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: January 2019)

ABSTRACT

In this blog post I examine the Bayesian formulation of neural networks due to MacKay [1]. I consider how the noise level (or "temperature") T in that formulation affects the posterior distribution. I use this lense to critique a popular consensus in machine learning that network parameters sampled from the posterior lead to better generalisation than those due to full batch training. While this may be the case for large data sets, my results show that for small data sets, just the opposite is true: classical training can lead to significantly improved generalisation because networks sampled from the posterior tend to display high classification error on both the training and test sets. It is demonstrated that a "low temperature" distribution derived from the posterior is more useful than the posterior itself for ML, and that in general, ML models exhibit "phase transitions" between accurate and inaccurate models when "heated". In particular, I highlight an approximate method for performing model selection on deep NNs.

I. STRUCTURE OF THIS BLOG POST

1. Outline of the Bayesian formulation of feed forward neural networks.

2. An analogy with physics, introducing an "energy" function for the network, and a "temperature", which controls the noise level.

3. Results section

4. Sampling methods (code in this repository). I built an HMC version of an accelerated sampling method from materials simulation, called Replica Exchange Molecular Dynamics which simulataneously explores the behaviour of a model accross a wide range of temperatures.

* rjnbaldock@gmail.com

II. BAYESIAN FORMULATION OF FEED FORWARD NEURAL NETWORKS AND TRAINING

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 Bayesian probability, assigned by the network, to the assertion that the input belongs to each individual class. We can write this as $\text{prob}\,(q|\mathbf{x},\mathbf{w})$ 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 eural 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 $\text{prob}\,(q=t_i|\mathbf{x}_i,\mathbf{w})$. This probability is a function of the weights $\mathbf{w}$. The probability that the network classfies the complete dataset correctly is called the *likelihood* of the data and is given by $\prod_i \text{prob}\,(q=t_i|\mathbf{x}_i,\mathbf{w})$.

If we assign a Gaussian prior with standard deviation σ to the parameters $\mathbf{w}$ then the *posterior* probability of the data is given by

$$\text{prob}\,(\mathbf{w}|D)=\prod_i \text{prob}\,(q=t_i|\mathbf{x}_i,\mathbf{w})e^{-\frac{\mathbf{w}^2}{2\sigma^2}}. \quad (1)$$

The posterior probability of the weights are therefore maximised when we minimise

$$J(\mathbf{w}|D)=-\log\text{prob}\,(\mathbf{w}|D) \quad (2)$$

$$=-\sum_i \log\left[\text{prob}\,(q=t_i|\mathbf{x}_i,\mathbf{w})\right]+\frac{\mathbf{w}^2}{2\sigma^2} \quad (3)$$

which can immediately be recognised as the cross entropy loss function with L2 regularisation. The minimum of (2) is called the maximum a posteriori solution.

In this blogpost I will be using a uniform prior for $\mathbf{w}$

$$\text{prob}\,(\mathbf{w})=\begin{cases}\frac{1}{\prod_{i=1}^d \sigma_i}, & |w_i|<\sigma_i/2\,\forall\, i,\\ 0, & \text{Elsewhere.}\end{cases} \quad (4)$$

Within the bounds of this prior cost function is given by

$$J(\mathbf{w}|D)=-\sum_i\left(\log\left[\text{prob}\,(q=t_i|\mathbf{x}_i,\mathbf{w})\right]-\log\sigma_i\right). \quad (5)$$

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

III. INTRODUCING ENERGY AND TEMPERATURE

One may define the "potential energy" over the parameters of the network [1]

$$E(\mathbf{w}|D) = -\sum_i \log\left[\text{prob}\left(q = t_i|\mathbf{x}_i, \mathbf{w}\right)\right] \tag{6}$$

and a "thermal" posterior distribution

$$\text{prob}\left(\mathbf{w}|T, D\right) = e^{-\frac{1}{T}E(\mathbf{w}|D)}\,\text{prob}(\mathbf{w}). \tag{7}$$

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

For this blog post, it is important to imagine how the thermal posterior behaves as we vary T. The distribution (7) tends towards the prior distribution for $T \to \infty$, and is equal to the true posterior distribution (1) for $T = 1$. At $T < 1$ (low temperature) the thermalised posterior (7) is concentrated around the maximum a posteriori solution (the minimum of (2)).

A. Temperature vs batch size

It has recently been discovered that mini-batch learning improves generalisation. There has been a lot of 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 though I wanted to explore instead using the temperature to control the noise in full batch training and as an alternative to early stopping.

IV. RESULTS

A. Sampling the posterior for small data sets

B. Phase transitions in parameter space

C. Hessian free Bayesian model selection for deep (or shallow) neural networks

I'll start with a quick overview of Bayesian model selection. Then I'll introduce the Laplace approximation (approximating a distribution by a Gaussian fitted to a given maximum of the true distribution). Finally, I'll show how you can compute the Bayesian evidence for that Gaussian distribution, by finding the determinant of the Hessian without calculating the Hessian directly.

1. Bayesian Model Selection

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 statistics we are able to calculate the probability of each model given the data, $\text{prob}\left(M|\mathbf{x}, \mathbf{t}\right)$. In Bayesian model selection you choose the model with the highest probability. Actually you never choose, you believe them all do different extents, according to their probabilities.

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

$$\text{prob}\left(M, \mathbf{t}|\mathbf{x}\right) = \text{prob}\left(M|\mathbf{x}, \mathbf{t}\right)\text{prob}\left(\mathbf{t}|\mathbf{x}\right) \tag{8}$$

$$= \text{prob}\left(\mathbf{t}|\mathbf{x}, M\right)\text{prob}\left(M|\mathbf{x}\right) \tag{9}$$

$$\Rightarrow \frac{\text{prob}\left(M_1|\mathbf{x}, \mathbf{t}\right)}{\text{prob}\left(M_2|\mathbf{x}, \mathbf{t}\right)}\frac{\text{prob}\left(\mathbf{t}|\mathbf{x}\right)}{\text{prob}\left(\mathbf{t}|\mathbf{x}\right)} = \frac{\text{prob}\left(\mathbf{t}|\mathbf{x}, M_1\right)}{\text{prob}\left(\mathbf{t}|\mathbf{x}, M_2\right)}\frac{\text{prob}\left(M_1\right)}{\text{prob}\left(M_2\right)} \tag{10}$$

$$\Rightarrow \frac{\text{prob}\left(M_1|\mathbf{x}, \mathbf{t}\right)}{\text{prob}\left(M_2|\mathbf{x}, \mathbf{t}\right)} = \frac{\text{prob}\left(\mathbf{t}|\mathbf{x}, M_1\right)}{\text{prob}\left(\mathbf{t}|\mathbf{x}, M_2\right)}\frac{\text{prob}\left(M_1\right)}{\text{prob}\left(M_2\right)} \tag{11}$$

The last term on the right in (11) is the ratio of our priors for the models. Typically we might initially have no preference between M_1 and M_2, in which case this last term is equal to 1.

In theory one can calculate the "marginal likelihood" of the model, $\text{prob}\left(\mathbf{t}|\mathbf{x}, M\right)$, as follows

$$\text{prob}\left(\mathbf{t}|\mathbf{x}, M\right) = \int d\mathbf{w}\text{prob}\left(\mathbf{t}, \mathbf{w}|\mathbf{x}, M\right) \tag{12}$$

$$= \int d\mathbf{w}\text{prob}\left(\mathbf{t}|\mathbf{w}, \mathbf{x}, M\right)\text{prob}\left(\mathbf{w}\right) \tag{13}$$

$$= \int d\mathbf{w}e^{-J(\mathbf{w}|D)} \tag{14}$$

In (14) we have made the connection with (5). There are lots of clever ways to calculate this integral. For deep neural networks it is almost always highly multimodal.

2. The Laplace approximation

One way to compute integral (13) is to make the Laplace approximation. We move to a local (ideally global) maxmimum of the posterior $\text{prob}\left(\mathbf{t}|\mathbf{w}, \mathbf{x}, M\right)\text{prob}\left(\mathbf{w}|M\right)$, which is at $\mathbf{w}_{\text{MP}}$. We approximate the integral (13) as the height of the peak of the integrand posterior, times its parameter space volume $\prod_{i=1}^{d}\sigma_{\text{MP},i}$ where $\{\sigma_{\text{MP},i}^{-2}\}$ are the eigenvalues of the Hessian of $J\left(\mathbf{w}_{\text{MP}}|D\right)$ calculated at $\mathbf{w}_{\text{MP}}$. In this case we can approximate the integral (13) as

$$\text{prob}\left(\mathbf{t}|\mathbf{x}, M\right) \simeq e^{-J(\mathbf{w}_{\text{MP}}|D)}\prod_{i=1}^{d}\sigma_{\text{MP},i} \tag{15}$$

The value of $J(\mathbf{w}_{\rm MP}|D)$ is directly obtained at the conclusion of minimisation. All we need is the $\prod_{i=1}^{d}\sigma_{{\rm MP},i}$ but we don't want to calculate or diagonalise the Hessian. The cost of doing so is cubic in the number of parameters, which could easily reach the millions for a production model.

3. *Hessian free estimation of the determinant of the Hessian*

Since we have assumed that the cost function is quadratic around $\mathbf{w}_{\rm MP}$ we can make use of a result from statistical mechanics, to calculate $\prod_{i=1}^{d}\sigma_{{\rm MP},i}$. Assuming the quadratic approximation does to the cost function does not place a lot of weight outside the bounds of the uniform prior (reasonable since there are many minima within that space) then we oobtain

$$\prod_{i=1}^{d}\sigma_{{\rm MP},i}^{2}\simeq\frac{d\langle E\rangle}{dT} \tag{16}$$

where the average is taken over the thermal posterior (7). This is almost an equality in the limit of small T. I need to check but I think that for a Gaussian prior I can obtain a similar result $\prod_{i=1}^{d}\sigma_{{\rm MP},i}^{2}\simeq\frac{d\langle J\rangle}{dT}$ if I also raise the prior to the power $1/T$ in (7).

I applied a new formulation of Replica Exchange Molecular Dynamics with Hamiltonian Monte Carlo which I developed (see Section V), to two NN classifiers. My approach allowed me to extract curves for $E(T)$. These networks and the MNIST data have already been described in Section IV A. I had 500 data points for each character type (5000 in total). I used 40 logistic neurons per hidden layer, with a softmax on the final layer, and investigated netoworks with one and three hidden layers. Following the standard procedure I initialised each weight and bias uniformly at random inside the region $[-\frac{1}{\sqrt{k}},\frac{1}{\sqrt{k}}]$ where k is the fan in of the neuron to which the weight points. The fan in includes the bias. The prior space was of the same shape but 1000 times wider. This was chosen because full batch optimisation of the cost function made some weights nearly this large. I believe three layers counts as a deep network. The results are shown in Figure 1.

Taking the minimum Energy values from these same datasets I obtain

$$\frac{\text{prob}\,(1\,\text{H Layer})}{\text{prob}\,(3\,\text{H Layers})}=\frac{e^{-E_1}}{e^{-E_2}}\frac{\prod_{i=1}^{d}\sigma_{prior,\,i}^{(3\,layers)}}{\prod_{i=1}^{d}\sigma_{prior,\,i}^{(1\,layer)}}\frac{\prod_{i=1}^{d}\sigma_{{\rm MP},i}^{(1\,layer)}}{\prod_{i=1}^{d}\sigma_{{\rm MP},i}^{(3\,layers)}} \tag{17}$$

$$\simeq\frac{e^{-7346}}{e^{-7313}}\frac{e^{2435060}}{e^{1410560}}\frac{\sqrt{402}}{\sqrt{233}} \tag{18}$$

$$\simeq 1\times 10^{444920}. \tag{19}$$

We can see that the "Occam's razor" term $\frac{\prod_{i=1}^{d}\sigma_{{\rm MP},i}}{\prod_{i=1}^{d}\sigma_{prior,\,i}}$

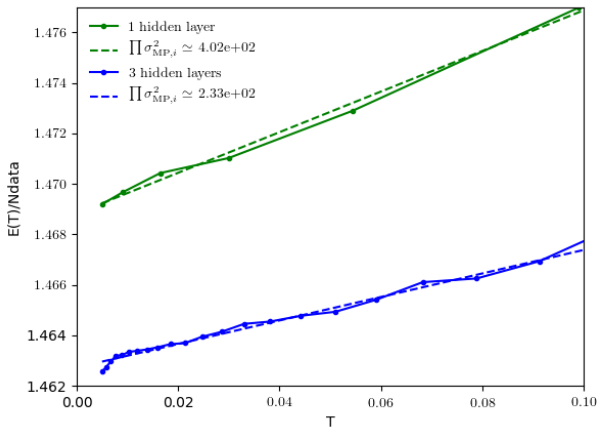

FIG. 1. Hessian free estimation of the determinant of the Hessian of the cost function. There are 5000 data points, so the y axis has been scaled by a factor of 1/5000, and the difference in the energy is in fact 33. Interestingly the hessians of the cost functions have almost equal determinant.

showed enormous preference for the 1 layer network, which did comparably well on the classification task.

What might well make this a useful approach is that relative prior probabilities of models can differ so enormously. Finally the sensitivity to $\prod_{i=1}^{d}\sigma_{{\rm MP},i}$ is low: an answer within an order of magnitude (or two!) may well be perfectly acceptable.

D. Discussion

V. SAMPLING METHODS

I developed a Hamiltonian Monte Carlo (HMC) formulation of a technique called Replica Exchange Molecular Dynamics (REMD) for use with NN models and study some classifiers for MNIST digits. REMD is a method for exploring the behaviour of a sampler simulataneously accross a range of temperatures. Samplers periodically attempt to exchange their current points in parameter space with samplers at neighbouring temperatures. This accelerates parameter space sampling at low temperatures because each set of coordinates spends some of the time at high temperatures where they move around much more rapidly.

A. Conceptual introduction

Critically the parameter swaps are done in a way as to preserve the joint distribution of the samplers. Put another way, each sampler is sampling from the correct distribution all of the time, including immediately after

swapping parameter points with samplers at neighbouring temperatures.

Replica Exchange is a Markov chain Monte Carlo (MCMC) algorithm for efficient sampling from a complex and highly multimodal distribution, and was developed in the field of molecular simulation. The typical usage case of replica exchange corresponds to sampling a thermal posterior distribution similar to (7) but with a uniform prior, rather than a Gaussian prior more typical in machine learning.

The posterior distribution ($\beta = 1$) is typically highly multimodal and therefore difficult to explore with MCMC samplers, which become trapped in poor local minima of the energy function $E(\mathbf{w}|D)$ and so require long periods of computer time to equilibrate. Since in (2) the energy $E(\mathbf{w}|D)$ is multiplied by $\beta = \frac{1}{T}$, at high temperatures the cost function (2) becomes much smoother, and the barriers around local minima are reduced by a factor of β. Therefore MCMC samplers at high T move much more rapidly through the parameter space. Indeed, for a uniform prior, the time taken for the sytem to escape a local minimum of the cost function is proportional to τ_0^β where τ_0 is proportional to the mean escape time at $\beta = 1$. In physical science this is called the Arhenius Law **SPELLING?**.

The essential insight of Replica Exchange is that we can appropriate this rapid exploration of parameter space at high temperature, for lower temperatures as well, by setting up a ladder of simulations at different temperatures (termed replicas), and occasionally attempting additional MCMC steps where we try to exchange the current parameter values between adjacent temperatures. These additional MCMC steps are performed in such a way as to ensure that the total simulation correctly samples the thermal posterior (7) at every temperature, all of the time.

B. Overview of MD and MCMC

Traditionally in atomistic simulation, one would use Molecular Dynamics (MD) to sample the parameter space, according to a posterior distribution with a uniform prior. The posterior distribution with uniform prior is in fact just the likelihood distribution, as we can see from (1).

In Molecular Dynamics we continuously evolve the parameters exactly as a small ball bearing would move, were it rolling around on the energy surface $E(\mathbf{w}|D)$ according to Newton's Laws of Motion. Assuming that the sampler is ergodic (that in principle it can reach all points in parameter space), these samplers are proven to sample from the likelihood function in the limit of inifinite computer time. It is also possible to use standard MCMC (also called Metropolis Monte Carlo) in place of Molecular Dynamics. In this case the total algorithm is instead known as "Parallel Tempering".

In contrast, in standard MCMC one evolves the parameters by proposing a series of random moves with no memory of the previous trajectory. The exploration of MCMC samplers is thus a diffusive motion, even on the timescale of a single move. (Strictly speaking MD is diffusive too, but the timescale over which the sampler's momentum become completely decorellated, and its motion becomes diffusive, is far longer.) One proposes these random changes according to a probability distribution specified by the MCMC practitioner. If this proposal distribution is symetrical, so that the probability of the reverse proposal is always equal to the probability of proposing the forward move, then we accept these proposals with probability

$$\text{prob}(\text{old} \to \text{new}) = \min(1, e^{-\beta(E(\mathbf{w}_{\text{new}} - E(\mathbf{w}_{\text{old}})))}). \quad (20)$$

If the energy of the new parameters is lower than the old then the move is accepted. If the energy of the new parameters is higher than the old then the move is accepted with an exponentially dereasing probability, depending on the energy change. It can quite straightforwardly be proven that, for simple choices of the proposal distribution, this leads to ergodic sampling of the parameter space according to the likeihood function (CITE FRENKEL SMIDT).

The advantages of using Newton's Laws of Motion to evolve the sampler, rather than the diffusive random motion of Metropolis Monte Carlo, are apparent when we imagine the small ball bearing rolling around on the energy surface according to Newton's Laws of Motion. In Molecular Dynamics, the sampler develops a forward momentum which carries it rapidly along valleys in the energy surface $E(\mathbf{w}|D)$. In high dimension parameter spaces, which is almost always the situation for neural network models, there are an enormously large number of "bad directions" where the gradient of the energy surface is extremely large and the energy increases rapidly. If, as we almost always do in high dimensional spaces, we choose a bad direction for the random move then the energy increases sharply and the move is almost always rejected, according to (20). The solution to this problem is simply to make very small moves. Therefore, not only does an MCMC sampler have no forward momentum along valleys of the energy function, but it takes very small moves.

VI. DETAILS OF MY REPLICA EXCHANGE ALGORITHM

A. An HMC algorithm

I used Hamiltonian Monte Carlo (HMC, which is also called Hybrid Monte Carlo) as a replacement for MD.**CITE NEAL.** HMC represents the unification of MD and MCMC into a single technique. HMC can be considered a form of MCMC where the proposal distribution is highly non-trivial: it the set of states reached

from the initial parameters by a relatively short MD trajectory with random initial momenta. The duration of these short trajectories should ideally be set to be slightly shorter than the typical timescale on which the momenta of the sampler become uncorellated: that is, slightly shorter than the timescale on which the MD dynamics becomes diffusive. On timescales shorter than that the motion of the sampler is faster than diffusive. In this way the motion of a well tuned HMC sampler can be nearly as efficient as MD in terms of phase space exploration. HMC has the advantage of allowing us to combine the sampling with other Monte Carlo (MC) moves without compromising mathematical rigour, and therefore seemed a natural tool for the task. Indeed, we will see that direct MD proved inefficient for the thermalised posterior distribution (7) in high dimensions and at high temperature, but HMC allowed me to achieve correct and efficient sampling of that distribution.

To be useful in HMC the MD propagation algorithm employed must be reversible, symplectic, and phase space volume conserving. There is not space here to go into the details of this, but many standard integrators, including for example the Velocity Verlet scheme and the Leapfrog scheme are suitable candidates. A thorough treatement of the topic is given in **MY THESIS, NEAL** among others.

The aim of my HMC sampler is to sample from the thermalised posterior (7) at a particular value of β. I found that MD has a surprising difficulty sampling the cost function (2). The combination of the sharply increasing quadratic potential at large $|\mathbf{w}|$ and the relatively large gradients of $E(\mathbf{w}|D)$ make it difficult to perform MD on this surface: integration errors accumulate when the system moves inwards at speed from large $|\mathbf{w}|$, since the timestep appropriate to the two potentials is rather different. This poor integration leads to low acceptance rates for the trajectories, and in turn it is required to use small time steps, which is inefficient. My solution was to produce samples with dynamics on the energy surface alone, and then incorporate the prior into the acceptance probability for the trajectory. That is, I accepted trajectories with a probability

$$\begin{aligned}\text{prob}(\mathbf{w}_{\text{old}} \to \mathbf{w}_{\text{new}}) =& \frac{\text{prob}(\mathbf{w}_{\text{new}})}{\text{prob}(\mathbf{w}_{\text{old}})} && (21)\\ =& \frac{e^{-\beta E(\mathbf{w}_{\text{new}}) - \frac{\mathbf{w}_{\text{new}}^2}{2\sigma^2}}}{e^{-\beta E(\mathbf{w}_{\text{old}}) - \frac{\mathbf{w}_{\text{old}}^2}{2\sigma^2}}} && (22)\\ =& e^{-\beta(E(\mathbf{w}_{\text{new}}) - E(\mathbf{w}_{\text{old}}))} && (23)\\ & \times e^{-\frac{1}{2\sigma^2}(\mathbf{w}_{\text{new}}^2 - \mathbf{w}_{\text{old}}^2)}.\end{aligned}$$

In HMC one samples not from parameter space ($\mathbf{w}$), but an expanded space called "phase space" ($\mathbf{w}, \mathbf{p}$) where a "momentum" p_i is associated with each parameter w_i. This momentum is $p_i = m_i v_i$ where v_i is the velocity (speed, which can be positive or negative, signifying the direction of travel) of parameter w_i through parameter space and m_i is a "mass" associated with that dimension, which can be set by the practitioner to control the relative speeds of the parameters. In this report I will simply set all masses equal to 1. This could be improved, but I leave such modifications to later work.

The total energy of a

Since we are sampling the phase

VII. MORE

One choice would be to do

I found that, for large β (high T) the sampler

using Hamilton's equations of motion (which are equivalent to Newton's Laws of Motion).

I implemented Hamiltonian Monte Carlo **CITE** (HMC), which is a hybrid of

The posterior distribution ((7) with $\beta = 1$) is just such a distribution.

In this section I will show how it can be used to rapily sample the thermal posterior distribution at low temperatures. The essential insight of Replica Exchange is that, at high temperatures (small β) a sampler is essentially exploring the prior distribution which typically has a simple form, so the sampler moves rapidly through the parameter space. In contrast, the posterior ($\beta = 1$) is difficult to explore, since the cost function contains many non-optimal local minima, and transitions between those minima happens slowly. At high temperatures the sampler transitions quickly between these local minima, and it quickly locates "free energy minima" which are broad regions of the cost function containing many reasonably deep local minima. In physics these free energy minima correspond to the stable arrangements of atoms that we see in nature at high temperature.

In replica exchange many independent samplers are propagated at a series of temperatures from low to high, and periodic random swaps of the configurations are made between neightbouring temperatures. Without coupling to higher temperatures the sampler at $\beta = 1$ would become trapped in poor local minima of the cost function. At high temperatures the samplers move quickly around the parameter space, quickly escaping such local minima and finding the broad, deep regions of the cost function. Periodic swaps between neighbouring energy levels are attempted and accepted with probability

$$\begin{aligned}\frac{\text{prob(new)}}{\text{prob(old)}} =& \frac{\text{prob}\left(\mathbf{w}_1|\beta_2, D\right)\text{prob}\left(\mathbf{w}_2|\beta_1, D\right)}{\text{prob}\left(\mathbf{w}_1|\beta_1, D\right)\text{prob}\left(\mathbf{w}_2|\beta_2, D\right)} && (24)\\ =& e^{(\beta_2 - \beta_1)(E(\mathbf{w}_1|D) - E(\mathbf{w}_2|D)))}. && (25)\end{aligned}$$

[1] D. J. MacKay, Neural computation **4**, 448 (1992).