$)
- a) likelihood :

  $P(u_1, \ldots, u_k \mid \tau_u) = (2\pi)^{-k/2} \tau_u^{k/2} \exp\left(-\tau_u \sum_i u_i^2 / 2\right)$
- b) prior : (Gamma prior)

  $\tau_u \sim \Gamma(\omega_u, \alpha_u)$

  $P(\tau_u) = \frac{(\alpha_u/2\omega_u)^{\alpha_u/2}}{\Gamma(\alpha_u/2)} \tau_u^{\alpha_u/2-1} \exp(-\tau_u \alpha_u / 2\omega_u)$
- c) posterior (conjugate relationship)

$$\begin{aligned} P(\tau_u \mid u_1, \ldots, u_k) &\propto \tau_u^{\alpha_u/2-1} \exp(-\tau_u \alpha_u / 2\omega_u) \cdot \tau_u^{k/2} \exp\left(-\tau_u \sum_i u_i^2 / 2\right) \\ &\propto \tau_u^{(\alpha_u+k)/2-1} \exp\left(-\tau_u \left(\alpha_u/\omega_u + \sum_i u_i^2\right) / 2\right) \end{aligned}$$

  (small values of α_u produce vague priors for τ_u)
- distribution above is what is needed for Gibbs sampling! (since given $u_1, \ldots u_k$, the value of τ_u is independent of other parameters!)

(2) noise levels in regression models

- $\tau_k = \sigma_k^{-2}$

  where τ_k is the standard deviation of the noise, w.r.t the k^{th} target value
- a) likelihood :

  $P\left(y_k^{(1)}, \ldots, y_k^{(n)} \mid x^{(1)}, \ldots, x^{(n)}, \theta, \tau_k\right) = (2\pi)^{-n/2} \tau_k^{n/2} \exp\left(-\tau_k \sum_c \left(y_k^{(c)} - f_k\left(x^{(c)}, \theta\right)\right)^2 / 2\right)$
- b) prior (Gamma prior)

  $P(\tau_k) = \frac{(\alpha/2\omega)^{\alpha/2}}{\Gamma(\alpha/2)} \tau_k^{\alpha/2-1} \exp(-\tau_k \alpha / 2\omega)$
- c) posterior (conjugate relationship)

  $P\left(\tau_k \mid \left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(n)}, y^{(n)}\right), \theta\right) \propto \quad \tau_k^{(\alpha+n)/2-1} \exp\left(-\tau_k \left(\alpha/\omega + \sum_c \left(y_k^{(c)} - f_k\left(x^{(c)}, \theta\right)\right)^2 / 2\right)\right.$

3-2-2. Hybrid Monte Carlo for network parameters

explore the entire posterior

must formulate the desired distribution in terms of "Potential Energy Function"

Notation

- q (= θ) : parameter of our interest / position
- p : auxiliary momentum variable / kinetic

Hyperparameters will remain FIXED throughout one hybrid MC update

Potential Energy :

- $E(\theta)=F(\gamma)-\log P(\theta \mid \gamma)-\sum_{c=1}^{n} \log P\left(y^{(c)} \mid x^{(c)}, \theta, \gamma\right)$

  where $F(\gamma)$: any function of the hyper parameters)
- (detailed form of energy function above will vary with network architecture, prior, and model)

Example)

suppose network parameters form 2 groups u and v

- parameters : $\theta=\left\{u_{1}, \ldots, u_{k}, v_{1}, \ldots, v_{h}\right\}$
- hyperparameters : $\gamma=\left\{\tau_{u}, \tau_{v}, \tau\right\}$

  (where $\tau_{u}=\sigma_{u}^{-2}, \tau_{v}=\sigma_{v}^{-2}$, and $\tau=\sigma^{-2}$)

Then, the potential energy function will be :

- $E(\theta)=\tau_{u} \sum_{i=1}^{k} u_{i}^{2} / 2+\tau_{v} \sum_{j=1}^{h} v_{j}^{2} / 2+\tau \sum_{c=1}^{n}\left(y^{(c)}-f\left(x^{(c)}, \theta\right)\right)^{2} / 2$
- looks similar as error function (with weight decay) !!

Recall! objective of MC is "NOT finding minimum of energy", "BUT sample from distribution"

Choosing good stepsize?

- Hamiltonian : $H(q, p)=q^{2} / 2 \sigma^{2}+p^{2} / 2$
- Leapfrog iteration :

  $$\hat{p}\left(\tau+\frac{\epsilon}{2}\right)=\hat{p}(\tau)-\frac{\epsilon}{2} q(\tau) / \sigma^{2}$$
  $$\hat{q}(\tau+\epsilon)=\hat{q}(\tau)+\epsilon \hat{p}\left(\tau+\frac{\epsilon}{2}\right)$$
  $$\hat{p}(\tau+\epsilon)=\hat{p}\left(\tau+\frac{c}{2}\right)-\frac{\epsilon}{2} q(\tau+\epsilon) / \sigma^{2}$$

  (another view = linear mapping fro m$(\hat{q}(\tau), \hat{p}(\tau))$ to $(\hat{q}(\tau+\epsilon), \hat{p}(\tau+\epsilon))$.)
- If we use the same step size for all components, we have to use step size less than $2 \sigma_{\min}$

  (If other of the σ_{i} are much larger than $\sigma_{\min}$, large number of leapfrog steps will be required! $\rightarrow$ inefficiency!)
- Thus, "Use different stepsize for each component"

  $\epsilon_{i} \approx \eta\left[\frac{\partial^{2} E}{\partial q_{i}^{2}}\right]^{-1 / 2}$

- Unfortunately, unable to set step size based on the actual values of $\frac{\partial^2 E}{\partial q_i^2}$ at the starting point

  (only allowed to use the current values of the hyper-parameters, which are fixed during update!)

3-2-3. Verifying Correctness

MCMC.. how to verify its correctness?

- dynamics should be reversible & preserve volume in phase space
- end-point of the trajectory be accepted/rejected based on a correct computation of the change in total energy

Errors

- can be recognized by high rejection rate

  (can be reduced by using small step size / by using very short trajectories)

3-3. A demonstration using hybrid Monte Carlo

3-3-1. robot arm problem

Problem setting

- actual relationship between $x \& y$:
  $y_1 = 2.0\cos(x_1) + 1.3\cos(x_1 + x_2) + e_1$
  $y_2 = 2.0\sin(x_1) + 1.3\sin(x_1 + x_2) + e_2$
  where $e_1 \sim N(0, 0.05^2)$ & $e_2 \sim N(0, 0.05^2)$

Network Architecture

- 1) input-to-hidden weights
- 2) hidden unit biases
- 3) hidden-to-output weights

(use 3 hyperparameters to control the standard deviation of each)

Hyperparameters

- Gamma prior to hyperparameters : $P(\tau_u) = \frac{(\alpha_u/2\omega_u)^{\alpha_u/2}}{\Gamma(\alpha_u/2)}\tau_u^{\alpha_u/2-1}\exp(-\tau_u\alpha_u/2\omega_u)$, where all $\alpha = 0.2$
- Noise precision be a hyperparameter as well : $P(\tau_k) = \frac{(\alpha/2\omega)^{\alpha/2}}{\Gamma(\alpha/2)}\tau_k^{\alpha/2-1}\exp(-\tau_k\alpha/2\omega)$, where $\alpha = 0.2$ and $\omega = 200$

3-3-2. Sampling using hybrid MC

Two phase = initial phase + sampling phase

- 1) Initial phase : start from initial state & simulate until it reaches rough approximation
- 2) sampling phase : continue from the end of 1)
- Those two phases may be repeated several times (with different random number seeds)

Goal : check how these two phases works in hybrid MC!

Specify the number of leap frog iterations L & adjustment factor η

(best value for L is different for the initial phase & sampling phase)

(1) Initial phase

Comparison of runs ($L \times \epsilon$ is same, with different L and ϵ)

(hence, approximately the same amount of computation time)

Setting

- $L \times \epsilon$ = 2^{10}
- candidate (1)$(L, \epsilon) = (2^{10}, 2^{0})$

  candidate (2)$(L, \epsilon) = (2^{6}, 2^{4})$

  candidate (3)$(L, \epsilon) = (2^{2}, 2^{8})$

Result

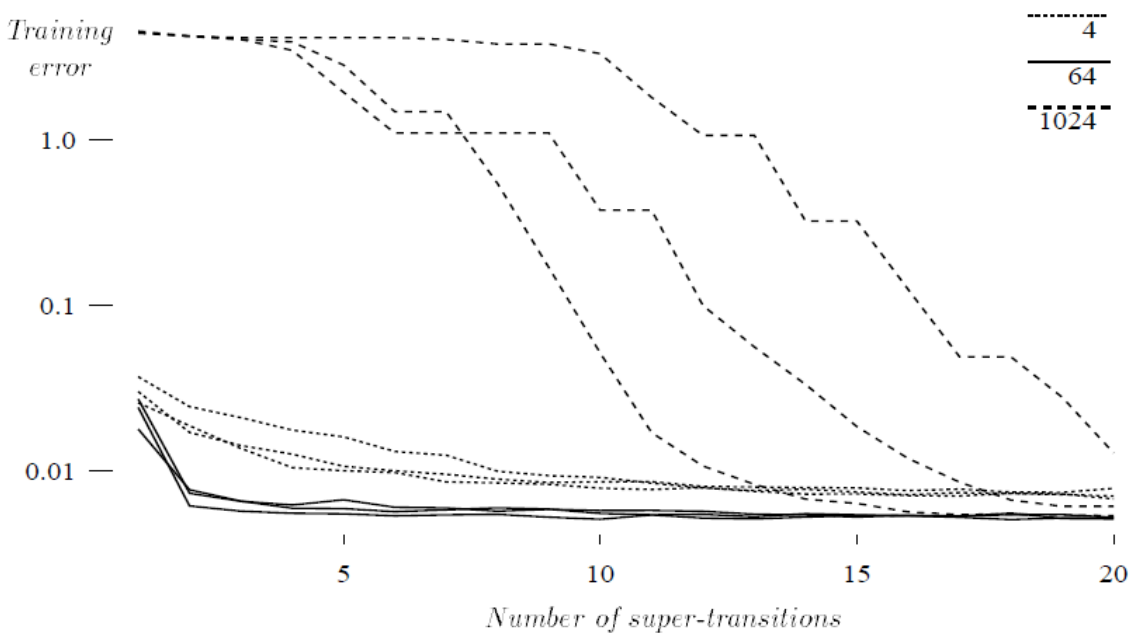

Figure 3.2: Progress of hybrid Monte Carlo runs in the initial phase. The plot shows the average squared error on the training set after each super-transition, on a log scale. The solid lines show the progress of three runs in which trajectories 64 leapfrog iterations long were used. Dotted lines show the progress of three runs with trajectories of 4 leapfrog iterations, dashed lines the progress of three runs with trajectories of 1024 leapfrog iterations.

- training error was initially very high

- once training error had largely stabilized at a lower value, all those chains had reached at least a rough approximation to the equilibrium distribution
- $L = 2^6 = 64$ seems the best!
- Initial Phase is complete!

(2) Sampling phase

- start from the end of Initial phase (let us use the chain with $L = 64$)
- using η randomly! (= adjustment factor)

  (review : $\epsilon_i \approx \eta \left[\frac{\partial^2 E}{\partial q_i^2}\right]^{-1/2}$)

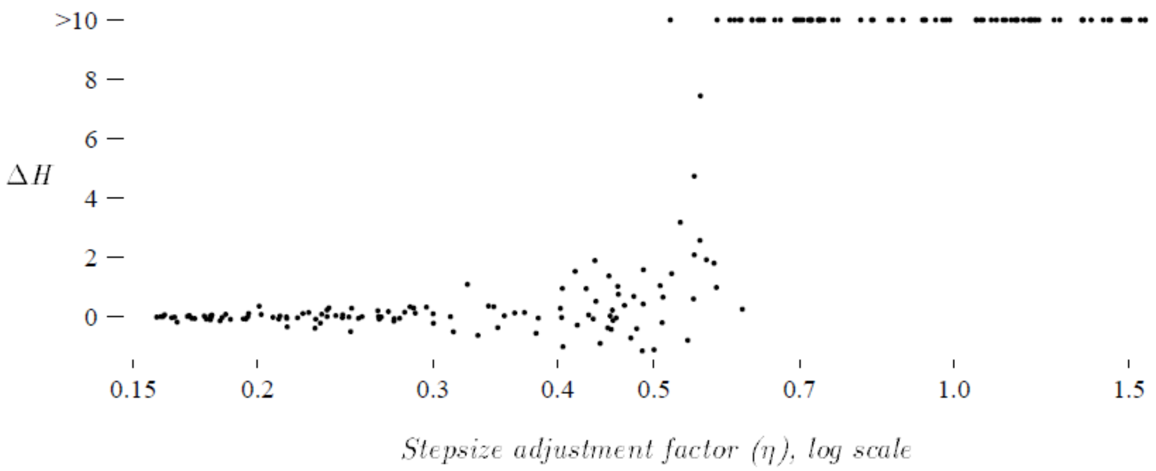

Figure 3.3: Error in energy for trajectories computed with different stepsizes. Each point plotted shows the change in total energy (H) for a trajectory of 100 leapfrog iterations in which stepsizes were adjusted by the factor given on the horizontal axis (with changes greater than 10 plotted at 10). The starting point for the first trajectory was the last state from one of the initial phase runs with $L = 64$ shown in Figure 3.2. Starting points for subsequent trajectories were obtained by continuing the simulation using hybrid Monte Carlo with these trajectories, along with Gibbs sampling updates of the hyperparameters. Values for η were randomly generated from a log-uniform distribution.

- η greater than 0.5 seems unstable (large errors)
- $\eta = 0.3$ seems close to optimal!

We have finished choosing η. Now, choose L

- variation of several quantities along a single trajectory 10000 leap frog iterations ($L \times \epsilon = 10000$)
- $\eta = 0.3$
- start from one of the final state of the initial phase

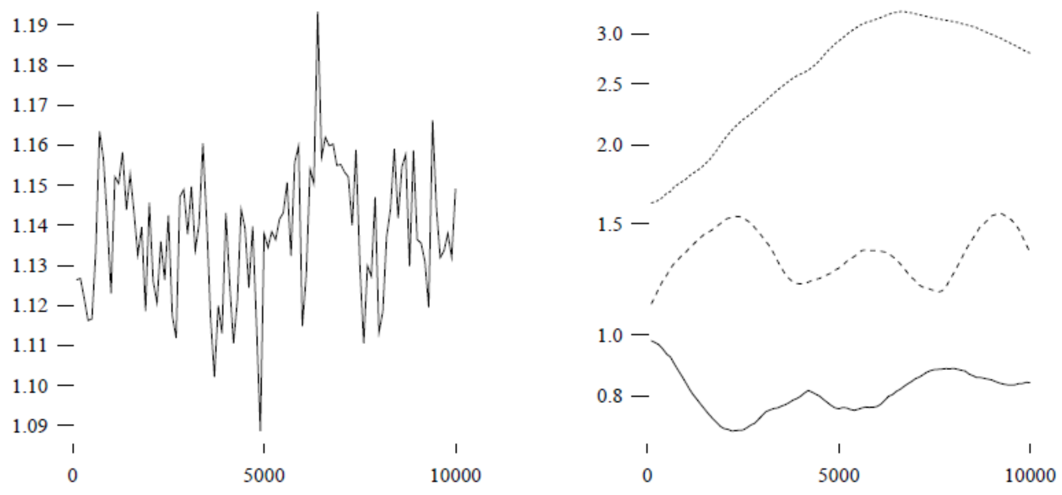

Figure 3.4: Degree of correlation along a trajectory. The plot on the left shows the first output of the network for inputs $(-1.47, 0.75)$, as the network parameters vary along a trajectory 10000 leapfrog interations long (with $\eta = 0.3$), plotted every 100 iterations. On the right, the variation in the square root of the average squared magnitude of parameters in three classes is shown — for input-hidden weights (solid), hidden biases (dotted), and hidden-output weights (dashed) — plotted on a log scale. The trajectory began with the state at the end of one of the initial phase runs with $L = 64$ shown in Figure 3.2.

- [Left plot] Although some short-range correlations are evident, these appear to die out within about 500 leapfrog iterations

  $\rightarrow L = 500$ might seem reasonable
- [Right plot] Correlations are evident in these quantities over spans of several thousands leapfrog iterations

  $\rightarrow L$ should be several thousands!
- 2 results conflicts! which one to believe?

Pay attention to "long trajectories"! reason?

- Reason 1) Initial phase produces a state that is only presumed to be from roughly the equilibrium distribution. Further exploration in the sampling phase might tell that "true equilibrium" is in fact different!
- Reason 2) It is possible that if the values were examined over a LONGER period, significant long-term correlations would be evident!

Setting

- $L \times \epsilon$ = 32000
- candidate (1)$(L, \epsilon) = (125, 256)$

  candidate (2)$(L, \epsilon) = (2000, 16)$

  candidate (3)$(L, \epsilon) = (32000, 1)$
- Graph : "Variation in the square root of the average squared magnitudes of parameters in the three classes"

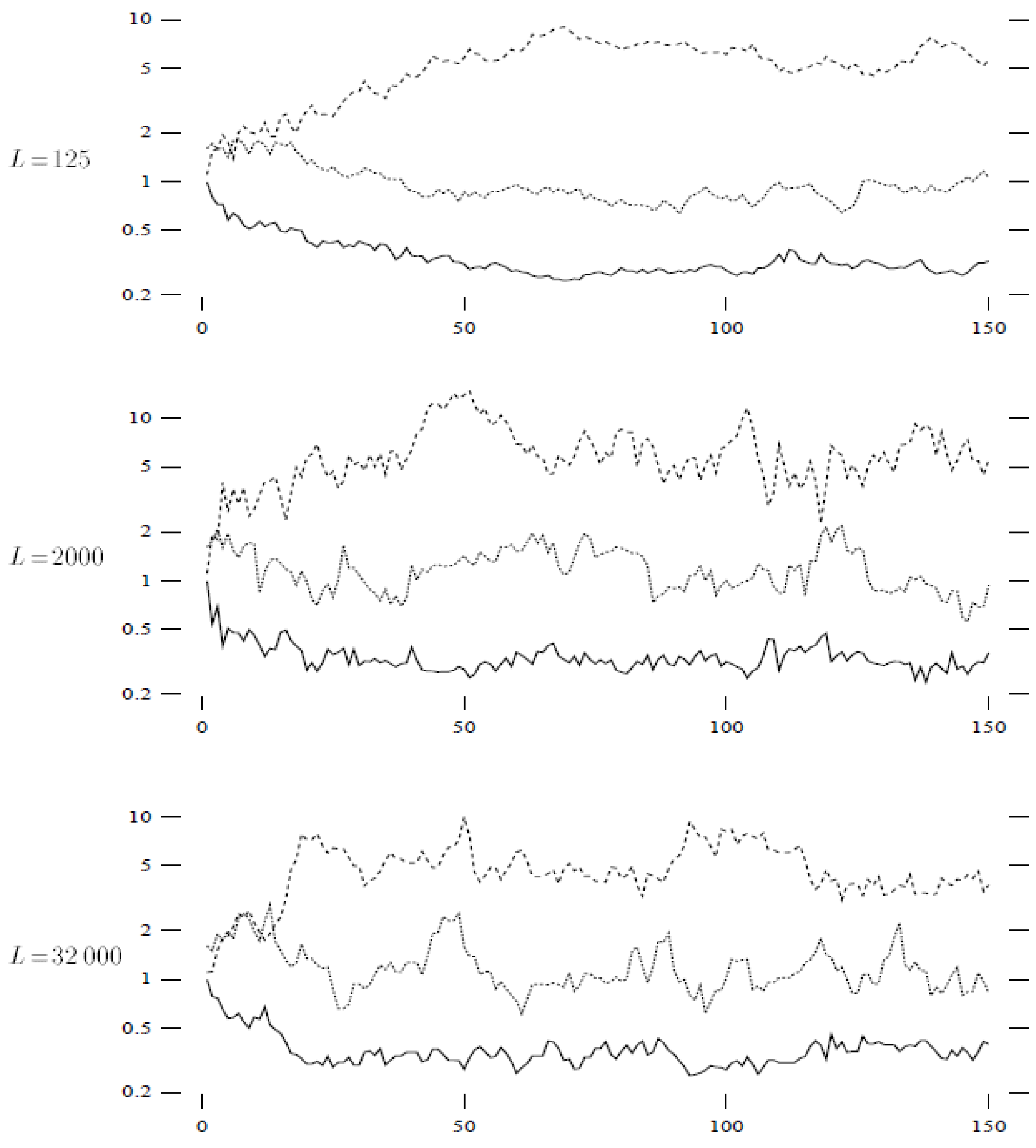

Figure 3.5: Progress of hybrid Monte Carlo runs in the sampling phase. These plots show the variation in the square root of the average squared magnitudes of parameters in the three classes during the course of hybrid Monte Carlo sampling runs using various trajectory lengths (L). The stepsize adjustment factor was $\eta = 0.3$ in all cases. The runs were started with the state at the end of one of the initial phase runs with $L = 64$ shown in Figure 3.2. The horizontal axes show the number of super-transitions, each consisting of 32000 leapfrog iterations. The vertical axes show the square roots of the average squared magnitudes on a log scale, with input-hidden weights shown with solid lines, hidden biases with dotted lines, and hidden-output weights with dashed lines.

- As seen above, "Initial phase had not fully converged" (1024 was not enough!)
- candidate (1)$(L = 125)$: needed about 50 sampling phase super-transitions to reach equilibrium

  candidate (2,3)$(L = 2000, 32000\text{each})$: needed about 25 sampling phase super-transitions to reach equilibrium

Graph for three classes (Input-Hidden weight & Hidden bias & Hidden-Output weight)

- $L = 8000$ have the smallest autocorrelations

- reasonable value for L can be selected before extensive computations are done!

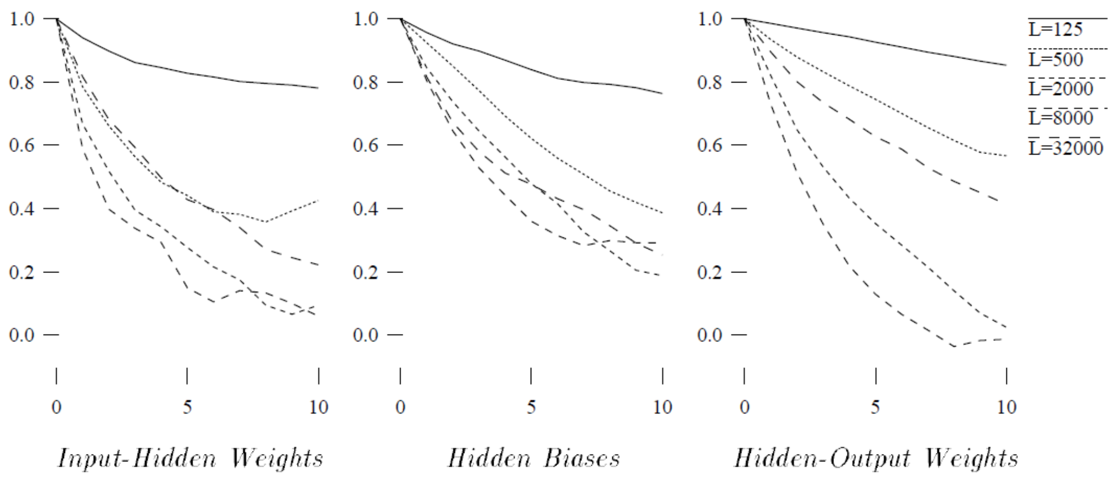

Figure 3.6: Autocorrelations for different trajectory lengths. The plots show autocorrelations for the square root of the average squared magnitude of network parameters in each of three classes. The

3-3-3. Making Predictions