Neural Network Models Implemented in the Flexible Bayesian Modelling Software

Radford M. Neal, 2022

This document gives the mathematical details of the Bayesian neural network models implemented in the FBM software. The notation used is summarized on the next page.

Network architectures

A network consists of a layer of input units, zero or more hidden layers of units with some activation function, and a layer of output units. Each hidden layer is connected to the preceeding hidden layer and to the input layer. The output layer is connected each of the hidden layers and to the input layer. Each unit in the hidden and output layers may have a bias, added to its input from other layers. Each unit in the input and hidden layers may have an offset, added to its output. Any of these sets of parameters may be missing in any particular network, which has the same effect as their being zero.

The software now also supports connecting a hidden layer to an earlier hidden layer other than the immediately preceeding one. Such non-sequential connections are handled in the obvious way, with the required extensions mostly not noted in this presentation.

The interpretation of the output units is determined by the data model, described below.

When all pairs of connected layers are fully connected, the following formulas define the outputs, v_i^O, of a network for given values of the inputs, v_i^I:

$$u_j^\ell \;=\; b_j^\ell \;+\; \sum_i w_{i,j}^{I,\ell}(v_i^I + t_i^I) \;+\; \sum_i w_{i,j}^{\ell-1,\ell}(v_i^{\ell-1} + t_i^{\ell-1}) \tag{1}$$

$$v_i^\ell \;=\; a^\ell(u_i^\ell) \tag{2}$$

$$v_j^O \;=\; b_i^O \;+\; \sum_i w_{i,j}^{I,O}(v_i^I + t_i^I) \;+\; \sum_\ell \sum_i w_{i,j}^{\ell,O}(v_i^\ell + t_i^\ell) \tag{3}$$

Here, a^ℓ is the activation function used for layer ℓ. The summations are over all units in the appropriate layer. (The number of units in each layer is part of the architecture specification, but these numbers are not given symbols here.) The term in the equation for u_i^ℓ involving layer $\ell-1$ is omitted for the first hidden layer; terms involving other pairs of layers will also be omitted when those connections are absent from the architecture.

The following activation functions of this form are supported:

tanh: $a(u) = \tanh(u)$

softplus: $a(u) = \log(1+\exp(u))$

softplus0: $a(u) = \log(1+\exp(u)) - \log(2)$

identity: $a(u) = u$

Alternatively, a layer, ℓ, may compute its values, v_i^ℓ from its summed inputs, u_i^ℓ, in a manner that does not decompose into computation of v_i^ℓ from u_i^ℓ separately for each i, as is the case for **softmax** and **normalization** layers. The softmax computation is

$$v_i^\ell = \exp(u_i^\ell) \,/\, \sum_j \exp(u_j^\ell) \tag{4}$$

Computation for normalization layers is defined by

$$s^\ell \;=\; \sqrt{0.01 \;+\; \textstyle\sum_j [u_j^\ell]^2 \,/\, n^\ell} \tag{5}$$

$$v_i^\ell \;=\; u_i^\ell \,/\, s \tag{6}$$

where n^ℓ is the number of units in layer ℓ. Normalization may instead be done separately for sub-groups of units.

Values associated with units

v_i^I	Value of ith input unit, before the offset is added
v_i^ℓ	Value of ith hidden unit in layer ℓ, before the offset is added
v_i^O	Value of the ith output unit
u_i^ℓ	Value of the input to the ith unit of hidden layer ℓ
a^ℓ	Activation function for hidden layer ℓ, used to compute $v_i^\ell = a^\ell(u_i^\ell)$

Parameters of the network

t_i^I	Offset for ith input unit
t_i^ℓ	Offset for ith hidden unit in layer ℓ
b_j^ℓ	Bias for jth unit in hidden layer ℓ
b_j^O	Bias for jth output unit
$w_{i,j}^{I,O}$	Weight from ith input unit to jth output unit
$w_{i,j}^{I,\ell}$	Weight from ith input unit to jth unit in hidden layer ℓ
$w_{i,j}^{\ell-1,\ell}$	Weight from ith unit in hidden layer $\ell-1$ to jth unit in hidden layer ℓ
$w_{i,j}^{\ell,O}$	Weight from ith unit in hidden layer ℓ to the jth output unit
b_k^ℓ	Bias k in configured biases for hidden layer ℓ
b_k^O	Bias k in configured biases for the output layer
$w_k^{I,\ell}$	Weight k in configured connections from the input layer to hidden layer ℓ
$w_k^{I,O}$	Weight k in configured connections from the input layer to the output layer ℓ
$w_k^{\ell-1,\ell}$	Weight k in configured connections from hidden layer $\ell-1$ to hidden layer ℓ
$w_k^{\ell,O}$	Weight k in configured connections from last hidden layer, ℓ, to the output layer
B^ℓ	Set of doublets, (j,k), giving configuration of biases for hidden layer ℓ
B^O	Set of doublets, (j,k), giving configuration of biases for the output layer
$C^{I,\ell}$	Set of triplets, (i,j,k), giving configuration of connections from the input layer to hidden layer ℓ
$C^{I,O}$	Set of triplets, (i,j,k), giving configuration of connections from the input layer to the output layer
$C^{\ell-1,\ell}$	Set of triplets, (i,j,k), giving configuration of connections from hidden layer $\ell-1$ to hidden layer ℓ
$C^{\ell,O}$	Set of triplets, (i,j,k), giving configuration of connections from the last hidden layer, ℓ, to the output layer

Hyperparameters defining priors for parameters

σ_t^I	Common sigma for offsets of input units
σ_t^ℓ	Common sigma for offsets of units in hidden layer ℓ
σ_b^ℓ	Common sigma for biases of units in hidden layer ℓ
σ_b^O	Common sigma for biases of output units
$\sigma_w^{I,O}$	Common sigma for weights from input units to output units
$\sigma_w^{I,\ell}$	Common sigma for weights from input units to units in hidden layer ℓ
$\sigma_w^{\ell-1,\ell}$	Common sigma for weights from units in hidden layer $\ell-1$ to units in hidden layer ℓ
$\sigma_w^{\ell,O}$	Common sigma for weights from units in hidden layer ℓ to output units
$\sigma_{w,i}^{I,O}$	Sigma for weights from input unit i to output units
$\sigma_{w,i}^{I,\ell}$	Sigma for weights from input unit i to units in hidden layer ℓ
$\sigma_{w,i}^{\ell-1,\ell}$	Sigma for weights from unit i in hidden layer $\ell-1$ to units in hidden layer ℓ
$\sigma_{w,i}^{\ell,O}$	Sigma for weights from unit i in hidden layer ℓ to output units
$\sigma_{a,j}^O$	Sigma adjustment for weights and biases into output unit j
$\sigma_{a,j}^\ell$	Sigma adjustment for weights and biases into unit j in hidden layer ℓ

Connections from an input layer to a hidden or output layer may not be fully connected, but can instead be configured by specifying a set, $C^{I,\ell}$ of triplets, (i, j, k), indicating that unit i in the input layer contributes to the input of unit j of the hidden layer, using the weight indexed by k, or similarly by a set $C^{I,O}$ for the output layer. The set of weights referenced may be smaller (or larger) than the number in that would be present when each input unit connects to each hidden or output unit, with an unshared weight. The configuration may specify that a weight is shared by more than one i,j pair, and a single i,j pair may be connected with more than one weight.

Similarly, connections from a hidden layer to the next hidden layer may be specified by a set of triplets $C^{\ell-1,\ell}$, and connections from the last hidden layer to the output layer may be specified by a set $C^{\ell,O}$, with ℓ the index of the last hidden layer.

The bias connections for a hidden or output layer may also be configured, by specifying a set, B^ℓ or B^O, of doublets, (j, k), indicating that the bias indexed by k is used for unit j. A bias may be shared among several units. Some units may have no bias, or more than one bias.

When the biases and connections to a hidden layer from inputs and the previous hidden layer are configured in this way, the input to a hidden unit is computed as follows:

$$u_j^\ell \;=\; \sum_{k\,:\,(j,k)\in B^\ell} b_k^\ell \;+\; \sum_{(i,k)\,:\,(i,j,k)\in C^{I,\ell}} w_k^{I,\ell}(v_i^I + t_i^I) \;+\; \sum_{(i,k)\,:\,(i,j,k)\in C^{\ell-1,\ell}} w_k^{\ell-1,\ell}(v_i^{\ell-1} + t_i^{\ell-1}) \tag{7}$$

When some of the connections or biases into a hidden layer are configured in this way, but others are fully-connected, the appropriate combination of equations (1) and (7) is used.

Data models

Networks are normally used to define models for the conditional distribution of a set of "target" values given a set of "input" values. There are three sorts of models, corresponding to three sorts of targets — real-valued targets (a "regression" model), binary-valued targets (a "logistic regression" model), and "class" targets taking on values from a (small) finite set (generalized logistic regression, or "softmax") — plus survival models, which are not described here. For regression and logistic regression models, the number of target values is equal to the number of network outputs. For the softmax model, there is only one target, with the number of possible values for this target being equal to the number of network outputs.

The distributions for real-valued targets, y_j, in a case with inputs v_i^I may be modeled by independent Gaussian distributions with means given by the corresponding network outputs, and with standard deviations given by the hyperparameters σ_j. The probability density for a target given the associated inputs and the network parameters is then

$$P(y_j \mid \text{inputs, parameters}) \;=\; \frac{1}{\sqrt{2\pi}\sigma_j} \exp\big(-(y_j - v_j^O)^2/2\sigma_j^2\big) \tag{8}$$

Alternatively, each case, c, may have its own set of standard deviations, $\sigma_{j,c}$, with the corresponding precisions, $\tau_{j,c} = \sigma_{j,c}^{-2}$ being given Gamma distributions with means of τ_j and shape parameter α_2 (called this for reasons that will be clear later):

$$P(\tau_{j,c} \mid \tau_j) \;=\; \frac{(\alpha_2/2\tau_j)^{\alpha_2/2}}{\Gamma(\alpha_2/2)}\, \tau_{j,c}^{\alpha_2/2-1} \exp\big(-\tau_{j,c}\alpha_2/2\tau_j\big) \tag{9}$$

The previous case corresponds to the degenerate Gamma distribution with $\alpha_2 = \infty$. Otherwise, integrating out $\tau_{j,c}$ gives a t-distribution for the target with α_2 "degrees of freedom":

$$P(y_j \mid \text{inputs, parameters}) \;=\; \frac{\Gamma((\alpha_2+1)/2)}{\Gamma(\alpha_2/2)\sqrt{\pi\alpha_2}\sigma_j} \big[1 + (y_j - v_j^O)^2/\alpha_2\sigma_j^2\big]^{-(\alpha_2+1)/2} \tag{10}$$

The probability that a binary-valued target has the value 1 is given by

$$P(y_j = 1 \mid \text{inputs, parameters}) \;=\; \big[1 + \exp(-v_j^O)\big]^{-1} \tag{11}$$

The probability that a class target, y, has the value j is given by

$$P(y = j \mid \text{inputs, parameters}) \quad = \quad \exp(v_j^O) \,/\, \sum_k \exp(v_k^O) \tag{12}$$

Prior distributions for parameters and hyperparameters

The prior distributions for the parameters of a network are defined in terms of hyperparameters. Conceptually, there is a one hyperparameter for every parameter, but these lowest-level hyperparameters are not explicitly represented. Mid-level hyperparameters control the distribution of a group of low-level hyperparameters that are all of one type and all associated with the same source unit. High-level (or "common") hyperparameters control the distribution of the mid-level hyperparameters, or of the low-level hyperparameters for parameter types with no mid-level hyperparameters. The same three-level scheme is used for the noise level in regression models.

These hyperparameters are represented in terms of "sigma" values, σ, but their distributions are specified in terms of the corresponding "precisions", $\tau = \sigma^{-2}$, which are given Gamma distributions. The top-level mean is given by a "width" parameter associated with the parameter type. The shape parameters of the Gamma distributions are determined by "alpha" values associated with each type of parameter. An alpha value of infinity concentrates the entire distribution on the mean, effectively removing one level from the hierarchy.

The sigma for a weight may also be multiplied by an "adjustment" that is associated with the destination unit. This gives the following generic scheme:

$$P(\tau_w) \quad = \quad \frac{(\alpha_{w,0}/2\omega_w)^{\alpha_{w,0}/2}}{\Gamma(\alpha_{w,0}/2)}\, \tau_w^{\alpha_{w,0}/2-1} \exp\big(-\tau_w \alpha_{w,0}/2\omega_w\big) \tag{13}$$

$$P(\tau_{w,i} \mid \tau_w) \quad = \quad \frac{(\alpha_{w,1}/2\tau_w)^{\alpha_{w,1}/2}}{\Gamma(\alpha_{w,1}/2)}\, \tau_{w,i}^{\alpha_{w,1}/2-1} \exp\big(-\tau_{w,i} \alpha_{w,1}/2\tau_w\big) \tag{14}$$

$$P(\tau_{a,j}) \quad = \quad \frac{(\alpha_a/2)^{\alpha_a/2}}{\Gamma(\alpha_a/2)}\, \tau_{a,j}^{\alpha_a/2-1} \exp\big(-\tau_{a,j} \alpha_a/2\big) \tag{15}$$

For weights from input units to output units, for example, τ_w will equal $\tau_w^{I,O} = [\sigma_w^{I,O}]^{-2}$, and similarly for $\tau_{w,i}$, while $\tau_{a,j}$ will equal $[\sigma_{a,j}^O]^{-2}$. The top-level precision value, ω_w, is derived from the "width" parameter specified for this type of weight. The positive (possibly infinite) values $\alpha_{w,0}$ and $\alpha_{w,1}$ are also part of the prior specification for input to output weights, while α_a is a specification associated with the output units (note that in this case the "width" parameter is fixed at one, as freedom to set it would be redundant).

Weights for connections between other pairs of layers are given priors in similar fashion, except that when a pair of layers has configured connections, rather than being fully connected, the prior specification for the weights must have infinite $\alpha_{w,1}$, and the adjustments for the destination layer are ignored (assumed to be one) — since neither of these makes sense when the weights are not associated with a particular source and destination unit.

The distribution for a weight from unit i of one layer to unit j of another layer may be Gaussian with mean zero and standard deviation given by $\sigma_{w,i}\sigma_{a,j} = [\tau_{w,i}\tau_{a,j}]^{-1/2}$. That is:

$$P(w_{i,j} \mid \sigma_{w,i},\, \sigma_{a,j}) \quad = \quad \frac{1}{\sqrt{2\pi}\sigma_{w,i}\sigma_{a,j}} \exp\big(-w_{i,j}^2/2\sigma_{w,i}^2\sigma_{a,j}^2\big) \tag{16}$$

Here, $w_{i,j}$ represents, for example, $w_{i,j}^{I,O}$, in which case $\sigma_{w,i}$ represents $\sigma_{w,i}^{I,O}$ and $\sigma_{a,j}$ represents $\sigma_{a,j}^O$. When the weights are for configured connections, $w_{i,j}$ is replaced by w_k, $\sigma_{w,i}$ is replaced by σ_w, and $\sigma_{a,j}$ is replaced by 1.

Alternatively, each individual weight may have its own "sigma", with the corresponding precision having a Gamma distribution with mean $\tau_{w,i}\tau_{a,j}$ and shape parameter given by $\alpha_{w,2}$. The previous case corresponds to the degenerate distribution with $\alpha_{w,2} = \infty$. Otherwise, we can integrate over the individual precisions and obtain t-distributions for each weight:

$$P(w_{i,j} \mid \sigma_{w,i},\, \sigma_{a,j}) \quad = \quad \frac{\Gamma((\alpha_{w,2}+1)/2)}{\Gamma(\alpha_{w,2}/2)\sqrt{\pi\alpha_{w,2}}\,\sigma_{w,i}\sigma_{a,j}} \big[1 + w_{i,j}^2/\alpha_{w,2}\sigma_{w,i}^2\sigma_{a,j}^2\big]^{-(\alpha_{w,2}+1)/2} \tag{17}$$

When the weights are for a configured set of connections, $w_{i,j}$ is replaced by w_k, and so forth, as above.

The same scheme is used for biases, except that there are in these cases no mid-level hyperparameters. We have

$$P(\tau_b) \;=\; \frac{(\alpha_{b,0}/2\omega_b)^{\alpha_{b,0}/2}}{\Gamma(\alpha_{b,0}/2)}\, \tau_b^{\alpha_{b,0}/2-1} \exp\big(-\tau_b\alpha_{b,0}/2\omega_b\big) \tag{18}$$

where τ_b might, for example, be $\tau_b^O = [\sigma_b^O]^{-2}$, etc.

The distribution of the biases is then either

$$P(b_i \mid \sigma_b,\, \sigma_{a,i}) \;=\; \frac{1}{\sqrt{2\pi}\sigma_b\sigma_{a,i}} \exp\big(-b_i^2/2\sigma_b^2\sigma_{a,i}^2\big) \tag{19}$$

if $\alpha_{b,1} = \infty$, or if not

$$P(b_i \mid \sigma_b,\, \sigma_{a,i}) \;=\; \frac{\Gamma((\alpha_{b,1}+1)/2)}{\Gamma(\alpha_{b,1}/2)\sqrt{\pi\alpha_{b,1}}\,\sigma_b\sigma_{a,i}} \big[1 + b_i^2/\alpha_{b,1}\sigma_b^2\sigma_{a,i}^2\big]^{-(\alpha_{b,1}+1)/2} \tag{20}$$

For the offsets added to hidden unit values, there are no mid-level hyperparameters, and neither are "adjustments" used. We have

$$P(\tau_t) \;=\; \frac{(\alpha_{t,0}/2\omega_t)^{\alpha_{t,0}/2}}{\Gamma(\alpha_{t,0}/2)}\, \tau_t^{\alpha_{t,0}/2-1} \exp\big(-\tau_t\alpha_{t,0}/2\omega_t\big) \tag{21}$$

where τ_t might, for example, be $\tau_t^I = [\sigma_t^I]^{-2}$, etc.

The distribution of the offsets is then either

$$P(y_i \mid \sigma_t) \;=\; \frac{1}{\sqrt{2\pi}\sigma_t} \exp\big(-t_i^2/2\sigma_t^2\big) \tag{22}$$

if $\alpha_{t,1} = \infty$, or if not

$$P(b_i \mid \sigma_t) \;=\; \frac{\Gamma((\alpha_{t,1}+1)/2)}{\Gamma(\alpha_{t,1}/2)\sqrt{\pi\alpha_{t,1}}\,\sigma_t} \big[1 + t_i^2/\alpha_{t,1}\sigma_t^2\big]^{-(\alpha_{t,1}+1)/2} \tag{23}$$

The scheme for noise levels in regression models is also similar, with τ_j, the precision for target j, being specified in terms of an overall precision, τ, as follows:

$$P(\tau) \;=\; \frac{(\alpha_0/2\omega)^{\alpha_0/2}}{\Gamma(\alpha_0/2)}\, \tau^{\alpha_0/2-1} \exp\big(-\tau\alpha_0/2\omega\big) \tag{24}$$

$$P(\tau_j \mid \tau) \;=\; \frac{(\alpha_1/2\tau)^{\alpha_1/2}}{\Gamma(\alpha_1/2)}\, \tau_j^{\alpha_1/2-1} \exp\big(-\tau_j\alpha_1/2\tau\big) \tag{25}$$

where ω, α_0, and α_1 are parts of the noise specification. A third alpha (α_2) is needed for the final specification of the noise in individual cases, as described in the Section .

Scaling of priors

The top-level precisions used in the preceding hierarchical priors (the ω values) may simply be taken as specified (actually, what is specified is the corresponding "width", $\omega^{-1/2}$). Alternatively, for connection weights only (not biases and offsets), the ω for parameters of one type may be scaled automatically, based on the number of source units that feed into each destination unit via connections of this type. This scaling is designed to produced sensible results as the number of source units goes to infinity, while all other specifications remain unchanged. Scaling is not allowed for configured connections, since the proper scaling would depend on the (unknown) way in which the configuration would change with the number of souce units.

The theory behind this scaling concerns the convergence of sums of independent random variables to "stable laws". The symmetric stable laws are characterized by a width parameter and an index, α, in the range (0,2]. If $X_1, \ldots, X_n$ are distributed according to a symmetric stable law of index α, then $(X_1 + \cdots + X_n)/n^{1/\alpha}$ is distributed according to the same stable law. The stable law with index 2 is the Gaussian. The sums of all random variables with finite variance converge to the Gaussian, along with some others. Typically, random variables whose moments are defined up to but not including α converge to the stable law with index α, for $\alpha < 2$.

This leads to the following scaling rules for producing ω based on the specified base precision, ω_0, the number of source units, n, and the relevant α value (see below):

$$\omega \quad = \quad \begin{cases} \omega_0 n & \text{for } \alpha = \infty \\ \omega_0 n\alpha/(\alpha-2) & \text{for } \alpha > 2 \\ \omega_0 n \log n & \text{for } \alpha = 2 \quad \text{(but fudged to } \omega_0 n \text{ if } n < 3) \\ \omega_0 n^{2/\alpha} & \text{for } \alpha < 2 \end{cases} \tag{26}$$

Here, α is $\alpha_{w,2}$ if that is finite, and is otherwise $\alpha_{w,1}$. The scheme doesn't really work if both $\alpha_{w,1}$ and $\alpha_{w,2}$ are finite. When $\alpha = 2$, the scaling produces convergence to the Gaussian distribution, but with an unusual scale factor, as the t-distribution with $\alpha = 2$ is in the "non-normal" domain of attraction of the Gaussian distribution.

Conditional distributions for hyperparameters

Implementation of Gibbs sampling for hyperparameters requires sampling from the conditional distribution for one hyperparameter given the values of the other hyperparameters and of the network parameters.

Lowest-level conditional distributions

The simplest conditional distributions to sample from are those for "sigma" hyperparameters that directly control a set of network parameters. This will be the situation for the lowest level sigmas, as well as for higher level sigmas when the lower level sigmas are tied exactly to this higher level sigma (i.e. when the "alpha" shape parameter for their distribution is infinite). The situation is analogous for sigma values relating to noise in regression models, except that the errors in training case are what is modeled, rather than the network parameters.

In general, we will have some hyperparameter $\tau = \sigma^{-2}$ that has a Gamma prior, with shape parameter we will call α, and with mean ω (which may be a higher-level hyperparameter). The purpose of τ is to specify the precisions for the independent Gaussian distributions of n lower-level quantities, z_i. In this situation, the conditional distribution for τ will be given by the following proportionality:

$$P(\tau \mid \{z_i\}, \ldots) \quad \propto \quad \tau^{\alpha/2-1} \exp(-\tau\alpha/2\omega) \cdot \prod_i \tau^{1/2} \exp(-\tau z_i^2/2) \tag{27}$$

$$\propto \quad \tau^{(\alpha+n)/2-1} \exp\left(-\tau(\alpha/\omega + \sum_i z_i^2)/2\right) \tag{28}$$

The first factor in equation (27) derives from the prior for τ, the remaining factors from the effect of τ on the probabilities of the z_i. The result is a Gamma distribution that can be sampled from by standard methods (Devroye 1986).

When the distributions of the z_i are influenced by "adjustments", $\tau_{a,i}$, the above formula is modified as follows:

$$P(\tau \mid \{z_i\}, \{\tau_{a,i}\}, \ldots) \quad \propto \quad \tau^{(\alpha+n)/2-1} \exp\left(-\tau(\alpha/\omega + \sum_i \tau_{a,i} z_i^2)/2\right) \tag{29}$$

Gibbs sampling for the adjustments themselves is done in similar fashion, using the weighted sum of squares of parameters influenced by the adjustment, with the weights in this case being the precisions associated with each parameter.

Higher-level conditional distributions

Sampling from the conditional distribution for a sigma hyperparameter that controls a set of lower-level sigmas is more difficult, but can be done in the most interesting cases using rejection sampling.

Assume that we wish to sample from the distribution for a precision hyperparameter τ, which has a higher level Gamma prior specified by α_0 and ω, and which controls the distributions of lower-level hyperparameters, τ_i, that have independent Gamma distributions with shape parameter α_1 and mean τ. The conditional distribution for τ is then given by the following proportionality:

$$P(\tau \mid \{\tau_i\}, \ldots) \propto \tau^{\alpha_0/2-1} \exp(-\tau\alpha_0/2\omega) \cdot \prod_i \tau^{-\alpha_1/2} \exp(-\tau_i\alpha_1/2\tau) \tag{30}$$

$$\propto \tau^{(\alpha_0-n\alpha_1)/2-1} \exp\Big(-\tau\alpha_0/2\omega - (\alpha_1 \sum_i \tau_i) \,/\, 2\tau\Big) \tag{31}$$

Defining $\gamma = 1/\tau$, we get:

$$P(\gamma \mid \{\tau_i\}, \ldots) \propto \tau^2 \, P(\tau \mid \{\tau_i\}, \ldots) \tag{32}$$

$$\propto \tau^{(\alpha_0-n\alpha_1)/2+1} \exp\Big(-\tau\alpha_0/2\omega - (\alpha_1 \sum_i \tau_i) \,/\, 2\tau\Big) \tag{33}$$

$$\propto \gamma^{(n\alpha_1-\alpha_0)/2-1} \exp\Big(-\gamma(\alpha_1 \sum_i \tau_i) \,/\, 2\Big) \cdot \exp\big(-\alpha_0/2\omega\gamma\big) \tag{34}$$

The first part of this has the form of a Gamma distribution for γ, provided $n\alpha_1 > \alpha_0$; the last factor lies between zero and one. If $n\alpha_1 > \alpha_0$, we can therefore obtain a value from the distribution for γ by repeatedly sampling from the Gamma distribution with shape parameter $n\alpha_1 - \alpha_0$ and mean $(n\alpha_1 - \alpha_0)/(\alpha_1 \sum_i \tau_i)$ until the value of γ generated passes an acceptance test, which it does with probability $\exp(-\alpha_0/2\omega\gamma)$. The probability of rejection will be reasonably low if α_0 is small, which is typical. It should be possible to develop better methods if necessary.

In some contexts, the values τ_i are not explicitly represented, and must themselves be found by sampling using the method of the previous section.

Calculation of derivatives

The hybrid Monte Carlo and other dynamical methods require calculation of the derivatives of the log of the likelihood of the parameter values given the data, and of the log of the prior probability of the parameter values. This section details how this is done.

Derivatives of the log prior probability

For fixed values of the explicitly-represented hyperparameters, one can easily obtain the derivatives of the log of the prior probability with respect to the network weights and other parameters. Generically, if $\alpha_{w,2} = \infty$, we get, from equation (16), that

$$\frac{\partial}{\partial w_{i,j}} \log P(w_{i,j} \mid \sigma_{w,i}, \sigma_{a,j}) = -\frac{w_{i,j}}{\sigma_{w,i}^2 \sigma_{a,j}^2} \tag{35}$$

while otherwise, we get, from equation (17), that

$$\frac{\partial}{\partial w_{i,j}} \log P(w_{i,j} \mid \sigma_{w,i}, \sigma_{a,j}) = -\frac{\alpha_{w,2}+1}{\alpha_{w,2}\sigma_{w,i}^2\sigma_{a,j}^2} \frac{w_{i,j}}{\big[1 + w_{i,j}^2/\alpha_{w,2}\sigma_{w,i}^2\sigma_{a,j}^2\big]} \tag{36}$$

This is modified in the obvious way when the weights are for configured connections (indexed by k rather than i, j). Similar formulas for derivatives with respect to the biases are obtained from equations (19) and (20) and for derivatives with respect to the offsets from equations (22) and (23).

Derivatives of the log likelihood with respect to output unit values

The starting point for calculating the derivatives of the log likelihood with respect to the network parameters is to calculate the derivative of the log likelihood due to a particular case with respect to the network outputs. For the regression model with $\alpha_2 = \infty$, we get from equation (8) that

$$\frac{\partial}{\partial v_j^O} \log P(y \mid v_j^O) \;=\; -\frac{y_j - v_j^O}{\sigma_j^2} \tag{37}$$

When α_2 is finite, we get from equation (10) that

$$\frac{\partial}{\partial v_j^O} \log P(y \mid v_j^O) \;=\; -\frac{\alpha_2 + 1}{\alpha_2 \sigma_j^2} \, \frac{y_j - v_j^O}{\left[1 + (y_j - v_j^O)^2 / \alpha_2 \sigma_j^2\right]} \tag{38}$$

For the model of binary targets given by equation (11), we get the following, after some manipulation:

$$\frac{\partial}{\partial v_j^O} \log P(y \mid v_j^O) \;=\; y_j \,-\, \left[1 + \exp(-v_j^O)\right]^{-1} \;=\; y_j \,-\, P(y_j = 1 \mid v_j^O) \tag{39}$$

For the many-way "softmax" classification model of equation (12), we get

$$\frac{\partial}{\partial v_j^O} \log P(y \mid \{v_k^O\}) \;=\; \delta(y, j) \,-\, \frac{\exp(v_j^O)}{\sum_k \exp(v_k^O)} \;=\; \delta(y, j) \,-\, P(y = j \mid \{v_k^O\}) \tag{40}$$

Here, $\delta(y, j)$ is one if $y = j$ and zero otherwise.

From now on, the log likelihood due to one case, $\log P(y \mid \text{inputs, parameters}) = \log P(y \mid \text{outputs})$, will be denoted by L.

Backpropagation to earlier unit values

Once the derivatives of the log likelihood with respect to the output unit values are known, the standard back-propagation technique can be used to find the derivatives with respect to the values of the hidden and input units. Using equations (1), (2), and (3), we get the following, which can be evaluated going backwards from the last hidden layer to the first hidden layer, and then the inputs (if required):

$$\frac{\partial L}{\partial v_i^\ell} \;=\; \sum_j w_{i,j}^{\ell,O} \frac{\partial L}{\partial v_j^O} \;+\; \sum_j w_{i,j}^{\ell,\ell+1} \frac{\partial L}{\partial u_j^{\ell+1}} \tag{41}$$

$$\frac{\partial L}{\partial u_i^\ell} \;=\; a'^{\ell}\big(u_i^\ell\big) \frac{\partial L}{\partial v_i^\ell} \tag{42}$$

$$\frac{\partial L}{\partial v_i^I} \;=\; \sum_j w_{i,j}^{I,O} \frac{\partial L}{\partial v_j^O} \;+\; \sum_\ell \sum_j w_{i,j}^{I,\ell} \frac{\partial L}{\partial u_j^\ell} \tag{43}$$

The second term in the first equation above is omitted for the last hidden layer; additional terms of this sort will be present when there are non-sequential connections between hidden layers.

When the connections from a hidden layer to the next hidden layer are configured, the first equation above is modified as follows:

$$\frac{\partial L}{\partial v_i^\ell} \;=\; \sum_j w_{i,j}^{\ell,O} \frac{\partial L}{\partial v_j^O} \;+\; \sum_{(j,k)\,:\,(i,j,k) \in C^{\ell,\ell+1}} w_{i,j}^{\ell,\ell+1} \frac{\partial L}{\partial u_j^{\ell+1}} \tag{44}$$

Similarly, when all connections from the inputs to hidden layers are configured, the last equation above is modified as follows:

$$\frac{\partial L}{\partial v_i^I} = \sum_j w_{i,j}^{I,O} \frac{\partial L}{\partial v_j^O} + \sum_\ell \sum_{(j,k)\,:\,(i,j,k)\in C^{I,\ell}} w_k^{I,\ell} \frac{\partial L}{\partial u_j^\ell} \tag{45}$$

The obvious modifications to the above are made when some of the hidden layers are fully-connected to the inputs and some have configured connections to the inputs.

In equation (42), $a'^\ell(u_i^\ell)$ is the derivative of the activation function, evaluated at u_i^ℓ. For computational reasons, it is convenient to express this derivative in terms of v_i^ℓ, which is possible since all the activation functions used are strictly monotonic. For the tanh activation function, the derivative of a is $1 - \left[v_i^\ell\right]^2$. For the softplus activation function, it is $1 - \exp(-v_i^\ell)$, and for softplus0, it is $1 - \exp(-v_i^\ell - \log(2))$.

For a softmax layer, equation (42) is replaced by

$$t^\ell = \sum_j \frac{\partial L}{\partial v_j^\ell} v_j^\ell \tag{46}$$

$$\frac{\partial L}{\partial u_i^\ell} = v_i^\ell \Big(\frac{\partial L}{\partial v_j^\ell} - t^\ell \Big) \tag{47}$$

where s^ℓ is defined by equation (5). This is modified in the obvious way when normalization is done separately for sub-groups.

For a normalization layer, equation (42) is replaced by

$$t^\ell = \sum_j \frac{\partial L}{\partial v_j^\ell} v_j^\ell \,/\, n^\ell \tag{48}$$

$$\frac{\partial L}{\partial u_i^\ell} = \frac{1}{s^\ell} \Big(\frac{\partial L}{\partial v_j^\ell} - v_k^\ell t^\ell \Big) \tag{49}$$

where s^ℓ is defined by equation (5). This is modified in the obvious way when normalization is done separately for sub-groups.

Derivatives of the log likelihood with respect to parameters

The derivatives of L with respect to the network parameters (with any explicitly represented noise sigmas fixed) are obtained using the derivatives with respect to unit values and unit inputs that were found in the previous section, as follows:

$$\frac{\partial L}{\partial b_i^O} = \frac{\partial L}{\partial v_i^O} \tag{50}$$

$$\frac{\partial L}{\partial b_i^\ell} = \frac{\partial L}{\partial u_i^\ell} \tag{51}$$

$$\frac{\partial L}{\partial t_i^\ell} = \frac{\partial L}{\partial v_i^\ell} \tag{52}$$

$$\frac{\partial L}{\partial t_i^I} = \frac{\partial L}{\partial v_i^I} \tag{53}$$

$$\frac{\partial L}{\partial w_{i,j}^{\ell,O}} = \frac{\partial L}{\partial v_j^O} \left(v_i^\ell + t_i^\ell\right) \tag{54}$$

$$\frac{\partial L}{\partial w_{i,j}^{\ell-1,\ell}} = \frac{\partial L}{\partial u_j^\ell}\,(v_i^{\ell-1} + t_i^{\ell-1}) \tag{55}$$

$$\frac{\partial L}{\partial w_{i,j}^{I,\ell}} = \frac{\partial L}{\partial u_j^\ell}\,(v_i^I + t_i^I) \tag{56}$$

$$\frac{\partial L}{\partial w_{i,j}^{I,O}} = \frac{\partial L}{\partial v_j^O}\,(v_i^I + t_i^I) \tag{57}$$

The derivatives found will then be summed across training cases, and added to the derivatives with respect to the log prior probability, to give the derivatives with respect to the log posterior probability, which controls the dynamics for HMC or gradient descent.

Heuristic choice of stepsizes

Stepsizes for Metropolis updates and dynamical trajectory computations are heuristically chosen based on the values of the training inputs and the current values of the hyperparameters. These stepsize choices are made on the assumption that the system is near equilibrium, moving about in a Gaussian hump. If the axes of this hump were aligned with the coordinate axes, the optimal stepsize along each axis would be in the vicinity of the standard deviation along that axis. Since the axes of the bowl may not be aligned with the coordinate axes, the actual stepsizes may have to be less than this. On the other hand, the estimates used are in some respects conservative. Adjustments to account for these factors are left to the user.

Estimates of the posterior standard deviations along the axes are based on estimates of the second derivatives of the log posterior probability along the axes. These second derivatives are estimated using estimates of the second derivatives of the log likelihood with respect to the values of units in the network.

For real-valued targets, with $\alpha_2 = \infty$, we get the following, using equation (37):

$$-\frac{\partial^2 L}{\partial (v_j^O)^2} = \frac{1}{\sigma_j^2} \tag{58}$$

while for finite α_2, we get from equation (38) that

$$-\frac{\partial^2 L}{\partial (v_j^O)^2} = \frac{\alpha_2 + 1}{\alpha_2 \sigma_j^2}\left[\left(1 + \frac{(v_j^O)^2}{\alpha_2 \sigma_j^2}\right)^{-1} + \frac{2(v_j^O)^2}{\alpha_2 \sigma_j^2}\left(1 + \frac{(v_j^O)^2}{\alpha_2 \sigma_j^2}\right)^{-2}\right] \tag{59}$$

This is estimated by its maximum value, which occurs at $v_j^O = 0$:

$$-\frac{\partial^2 L}{\partial (v_j^O)^2} \approx \frac{\alpha_2 + 1}{\alpha_2 \sigma_j^2} \tag{60}$$

For binary-valued targets, equation (39) gives

$$-\frac{\partial^2 L}{\partial (v_j^O)^2} = \frac{1}{[1 + \exp(v_j^O)][1 + \exp(-v_j^O)]} \approx \frac{1}{4} \tag{61}$$

Again, the estimate is based on the maximum possible value, which occurs when $v_j^O = 0$.

We get a similar estimate for a class target, using equation (40):

$$-\frac{\partial^2 L}{\partial (v_j^O)^2} = \frac{\exp(v_j^O)}{\sum_k \exp(v_k^O)}\left[1 - \frac{\exp(v_j^O)}{\sum_k \exp(v_k^O)}\right] \approx \frac{1}{4} \tag{62}$$

These estimates for the second derivatives of L with respect to the outputs are propagated backward to give estimates for the second derivatives of L with respect to the values of hidden and input units.

When doing this backward propagation, we need an estimate of the second derivative of L with respect to the summed input to a hidden unit, given its second derivative with respect to the unit's output. Letting the hidden unit output be $v = a(u)$, we have

$$\frac{d^2L}{du^2} \;=\; \frac{d}{du}\left[\frac{dv}{du}\,\frac{dL}{dv}\right] \;=\; a''(u)\frac{dL}{dv} + a'(u)^2\frac{d^2L}{dv^2} \;\approx\; a'(u)^2\frac{d^2L}{dv^2} \;\approx\; \max[a'(u)^2]\,\frac{d^2L}{dv^2} \tag{63}$$

The first approximation assumes that since the first term may be either positive or negative, its effects will (optimistically) cancel when averaged over the training set. Since u is not known, the second approximation above takes the maximum with respect to u. For the currently-used unit-by-unit activation functions — tanh, softplus, softplus0, and identity — $\max[a'(u)^2] = 1$, so nothing need actually be done for this. For normalization layers, the partial derivative of a unit's output with respect to its input is no more than $1/s^\ell$, where s^ℓ is defined by equation (5). An estimate for $[1/s^\ell]^2 = 1/(0.01 + \sum_i [u_i^\ell]^2 \,/\, n^\ell)$ is therefore needed. This requires an estimate for a typical value of $[u_i^\ell]^2$, which is discussed below.

Note that the backward propagation ignores any interactions between multiple connections from a unit.

Since the stepsizes chosen are not allowed to depend on the actual values of the network parameters, the magnitude of each weight is taken to be equal to the corresponding sigma hyperparameter, multiplied by the destination unit adjustment, if present. This gives the following generic estimate, when layers are fully connected:

$$\frac{\partial^2 L}{\partial (v_i^S)^2} \;\approx\; \sum_D \sum_j (\sigma_{w,i}^{S,D}\,\sigma_{a,j}^D)^2\,\frac{\partial^2 L}{(u_j^D)^2} \tag{64}$$

Here, S is the source layer, D goes through the various layers receiving connections from S, $\sigma_{w,i}^{S,D}$ is the hyperparameter controlling weights to layer D out of unit i in S, and $\sigma_{a,j}^D$ is the sigma adjustment for unit j in D. For the output layer, u_j^O is the same as v_j^O. When connections are configured, the formula is

$$\frac{\partial^2 L}{\partial (v_i^S)^2} \;\approx\; \sum_D \sum_{(j,k)\,:\,(i,j,k)\in C^{S,D}} (\sigma_w^{S,D})^2\,\frac{\partial^2 L}{(u_j^D)^2} \tag{65}$$

The second derivative of L with respect to a weight, $w_{i,j}^{S,D}$, can be expressed as follows:

$$\frac{\partial^2 L}{\partial (w_{i,j}^{S,D})^2} \;=\; (v_i^S)^2\,\frac{\partial^2 L}{\partial (u_j^D)^2} \tag{66}$$

To estimate this, an estimate of $(v_i^S)^2$ is needed.

When the weight is on a connection from an input unit, $v_i^S = v_i^I$ is the ith input for this training case, which is known. The sum of $(v_i^I)^2$ for all training cases is computed once, for each i, and then used for stepsize heuristics thereafter. From this sum, the mean is also computed, and used as the "typical" squared value of that input unit.

Estimates of "typical" squared values for hidden units are made by simulating a forward pass, in which actual parameter values are not used, but instead parameters are assumed to be typical, given the current hyperparameters. Just one simulation is done for each update needing stepsizes, not one for every training case — the typical squared values are taken to be the same for each training case, approximating (roughly) the mean over training cases.

The typical squared value for a hidden unit with the identity activation function is found from the sum of the previously-estimated typical squared values of the units connecting to this unit (one for a bias connection), with the prior variance of offsets added (if present), multiplied by the prior variance of the weight (or bias) on that connection, as given by the controlling hyperparameter. (For parameters with t distributions, the variance is capped at what it would be if the degrees of freedom were three.)

For a layer of tanh hidden units, the typical squared value for the unit's input found in this way is reduced to one if it exceeds one, since one is the maximum squared value of tanh. For the softplus activation function, if

the typical squared value of the unit's input is less than $\log(2)^2$, the value is increased to $\log(2)^2$, which is the value when the inputs are zero. For units with the softplus0 activation function, the typical squared input value is used unchanged, as for the identity activation function. For normalization layers, units are taken to have a typical squared value for their output equal to the number of units in their group (the maximum possible).

The typical squared value for the input to a unit, $[u_i^\ell]^2$, found as above, is also used for estimating d^2L/du^2 for a normalization layer, as discussed earlier.

Second derivatives with respect to biases are simply equal to the second derivatives with respect to the associated summed input to the hidden or output units. Second derivatives with respect to offsets are equal to the second derivatives with respect to the associated input unit, or with respect to the output of the associated hidden unit.

These heuristic estimates for the second derivatives of L due to each training case with respect to the various network parameters are summed for all cases in the training set. (No actual summation is needed, however, only a multiplication, since the typical squared values used are the same for all training cases.) To these estimates are added estimates of the second derivatives of the log prior probability with respect to each parameter, giving estimates of the second derivatives of the log posterior.

For the second derivative of the log prior with respect to weight $w_{i,j}$, we have

$$-\frac{\partial^2}{\partial w_{i,j}^2} \log P(w_{i,j} \mid \sigma_{w,i},\, \sigma_{a,j}) \quad = \quad \frac{1}{\sigma_{w,i}^2 \sigma_{a,j}^2} \tag{67}$$

if α_2 is infinite, while for finite α_2, we use an estimate analogous to equation (60):

$$-\frac{\partial^2}{\partial w_{i,j}^2} \log P(w_{i,j} \mid \sigma_{w,i},\, \sigma_{a,j}) \quad \approx \quad \frac{\alpha_2 + 1}{\alpha_2 \sigma_{w,i}^2 \sigma_{a,j}^2} \tag{68}$$

Biases and offsets are handled similarly, as are weights for configured connections.

Finally, the stepsize used for a parameter (before the adjustment factor is applied) is the reciprocal of the square root of minus the estimated second derivative of the log posterior with respect to that parameter.

Rejection sampling from the prior

In addition to the Monte Carlo implementations based on Markov chain sampling, a simple Monte Carlo procedure using rejection sampling has also been implemented. This procedure is very inefficient; it is intended for use only as a means of checking the correctness of the Markov chain implementations.

The rejection sampling procedure is based on the idea of generating networks from the prior, and then accepting some of these networks with a probability proportional to the likelihood given the training data of the generated parameter and hyperparameter values, thereby producing a sample from the posterior. For data models with discrete targets, this idea can be implemented directly, as the likelihood is the probability of the targets in the training set, which can be no more than one. For regression models, the likelihood is the probability density of the targets, which can be greater than one, making its direct use as an acceptance probability invalid. If the noise levels for the targets are fixed, however, the likelihood is bounded, and can be used as the acceptance probability after rescaling. For a Gaussian noise model (equation (8)), this is accomplished by simply ignoring the factors of $1 \,/\, \sqrt{2\pi}\sigma_j$ in the likelihood; the analogous procedure can be used for noise from a t-distribution (equation (10)).

When the noise levels are variable hyperparameters, a slightly more elaborate procedure must be used, in which the noise levels are not generated from the prior, but rather from the prior multiplied by a bias factor that gives more weight to higher precisions (lower noise). This bias factor is chosen so that when it is cancelled by a corresponding modification to the acceptance probability, these probabilities end up being no greater than one.

Specifically, the overall noise precision, τ, and the noise precisions for individual targets, the τ_j, are sampled from Gamma distributions obtained by modifying the priors of equations (24) and (25) as follows:

$$\begin{aligned} f(\tau) \quad &\propto \quad \tau^{nm/2}\, P(\tau) \quad \propto \quad \tau^{(\alpha_0+nm)/2-1} \exp\big(-\tau\alpha_0/2\omega\big) \\ f(\tau_j \mid \tau) \quad &\propto \quad \tau_j^{n/2}\, P(\tau_j \mid \tau) \quad \propto \quad \tau^{-(\alpha_1+n)/2}\, \tau_j^{(\alpha_1+n)/2-1} \exp\big(-\tau_j\alpha_1/2\tau\big) \end{aligned} \tag{69}$$

Here, n is the number of training cases and m is the number of targets. The resulting joint sampling density is

$$f(\tau, \{\tau_j\}) \quad = \quad f(\tau) \prod_{j=1}^{m} f(\tau_j \mid \tau) \quad \propto \quad P(\tau, \{\tau_j\}) \prod_{j=1}^{m} \tau_j^{n/2} \tag{70}$$

Since this sampling density is biased in relation to the prior by the factor $\prod_{j=1}^{m} \tau_j^{n/2}$, when constructing the acceptance probability we must multiply the likelihood by the inverse of this factor, $\prod_{j=1}^{m} \tau_j^{-n/2} = \prod_{c=1}^{n} \prod_{j=1}^{m} \sigma_j$. This cancels the factors of $1/\sigma_j$ in the target probabilities of equations (8) and (10), leaving an acceptance probability which is bounded, and can be adjusted to be no more than one by ignoring the remaining constant factors.