On Bayesian Neural Networks at Finite Temperature

Robert J.N. Baldock[1, *]

[1] *Theory and Simulation of Materials (THEOS), and National Centre for*
Computational Design and Discovery of Novel Materials (MARVEL),
École Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland
(Dated: March 2019)

We recapitulate the Bayesian formulation of neural network based classifiers and show that, while sampling from the posterior does indeed typically lead to better generalisation than is obtained by standard optimisation of the cost function, even better performance can in general be obtained by sampling finite temperature distributions derived from the posterior. Taking the example of two different deep (3 hidden layers) classifiers for MNIST data, we find quite different temperatures to be appropriate in each case. As the temperature is increased from the value of minimum test error, classifiers transition from accurate classifiers to classifiers that are worse than assigning equal probability to each class. Efficient studies of these temperature-induced effects are enabled using a replica-exchange Hamiltonian Monte Carlo simulation technique. Finally, we show how thermodynamic integration can be used to perform model selection for deep neural networks. Similar to the Laplace approximation, this approach assumes that the posterior is dominated by a single mode. Crucially, however, no assumption is made about the shape of that mode and it is not required to compute and invert the Hessian.

I. BAYESIAN FORMULATION OF NEURAL NETWORK CLASSIFIERS

The Bayesian formulation of neural network classifiers was introduced by David MacKay [12]. In training a classifier one typically uses a minimisation algorithm to find a local minimum of a cost function defined over the parameters of the neural network model. If we use a softmax then the activation of the output units sum to one, and can be interpreted as the probability, assigned by the network, to the assertion that the input belongs to each individual class. We can write this as $\mathrm{prob}\,(q|\mathbf{x}, \mathbf{w})$ where q is the index of the class, $\mathbf{x}$ is the single data example to be classified and $\mathbf{w}$ represents the parameters of the neural network. For data in the training set $D = \{\mathbf{x}_i, t_i\}$ where $\mathbf{x}_i$ is an input data example and t_i the corresponding class label, then the probability that the network classifies $\mathbf{x}_i$ correctly is $\mathrm{prob}\,(q = t_i|\mathbf{x}_i, \mathbf{w})$. This probability is a function of the weights $\mathbf{w}$. The probability that the network classifies the complete data set correctly is called the *likelihood* of the data, $\mathrm{prob}\,(D|\mathbf{w})$, and is given by $\prod_i \mathrm{prob}\,(q = t_i|\mathbf{x}_i, \mathbf{w})$.

---

* rjnbaldock@gmail.com

In general, the posterior probability for the weights, which expresses what we have learned about the weights from the data, is given by

$$\mathrm{prob}\,(\mathbf{w}|D) = \frac{\mathrm{prob}\,(D|\mathbf{w})\,\mathrm{prob}\,(\mathbf{w})}{\mathrm{prob}\,(D)} \tag{1}$$

$$\propto \prod_i \mathrm{prob}\,(q = t_i|\mathbf{x}_i, \mathbf{w})\mathrm{prob}\,(\mathbf{w}) \tag{2}$$

where $\mathrm{prob}\,(\mathbf{w})$ is the prior probability distribution for the weights, and represents our initial state of knowledge before the collection of any data.

If we assign a Gaussian prior with standard deviation σ to the parameters $\mathbf{w}$ then the posterior (2) is given by

$$\mathrm{prob}\,(\mathbf{w}|D) \propto \prod_i \mathrm{prob}\,(q = t_i|\mathbf{x}_i, \mathbf{w})e^{-\frac{\mathbf{w}^2}{2\sigma^2}}. \tag{3}$$

The posterior probability of the weights (3) is therefore maximised when we minimise

$$J(\mathbf{w}|D) = -\log\left[\mathrm{prob}\,(D|\mathbf{w})\,\mathrm{prob}\,(\mathbf{w})\right] \tag{4}$$

$$= -\sum_i \log\left[\mathrm{prob}\,(q = t_i|\mathbf{x}_i, \mathbf{w})\right] + \frac{\mathbf{w}^2}{2\sigma^2}. \tag{5}$$

which can immediately be recognised as the cross entropy loss function with L2 regularisation. The minimum of (5) is called the maximum a posteriori solution.

In this paper we will use a uniform prior for $\mathbf{w}$

$$\mathrm{prob}\,(\mathbf{w}) = \begin{cases} \frac{1}{\prod_{i=1}^d \sigma_i}, & |w_i| < \sigma_i/2 \,\forall\, i, \\ 0, & \text{Elsewhere.} \end{cases} \tag{6}$$

Within the bounds of this prior the cost function is given by

$$J(\mathbf{w}|D) = -\sum_i \left(\log\left[\mathrm{prob}\,(q = t_i|\mathbf{x}_i, \mathbf{w})\right] - \log \sigma_i\right). \tag{7}$$

Outside the bounds of the prior the cost function is infinite. Minimising the cost function will simply mean minimising the likelihood within these bounds.

II. ENERGY AND TEMPERATURE

One may define the *potential energy* over the parameters of the network [12]

$$E(D|\mathbf{w}) = -\log \mathrm{prob}\,(D|\mathbf{w}) \tag{8}$$

$$= -\sum_i \log\left[\mathrm{prob}\,(q = t_i|\mathbf{x}_i, \mathbf{w})\right] \tag{9}$$

and the *"temperature-adjusted"* posterior distribution

$$\text{prob}\left(\mathbf{w}|T, D\right) \propto e^{-\frac{1}{T}E(D|\mathbf{w})}\,\text{prob}(\mathbf{w}). \qquad (10)$$

where T controls the noise in our posterior. We name T the *"temperature"* in analogy with thermodynamic temperature.

For this paper, it is important to imagine how the temperature-adjusted posterior behaves as we vary T. The distribution (10) tends towards the prior distribution for $T \to \infty$, and is equal to the true posterior distribution (2) for $T = 1$. At $T < 1$ (low temperature) the temperature-adjusted posterior (10) is concentrated around the maximum a posteriori solution (the minimum of (5)).

A. TEMPERATURE VS BATCH SIZE

It is known that mini-batch learning improves generalisation (e.g. [17]). Recently there has been much interest in how the batch size in mini-batch learning controls the noise level during learning, and under what conditions the optimiser can be said to be approximately sampling from the posterior. Here we instead use temperature to precisely control the noise in full batch training and explore the use of $T > 0$ as an alternative to early stopping.

III. ON THE OPTIMAL TEMPERATURE FOR BAYESIAN NEURAL NETWORKS

Here we examine the average training and test errors of two deep neural network classifiers with parameters sampled from the temperature-adjusted posterior (10), across a wide range of temperatures. We find that sampling from the posterior ($T = 1$) indeed leads to better test error than standard optimisation of the cost function. However, we also find that in general, improved performance can be obtained at temperatures other than $T = 1$ and that the optimal value of T depends on the neural network and the data set.

A. DATA AND NEURAL NETWORK ARCHITECTURES

All calculations reported in this paper are performed using the MNIST data set. For computational speed the MNIST data images are rescaled down from 28 x 28 pixels to 16 x 16 (256 input dimensions). Standard normalisation is applied to the full MNIST data set. A random selection of rescaled data is shown in Figure 1.

Calculations are performed on stratified samples from the full MNIST training set. Remaining samples from the MNIST training set, not included in these reduced stratified training sets, are appended to the MNIST test set. We will adopt the convention that a data set containing n data points be denoted D_n. Thus D_{500} is a
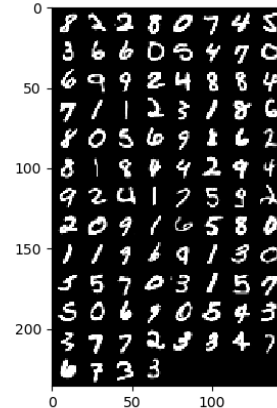

FIG. 1. 100 random data samples from the MNIST training set, transformed as described in the text.

data set of 500 images, 50 for each class. Each time a data set of a new size is generated, it is stored. Thus all calculations using data sets of the same size make use of exactly the same images. These data sets are available from an open source repository, together with the code required to run these calculations [14].

In this section we make use of two neural networks. The first, $M^{(3)}$, has 256 input neurons, 3 hidden layers of 40 logistic neurons, and an output layer of 10 linear neurons to which a softmax is applied. Thus, the loss function for this network is the cross-entropy loss function. A second network $M^{(3*)}$ has the same structure as $M^{(3)}$, except that the final layer of linear neurons is replaced by a layer of *logistic* neurons. A softmax is also applied to the output of these logistic neurons, so the loss function for this network is also the cross entropy loss function. The use of logistic neurons in the final layer drastically reduces the capacity of $M^{(3*)}$ as compared to $M^{(3)}$, since in $M^{(3*)}$ the values entering the softmax are constrained to the interval $(0, 1)$.

B. CHOICE OF PRIOR

The standard procedure for initialising the weights and biases of a neural network before training is to choose them uniformly at random inside the region $[-\frac{1}{\sqrt{k_i}}, \frac{1}{\sqrt{k_i}}]$. Here i is an integer labelling the weight or bias, and k_i is the fan in, including the bias, of the neuron to which weight or bias i points.

In this paper we use a (uniform) prior (6) of this same shape, but made 50 times wider for $M^{(3)}$ and 1000 times wider for $M^{(3*)}$. Including the factor $\frac{1}{2}$ from (6) for $M^{(3)}$ we therefore have

$$\sigma_i = \frac{100}{\sqrt{k_i}}. \qquad (11)$$

Similarly, for $M^{(3*)}$ the prior is as in (11), but with a factor 2000 in place of 100.

These factors are chosen to create priors that capture important minima of $E(D|\mathbf{w})$, without being overly specific. We choose the factors 50 and 1000 because, after standard initialisation followed by minimisation we found $\max_i \left(w_i \div \frac{1}{\sqrt{k_i}} \right)$ to be 30.3 for $M^{(3)}$ and 890.0 for $M^{(3*)}$. We repeated this procedure 25 times for each network to be sure these are typical values. Minimisation is performed using Alg. 3 (see Sec. III C 2), which is related to the FIRE optimisation algorithm [3].

C. METHODS

We develop a Hamiltonian (also called Hybrid) Monte Carlo (HMC) [4, 16] formulation of a technique called replica-exchange molecular dynamics (REMD) [19, 21] for use with NN models. This method enables efficient exploration of the temperature-adjusted posterior (10) across a range of temperatures, far faster than sampling those temperatures independently.

To our knowledge, this is the first time a replica-exchange HMC (RE-HMC) approach has been applied to study neural networks. HMC was first introduced to neural networks by Radford Neal [15], who showed HMC to be an efficient means of sampling the posterior (1). Different RE-HMC algorithms specialised for different application domains can be found in [8, 23].

In RE-HMC the user specifies a temperature range to study. Normally this range goes from the lowest temperature of interest, up to an artificially high temperature where the sampler moves very quickly. Independent samplers are created at particular temperatures throughout that range. These samplers explore $\mathbf{w}$ space according to (10) at their respective temperatures. Periodic attempts are made to exchange $\mathbf{w}$ vectors between neighbouring temperatures. This accelerates $\mathbf{w}$ space sampling at low temperatures because each parameter vector spends some of the time at high temperature where it explores $\mathbf{w}$ space much more rapidly. Importantly, samplers at each temperature are always sampling from the respective temperature-adjusted posterior (10) even immediately after a parameter swap.

A geometric series of temperatures is often employed for REMD in large parameter spaces. This is known [9, 19, 20] to yield nearly equal acceptance rates between neighbouring temperatures, unless the problem is pathological.

1. Hamiltonian Monte Carlo

HMC [4] unifies two schools of statistical sampling: Markov chain Monte Carlo (MCMC) [13] and Molecular Dynamics (MD) [1]. MD introduces the concept of forward momentum $\mathbf{p}$ and makes use of the gradient of the cost function. Together $\mathbf{p}$ and the gradient information guide $\mathbf{w}$ rapidly through complex spaces. Standard MCMC makes no use of $\mathbf{p}$ or the gradient: instead MCMC makes small random moves and diffuses inefficiently through $\mathbf{w}$ space. However, if an intelligent proposal is known, perhaps a move between regions of state space with large weight in the posterior distribution, but separated by long distances and barriers of low probability density, then MCMC can be an extremely powerful approach.

HMC combines the benefits of MD and MCMC. A short MD trajectory is performed to rapidly displace $\mathbf{w}$. So long as the MD trajectory is reversible and symplectic, then this move can be treated as any other MCMC proposal. Thus HMC enables the combination of intelligent MCMC moves with rapid MD trajectories. HMC has been shown [16] to improve the scaling of sampling a $d-$dimensional multivariate Gaussian distribution from d^2 in MCMC to $d^{\frac{5}{4}}$. The cost of HMC can further be reduced by a factor if one avoids backward motion between MD trajectories [7, 18]. However, in Alg. 1 we present a simple HMC algorithm since this already improves the scaling, and $M^{(3)}$ and $M^{(3*)}$ have 10690 dimensions each. We used the Velocity Verlet scheme [22] to propagate the MD trajectories.

Algorithm 1 One Hamiltonian Monte Carlo trajectory. Samples $\text{prob}(\mathbf{w}) \propto e^{-J(\mathbf{w}|D)}$.

---

procedure HMC($\mathbf{w}$)
    $\mathbf{w}_o = \mathbf{w}$ ▷ Copy initial $\mathbf{w}$
    Draw random $p_{0,i} \sim \mathcal{N}(0,T) \forall i$ ▷ Random $\mathbf{p}_o$
    $U_o = J(\mathbf{w}_o|D) + \frac{\mathbf{p}_o^2}{2}$ ▷ Initial total energy
    Propagate $(\mathbf{w}_o, \mathbf{p}_o)$ using the Velocity Verlet algorithm for L time steps of length dt. Final coordinates are $(\mathbf{w}_n, \mathbf{p}_n)$.
    $U_n = J(\mathbf{w}_n|D) + \frac{\mathbf{p}_n^2}{2}$ ▷ Final total energy
    Set $\mathbf{w} = \mathbf{w}_n$ with probability
    $\min \left[1, e^{\frac{1}{T}(U_o - U_n)} \right]$.
return $\mathbf{w}$.
end procedure

---

2. Replica-Exchange Hamiltonian Monte Carlo

Our RE-HMC algorithm proceeds as follows. We used a geometric series of N_T temperatures, as described in Sec. III C. The current parameters values for the sampler at temperature T_i are $\mathbf{w}_i$. To save space, we will write $E_i = E(D|\mathbf{w}_i)$.

In this fresh implementation independent time steps dt are set for each temperature by running additional HMC trajectories which are not included in the main simulation, and measuring the acceptance rate of the trajectories. The time step is updated to obtain an acceptance rate inside a the range (0.6–0.7), which is centred around the optimal value for HMC sampling of a multivariate Gaussian distribution, 0.65 [16].

Algorithm 2 Replica-exchange Hamiltonian Monte Carlo

---

procedure RE-HMC($\{(T_i, \mathbf{w}_i)\}$)
  loop
    for T_i in $\{T\}$ **do**
      Update sample $\mathbf{w}_i$ from (10) by performing N_{traj}
HMC trajectories of length L and time step dt.
    end for
    for $i \leftarrow 1, N_T$ **do**
      Choose random adjacent T pair (T_j, T_{j+1}).
      Exchange $(\mathbf{w}_j, \mathbf{w}_{j+1})$ with probability
$$\min\left[1, e^{\left(\frac{1}{T_j} - \frac{1}{T_{j+1}}\right)\left(E_j - E_{j+1}\right)}\right].$$
    end for
  end loop
end procedure

At the start of the simulation the parameters of the neural network are initialised independently for each temperature using the following approach:

1. Draw $\mathbf{w}$ uniformly at random from the region $w_i[-\frac{1}{\sqrt{k_i}}, \frac{1}{\sqrt{k_i}}]$ with k_i as defined in Sec. III B.

2. Minimise the cost function (7) using Alg. 3. This minimisation avoids starting the dynamics from parameter values where the gradient is extremely large.

3. Perform a number of burn-in trajectories at the appropriate temperature.

Algorithm 3 Fast minimisation of the cost function (7).

---

procedure RMIN($\mathbf{w}$)
  $\mathbf{p} = 0$ ▷ Set initial momenta to zero.
  for $i = 1, N_{\mathrm{steps}}$ **do**
    $\mathbf{w}_{\mathrm{save}}, J_{\mathrm{save}} = \mathbf{w}, J(\mathbf{w}|D)$ ▷ Save lowest cost state.
    Propagate $(\mathbf{w}, \mathbf{p})$ through 1 Velocity Verlet time step, duration dt.
    if $J(\mathbf{w}|D) < J_{\mathrm{save}}$ **then**   ▷ Downwards step
      $dt = dt + 0.05$.   ▷ Slowly increase dt.
    else   ▷ Upwards step
      $\mathbf{p} = 0$ ▷ Zero momenta
      $\mathbf{w} = \mathbf{w}_{\mathrm{save}}$ ▷ Return to lowest cost $\mathbf{w}$
      $dt = dt * 0.95$ ▷ Rapidly decrease dt.
    end if
  end for
end procedure

Alg. 3 is related to the FIRE optimisation algorithm [3]. We found Alg. 3 to be extremely efficient for these simple networks, requiring just a few hundred steps to reach $E(D|\mathbf{w}) = 0$ in $M^{(3)}$.

Fig. 2 shows the trajectory of a RE-HMC simulation with $M^{(3)}$ and D_{500}. The parameters used are given in Table I. The typical value of E explores a narrow range which is specified by the temperature. At high T $M^{(3)}$
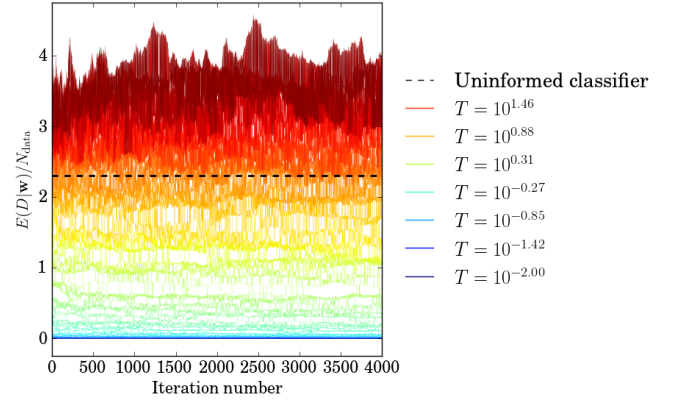
FIG. 2. Trajectory of a replica-exchange HMC Simulation for $M^{(3)}$ with D_{500}.

performs worse than an uninformed classifier that assigns equal probability to each digit class for every image.

TABLE I. RE-HMC Parameters in Figs. 2, 3 and 4.

PARAMETER	FIGS. 2 AND 3	FIG. 4
N_T	112	84
$T_{\min}$	10^{-2}	$10^{-\frac{3}{2}}$
$T_{\max}$	10^2	$10^{\frac{3}{2}}$
N_{traj}	10	10
L	100	100

D. RESULTS: THE BEST GENERALISATION IS NOT ALWAYS FOUND AT $T = 1$

There is broad agreement in the Bayesian learning community that sampling from the posterior leads to better generalisation than minimising the cost function. Our results strongly support this assertion. However, we also observe that improved performance can in general be obtained at temperatures other than $T = 1$, and that the optimal value of T depends on the network and the data set.

Fig. 3 shows the sampled average of the potential energy $\langle E(D|\mathbf{w}) \rangle_T$ for $M^{(3)}$ across a wide range of T, and using different sized training sets. The parameters of this calculation are shown in Table I.

In all data sets sampling from the posterior yields much improved generalisation as compared to classical optimisation. In this case, we show the result of optimisation from $\mathbf{w}$ chosen according to the standard initialisation described in Sec. III B and optimised using Alg. 3.

At low temperatures $M^{(3)}$ achieves perfect performance on all training sets. Conversely, clear minima are observed in the test error. For the smaller data sets, D_{50}

and D_{500}, the minimum test error does indeed occur in the region of $T \sim 1$. However, for the larger data set D_{5000} (which is still relatively small) that minimum occurs at $T > 1$.

Sampling at a range of temperatures to identify that with the lowest test error is the statistical equivalent of early stopping: at low temperatures the temperature-adjusted posterior lies close to the global minimum of $E(D|\mathbf{w})$ and over fits the noise in these small data sets. At very high T the network is essentially untrained, and between these two extremes the network learns, but not so much as to over fit the data. This leads to the observed minima.

At high T and for all data sets, $M^{(3)}$ performs worse than an "uninformed" classifier, which assigns an equal probability of 0.1 to each digit class, independent of the image. As the data set is increased, the temperature at which the network transitions to this state becomes higher. For the smallest data set D_{50} the network is already in this state at $T = 1$.

Finally we remark that in $M^{(3)}$, which is a small neural network, the transition from accurate to inaccurate classifier is gradual, and does not exhibit a sharp jump. A statistical physicist would say that no "first-order phase transition" is observed. If a sharp transition between accurate and inaccurate classifiers were observed then the sampler would need to make frequent transitions between the two states in order to obtain balanced sampling of temperature-adjusted posterior. These transitions would be unlikely to occur rapidly without specific assistance. Efficient sampling of the temperature-adjusted posterior would therefore require a specialised technique such as Metadynamics [2]. From this perspective, the absence of a first-order phase transition is an important result, since it makes sampling the temperature-adjusted posterior distribution relatively simple and inexpensive.

In order to check the generality of these observations we repeated the calculations shown in Fig. 3 for the neural network $M^{(3*)}$. The results for $M^{(3*)}$ are shown in Fig. 4. The parameters for this calculation can be found in Table I.

From Fig. 4 we see that the same trends are observed for the training error in both $M^{(3)}$ and $M^{(3*)}$. In particular, for $M^{(3*)}$ $T = 1$ does not lead to optimal generalisation. Rather the best generalisation is achieved at low T: no minimum in the test error is apparent across the temperatures studied. Nevertheless we anticipate a minimum at very low T. As in $M^{(3)}$, a transition is observed from relatively accurate to inaccurate classification as T is increased. Overall $M^{(3*)}$ generalises approximately as well as $M^{(3)}$ for the smallest data sets D_{50} and D_{500}, but for D_{5000} $M^{(3)}$ shows far better performance than $M^{(3*)}$. We anticipate this trend to continue, since the minimum training errors for any data set, obtained by $M^{(3)}$ and $M^{(3*)}$ are 0 and 1.46 respectively.
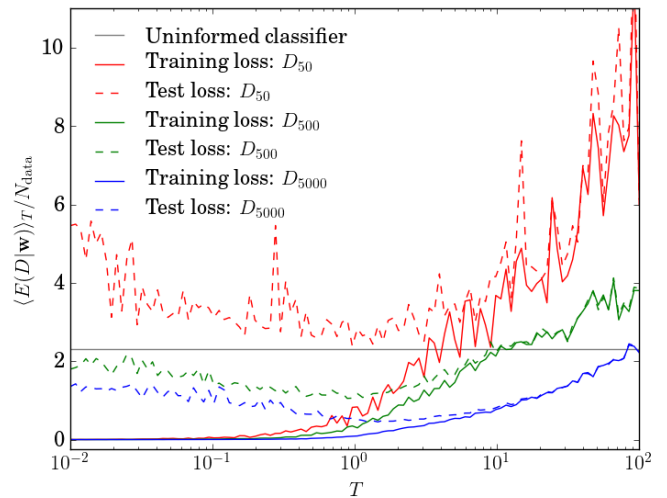

FIG. 3. Empirical $\langle E(D|\mathbf{w})\rangle_T$ for $M^{(3)}$ at different temperatures T. Sharp variations are due to sampling noise.
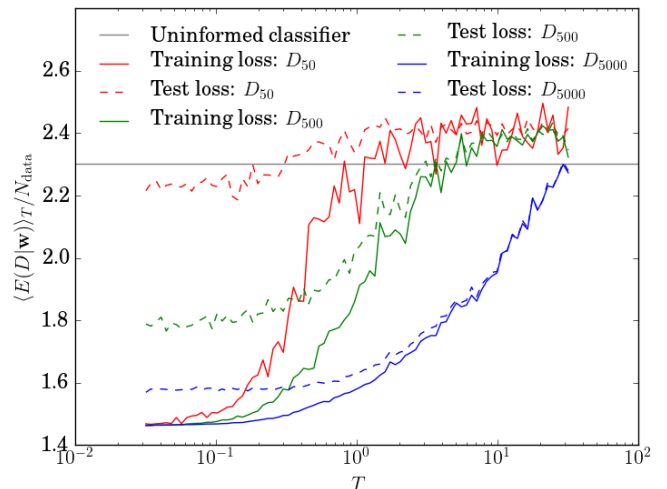

FIG. 4. Empirical $\langle E(D|\mathbf{w})\rangle_T$ for $M^{(3*)}$ across a range of temperatures T. Sharp variations are due to sampling noise.

E. SUMMARY

We found that, for $M^{(3)}$, which is a typical neural network classifier, sampling the posterior distribution does indeed give lower test error than standard optimisation of the cost function. However, even better performance can in general be obtained by sampling the temperature-adjusted posterior across a wide range of temperatures to identify that with the lowest test error. Furthermore, quite different values of T are appropriate to different neural networks and data sets.

IV. BAYESIAN MODEL SELECTION FOR DEEP (AND SHALLOW) NEURAL NETWORKS WITH THERMODYNAMIC INTEGRATION

In this section we demonstrate how thermodynamic integration (TI), can be used to perform Bayesian model selection between two neural network models: one deep and one shallow network. We begin with a quick overview of Bayesian model selection before describing TI, the details of our calculation, and presenting the results.

A. NEURAL NETWORK ARCHITECTURES AND PRIORS

In this section we reuse the neural network $M^{(3)}$, described in Sec. III A. We shall compare $M^{(3)}$ to a second neural network, $M^{(1)}$, which is similar to $M^{(3)}$ except that it only contains a single hidden layer of 40 logistic neurons.

The priors of these two networks are different because $M^{(1)}$ has fewer parameters than $M^{(3)}$. Standard training of $\mathbf{w}$ for $M^{(1)}$ using Alg. 3 gave $\max_i \left(w_i \div \frac{1}{\sqrt{k_i}} \right) = 33.3$ at $E(D_{500}|\mathbf{w}) = 0$. This is similar to the value 30.3 obtained for $M^{(3)}$. Consequently we used the same width factor 50 to define the prior for $M^{(1)}$, as in $M^{(3)}$, thereby assigning the bounds of the prior σ according to (11).

B. INTRODUCTION TO BAYESIAN MODEL SELECTION

Bayesian model selection for neural networks was first formulated by David MacKay [10, 11]. Imagine we have two different machine learning models M_1 and M_2, and a data set $D = \{\mathbf{x}, t_i\}$, or in vector notation $D = (\mathbf{x}, \mathbf{t})$ where $\mathbf{x}$ is a matrix of all the input data and $\mathbf{t}$ a vector of the corresponding labels.

In Bayesian model selection one computes and compares the probabilities of the models given the data, $\text{prob}(M|\mathbf{x}, \mathbf{t})$. A true Bayesian would believe both models to different extents, according to their probabilities. However it is common practice to choose the model with highest probability, discarding all others.

The ratio of the probabilities for M_1 and M_2 can be found as follows

$$\text{prob}(M, \mathbf{t}|\mathbf{x}) = \text{prob}(M|\mathbf{x}, \mathbf{t})\,\text{prob}(\mathbf{t}|\mathbf{x}) \tag{12}$$

$$= \text{prob}(\mathbf{t}|\mathbf{x}, M)\,\text{prob}(M|\mathbf{x}) \tag{13}$$

$$\Rightarrow \frac{\text{prob}(M_1|\mathbf{x}, \mathbf{t})}{\text{prob}(M_2|\mathbf{x}, \mathbf{t})} = \frac{\text{prob}(\mathbf{t}|\mathbf{x}, M_1)}{\text{prob}(\mathbf{t}|\mathbf{x}, M_2)} \frac{\text{prob}(M_1)}{\text{prob}(M_2)} \tag{14}$$

The term furthest to the right in (14) is the ratio of our priors for the models and represents our initial bias towards either model. Typically we have no initial preference between M_1 and M_2, in which case this term is equal to 1.

In theory one can calculate $\text{prob}(\mathbf{t}|\mathbf{x}, M)$ as follows

$$\text{prob}(\mathbf{t}|\mathbf{x}, M) = \int d\mathbf{w}\,\text{prob}(\mathbf{t}, \mathbf{w}|\mathbf{x}, M) \tag{15}$$

$$= \int d\mathbf{w}\,\text{prob}(\mathbf{t}|\mathbf{w}, \mathbf{x}, M)\,\text{prob}(\mathbf{w}) \tag{16}$$

$$= \int d\mathbf{w}\,e^{-J(\mathbf{w}|D)} \tag{17}$$

In (17) we have made a substitution from (4).

C. METHOD: THERMODYNAMIC INTEGRATION

In this section we describe how (17) can be calculated using TI. The TI algorithm we describe here was originally introduced in [5]. This approach asserts that the integrand of (17) is approximately unimodal, with a single maximum in the region surrounding a particular local minimum of $J(\mathbf{w}|D)$, at $\mathbf{w}_0$.

We then sample $e^{-J(\mathbf{w}|D)}$ in the region of $\mathbf{w}_0$ and fit an approximate quadratic form for $J(\mathbf{w}|D)$

$$\frac{1}{2}(\mathbf{w} - \mathbf{w}_0)^{\mathsf{T}}\mathbf{k}(\mathbf{w} - \mathbf{w}_0) \simeq J(\mathbf{w}|D). \tag{18}$$

The evidence for that quadratic form

$$Z_0 = \int_{|\mathbf{w}| < \frac{\sigma}{2}} e^{-(\mathbf{w} - \mathbf{w}_0)^{\mathsf{T}} \frac{\mathbf{k}}{2}(\mathbf{w} - \mathbf{w}_0)} \tag{19}$$

is known exactly, provided $\mathbf{k}$ is diagonal. For this reason we fit a diagonal matrix $\mathbf{k}$ with

$$k_{ii} = \frac{1}{\langle (w_i - w_{0,i})^2 \rangle}. \tag{20}$$

The average $\langle (w_i - w_{0,i})^2 \rangle$ is taken by sampling from the posterior distribution (2). If one is using a uniform prior then it is important that, for this averaging alone, we should instead sample from $\text{prob}(\mathbf{w}) \propto e^{-E(D|\mathbf{w})}$ without applying hard boundaries to $\mathbf{w}$. This ensures that the approximation (18) is a good fit to $J(\mathbf{w}|D)$.

Next in TI one sequentially samples a series of N_{bridge} bridging potentials constructed to interpolate between $J(\mathbf{w}|D)$ and the approximation (18).

$$J_{\text{bridge}}(\mathbf{w}|D, \lambda) = (1 - \lambda)(J(\mathbf{w}|D) - J(\mathbf{w}_0|D))$$
$$+ \lambda(\mathbf{w} - \mathbf{w}_0)^{\mathsf{T}} \frac{\mathbf{k}}{2}(\mathbf{w} - \mathbf{w}_0) + J(\mathbf{w}_0|D). \tag{21}$$

Thus $J_{\text{bridge}}(\mathbf{w}|D, \lambda = 0) = J(\mathbf{w}|D)$ and $J_{\text{bridge}}(\mathbf{w}|D, \lambda = 1) = (\mathbf{w} - \mathbf{w}_0)^{\mathsf{T}} \frac{\mathbf{k}}{2}(\mathbf{w} - \mathbf{w}_0)$.

Defining

$$F = -\log \int_{|\mathbf{w}| < \frac{\sigma}{2}} e^{-J(\mathbf{w}|D)} \tag{22}$$

$$F_0 = -\log Z_0 \tag{23}$$

it can be shown [6] that

$$F = F_0 + \int_{\lambda=1}^{\lambda=0} d\lambda \left\langle \frac{\partial J_{\text{bridge}}(\mathbf{w}|D,\lambda)}{\partial \lambda} \right\rangle_\lambda, \qquad (24)$$

where

$$\left\langle \frac{\partial J_{\text{bridge}}(\mathbf{w}|D,\lambda)}{\partial \lambda} \right\rangle_\lambda =$$

$$\left\langle \sum_{i=1}^d \frac{k_{ii}}{2}(w_i - w_{0,i})^2 - [J(\mathbf{w}|D) - J(\mathbf{w}_0|D)] \right\rangle_\lambda. \quad (25)$$

Averaging in (25) occurs over the distribution $\text{prob}(\mathbf{w}) \propto e^{-J_{\text{bridge}}(\mathbf{w}|D,\lambda)}$.

In this way, TI evaluates the evidence for a model,

$$\text{prob}(\mathbf{t}|\mathbf{x}, M) = e^{-F} \qquad (26)$$

as a correction to Z_0.

1. Experimental Setup

We reused our HMC implementation for these calculations. HMC trajectories are all performed using $J_{\text{bridge}}(\mathbf{w}|D,\lambda)$. Before fitting $\mathbf{k}$, dt is adjusted to obtain an acceptance rate in the range (0.6–0.7). For fitting $\mathbf{k}$ we set $\lambda = 0$ and performed 1000 trajectories of burn-in each $L = 100$ steps long, then a further 1000 trajectories of the same length to obtain an estimate of $\mathbf{k}$ according to (20).

We used 100 bridging distributions (102 distributions in total). These distributions are sampled sequentially, from $\lambda = 0$ to $\lambda = 1$. To evaluate each expectation (25) 100 trajectories of HMC with $L = 100$ are performed as burn-in, before a further 100 trajectories of the same length to collect samples. We updated dt after every 10 bridging distributions to recover an acceptance rate in the range (0.6–0.7). Additional trajectories, not included in any averaging or burn-in, are performed in order to set dt. The integration in (24) is performed using Simpson's rule.

D. RESULTS: BAYESIAN SELECTION BETWEEN $M^{(1)}$ AND $M^{(3)}$

In this section we perform a Bayesian model selection between the neural networks $M^{(3)}$ which has three hidden layers, and $M^{(1)}$ which has one. These networks are fully described in Sec. III A.

To obtain $\mathbf{w}_0$ for each model we followed the standard training procedure: weights and biases are initialised uniformly at random inside the region $w_i \in [-\frac{1}{\sqrt{k_i}}, \frac{1}{\sqrt{k_i}}]$ where k_i is the corresponding fan in, as described in Sec. III B. The priors for the two models are described in Sec. IV A. All calculations are performed with data set D_{500} (see Sec. III A).

For the uniform prior (6)

$$\text{prob}(\mathbf{t}|\mathbf{x}, M) = \int d\mathbf{w} e^{-E(D|\mathbf{w})} \times \frac{1}{\prod_{i=1}^d \sigma_i}. \qquad (27)$$

Using TI we obtain

$$\log\left(\int_{|\mathbf{w}|<\frac{\sigma}{2}} d\mathbf{w} e^{-E^{(3)}(\mathbf{w}|D)}\right) = (26.48 \pm 0.17) \times 10^3$$

$$\log\left(\int_{|\mathbf{w}|<\frac{\sigma}{2}} d\mathbf{w} e^{-E^{(1)}(\mathbf{w}|D)}\right) = (19.79 \pm 0.01) \times 10^3$$

The uncertainties shown correspond to the standard deviations of 28 independent calculations for each model. Additionally, for our prior we have

$$\log\left(\prod_{i=1}^d \sigma_i^{(3)}\right) = 28.960 \times 10^3$$

$$\log\left(\prod_{i=1}^d \sigma_i^{(1)}\right) = 19.946 \times 10^3$$

Thus we obtain

$$\frac{\text{prob}(M^{(3)}|\mathbf{x}, \mathbf{t})}{\text{prob}(M^{(1)}|\mathbf{x}, \mathbf{t})} = \frac{e^{(26.475 \pm 0.173) \times 10^3}}{e^{(19.793 \pm 0.013) \times 10^3}} \times \frac{e^{19946}}{e^{28960}} \qquad (28)$$

$$= e^{6682} \times e^{-9014} \qquad (29)$$

$$\simeq e^{-2332}$$

Bayes theorem clearly favours the shallow network $M^{(1)}$ over the deep network $M^{(3)}$. For larger data sets, this preference may well be reversed, but for D_{500} the accuracy of $M^{(3)}$ is not enough to justify its larger parameter space.

V. CONCLUSION

We have seen that for a number of small MNIST training sets sampling from the posterior does indeed lead to markedly improved generalisation than standard optimisation of the cost function. Even better generalisation can often be obtained by sampling the temperature-adjusted posterior (10) at $T \neq 1$. The optimal temperature, with the lowest test error, depends on the network and the data set. We saw that for network parameters sampled from the temperature-adjusted posterior across a range of temperatures, neural network based classifiers exhibit a transition between accurate and inaccurate classifiers, although no sharp "phase transition" occurs between these states. The absence of such a sharp transition makes sampling the temperature-adjusted posterior relatively simple and inexpensive. Finally we showed how thermodynamic integration can be used to perform model selection on deep and shallow neural networks, avoiding the need to ever calculate or invert the Hessian of the cost function.

[1] Alder, Berni J, & Wainwright, Thomas Everett. 1959. Studies in molecular dynamics. i. general method. *The journal of chemical physics*, **31**(2), 459–466.

[2] Barducci, Alessandro, Bonomi, Massimiliano, & Parrinello, Michele. 2011. Metadynamics. *Wiley interdisciplinary reviews: Computational molecular science*, **1**(5), 826–843.

[3] Bitzek, Erik, Koskinen, Pekka, Gähler, Franz, Moseler, Michael, & Gumbsch, Peter. 2006. Structural relaxation made simple. *Phys. rev. lett.*, **97**(Oct), 170201.

[4] Duane, Simon, Kennedy, Anthony D, Pendleton, Brian J, & Roweth, Duncan. 1987. Hybrid monte carlo. *Physics letters b*, **195**(2), 216–222.

[5] Frenkel, Daan, & Ladd, Anthony JC. 1984. New monte carlo method to compute the free energy of arbitrary solids. application to the fcc and hcp phases of hard spheres. *The journal of chemical physics*, **81**(7), 3188–3193.

[6] Frenkel, Daan, & Smit, Berend. 2001. *Understanding molecular simulation: from algorithms to applications*. Vol. 1. Elsevier.

[7] Horowitz, Alan M. 1991. A generalized guided monte carlo algorithm. *Physics letters b*, **268**(2), 247 – 252.

[8] Jenkins, R., Curotto, E., & Mella, Massimo. 2013. Replica exchange with smart monte carlo and hybrid monte carlo in manifolds. *Chemical physics letters*, **590**, 214 – 220.

[9] Kofke, David A. 2002. On the acceptance probability of replica-exchange monte carlo trials. *The journal of chemical physics*, **117**(15), 6911–6914.

[10] MacKay, David J.C. 1991. Bayesian interpolation. *Neural computation*, **4**, 415–447.

[11] MacKay, David JC. 1992a. The evidence framework applied to classification networks. *Neural computation*, **4**(5), 720–736.

[12] MacKay, David JC. 1992b. A practical bayesian framework for backpropagation networks. *Neural computation*, **4**(3), 448–472.

[13] Metropolis, Nicholas, Rosenbluth, Arianna W, Rosenbluth, Marshall N, Teller, Augusta H, & Teller, Edward. 1953. Equation of state calculations by fast computing machines. *The journal of chemical physics*, **21**(6), 1087–1092.

[14] Name, Double Blind. 2019. *Repository*. `https://github.com/REDACTED/REDACTED`.

[15] Neal, Radford M. 2012. *Bayesian learning for neural networks*. Vol. 118. Springer Science & Business Media.

[16] Neal, Radford M, *et al.* . 2011. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, **2**(11), 2.

[17] Smith, Samuel L, & Le, Quoc V. 2017. A bayesian perspective on generalization and stochastic gradient descent. *arxiv preprint arxiv:1710.06451*.

[18] Sohl-Dickstein, Jascha, Mudigonda, Mayur, & DeWeese, Michael R. 2014. Hamiltonian monte carlo without detailed balance. *arxiv preprint arxiv:1409.5191*.

[19] Sugita, Yuji, & Okamoto, Yuko. 1999. Replica-exchange molecular dynamics method for protein folding. *Chemical physics letters*, **314**(1), 141 – 151.

[20] Sugita, Yuji, Kitao, Akio, & Okamoto, Yuko. 2000. Multidimensional replica-exchange method for free-energy calculations. *The journal of chemical physics*, **113**(15), 6042–6051.

[21] Swendsen, Robert H., & Wang, Jian-Sheng. 1986. Replica monte carlo simulation of spin-glasses. *Phys. rev. lett.*, **57**(Nov), 2607–2609.

[22] Swope, William C, Andersen, Hans C, Berens, Peter H, & Wilson, Kent R. 1982. A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters. *The journal of chemical physics*, **76**(1), 637–649.

[23] Venditto, J.G., Wolf, S., Curotto, E., & Mella, Massimo. 2015. Replica exchange hybrid monte carlo simulations of the ammonia dodecamer and hexadecamer. *Chemical physics letters*, **635**, 127 – 133.