Training Stochastic Feedforward Binary Neural Networks with a Gibbs-flavored MCMC Strategy

Victor Biaggi

October 17, 2017

Here we try to design an algorithm training Stochastic Feedforward Binary Neural Networks (SFBNN) with a Gibbs-flavored MCMC strategy. These networks are standard feedforward neural network except that they sample instead of average.

Let L the number of hidden layers of our SFBNN (our algorithm will be implemented in the general case and tested for $L = 1$), $W = (W^1, .., W^L, W^O)$ the weights of our SFBNN and $H = (h^1, ..., h^L)$ its nodes.

In a SFBNN, the l^{th} hidden layer is defined as :

$$h^l | W^l \propto Bern(\sigma(W^l h^{l-1}))$$

$$y \propto \mathcal{N}(W^O h^L, \sigma^2)$$

This led us to consider a Gibbs-flavored MCMC scheme for SFBNN, which is the following algorithm :

$\triangle$ Suppose that after iteration t, we have the following data :

$$W_t = (W_t^1, .., W_t^L, W_t^O)$$

$$H_t = (h_t^1, ..., h_t^L)$$

$\triangle$ Then, we want to sample

$$(h_{t+1}^1)_1, ..., (h_{t+1}^1)_{l_1}, (h_{t+1}^2)_1, ..., (h_{t+1}^2)_{l_2}, ..., (h_{t+1}^L)_1, ..., (h_{t+1}^L)_{l_L}$$

from

$$p(h_i^k | h_{j \neq i}, h^{k+1}, h^{k-1}, W, X, y)$$

Where $l_1, ..., l_L$ are the number of nodes for each layer

We note that this can be done via Metropolis-Hastings(see below H update)

$\triangle$ Finally, given H_{t+1}, we sample (or train) our parameters to yield W_{t+1} using SGD Langevin Diffusion or Bayesian regression.

We would run enough iterations to train the networks.

We keep in mind that it is more suitable to learn a model that can carry out a sequence of stochastic operations internally, in order to represent a complex stochastic process.

1 H update

The key idea :

$$\begin{aligned} p(h_i^k|h_{j\neq i}, h^{k+1}, h^{k-1}, W, X, y) &\propto p(h_i^k|h_{j\neq i}, h^{k+1}, h^{k-1}, W^k, W^{k+1}) \\ &\propto p(h_i^k|h_{j\neq i}^k, h^{k-1}, W^k)\ p(h^{k+1}|h^k, W^{k+1}) \\ &\propto Bern(\sigma(W^k h^{k-1}))_i\ p(h^{k+1}|h^k, W^{k+1}) \end{aligned}$$

with "$h^{k-1} = X$" if $k = 1$ and $h^{k+1} = y$ if $k = L$

$\clubsuit$ If $k < L$:

$$p(h_i^k|h_{j\neq i}, h^{k+1}, h^{k-1}, W, X, y) \propto Bern(\sigma(W^k h^{k-1}))_i \prod_j Bern(\sigma(W^{k+1}h^k))_j$$

Thus the A-R ratio is :

$$\alpha = min(1, \frac{P(Bern(\sigma(W^k h^{k-1}))_i = h_{i,new}^k)\ \prod_j P(Bern(\sigma(W^{k+1}h_{new}^k)_j) = h_j^{k+1})}{P(Bern(\sigma(W^k h^{k-1}))_i = h_i^k)\ \prod_j P(Bern(\sigma(W^{k+1}h^k)_j) = h_j^{k+1})})$$

Since,

$$\forall a \in [0,1],\ \forall \theta \in \{0,1\},\ P(Bern(a) = \theta) = a\ \theta\ +\ (1-a)\ (1-\theta) = B(a,\theta)$$

Therefore,

$$\alpha = min(1, \frac{B(\sigma(W^k h^{k-1})_i, h_{i,new}^k)\ \prod_j B(\sigma(W^{k+1}h_{new}^k)_j, h_j^{k+1})}{B(\sigma(W^k h^{k-1})_i, h_i^k)\ \prod_j B(\sigma(W^{k+1}h^k)_j, h_j^{k+1})}) \quad (E)$$

Note : Sampling at the bottom of the network may be an issue but we can try to sample for all x then A-R with a new ratio from these samplings - we compute the sum of the results S_x for all elements in X then do a final A-R with the final ratio S_x/l_x where l_x is the number of training examples in X. As in the note below, this method could be discussed.

♣ If $k = L$, considering y as a single training example (see the note below) :

$$p(h_i^k|h_{j\neq i}, h^{k+1}, h^{k-1}, W, X, y) \propto Bern(\sigma(W^k h^{k-1}))_i \ \mathcal{N}(y \mid W^O h^L, \sigma^2)$$

Therefore, the A-R ratio in this case is :

$$\alpha = min(1, \frac{B(\sigma(W^k h^{k-1})_i, h_{i,new}^k) \ \mathcal{N}(y \mid W^O h_{new}^L, \sigma^2)}{B(\sigma(W^k h^{k-1})_i, h_i^k) \ \mathcal{N}(y \mid W^O h^L, \sigma^2)}) \quad (E^{'})$$

Note : At the top, we propose a similar procedure - we compute the sum of the results S_y for all elements in y then do a final A-R with the final ratio S_y/l_y where l_y is the lenght of y. I am not sure about this method, we could also do the product of the normals in (E') for all the elements in y, this will have to be discussed.

2 W update

a) Bayesian Logistic / Sevan Mixture Gaussian

b) Or the SGD Langevin Diffusion