Heuristics to improve performance of the Backpropagation algorithm[1]

i) **Use sequential mode of learning**. This mode of learning enables weights and biases to update after presentation of a pattern within a training epoch. As soon as we discover that a certain pattern did not produce the desired output pattern, we change the parameters. Compare this with the batch mode where the entire set of training instances are shown to the network and parameters are updated only after the epoch.

ii) **Perturb the order of presentation of the input patterns** to the network. This randomization will tend to make the search in the parameter space stochastic thus possibly avoiding local minima. A combination of sequential mode of learning and the randomization of the order of presentation of the input is sometimes referred to as 'stochastic gradient descent' or 'SGD'[2].

iii) **Use the momentum factor**. Momentum factor uses a fraction of the last weight update with the current one. This helps create 'momentum' effect thereby making it possible to get out of a local extrema. Momentum is applied as follows:

$$w_{ij}{}^{t+1} = w_{ij}{}^{t} + \Delta w_{ij}{}^{t} + \beta(\Delta w_{ij}{}^{t-1})$$; where 't' represents previous pattern or epoch.

iv) **Use an antisymmetric output function**. For faster convergence of network parameters, use an antisymmetric logistic function, aka tangent sigmoid or hyperbolic tangent. The output for such a function is as follows:

$$\sigma(-x) = -\sigma(x)$$

v) **Choose target values within the range of the sigmoid activation**. It is important to take this step as it will ensure that the backpropagation algorithm may not drive the parameters to infinity. For example, in case of hyperbolic tangent output function, we can choose the desired response of the j^{th} node in the output layer as follows:

$d_j = a \pm \epsilon$; where a is the limiting value of the hyperbolic tangent function and can be set at ±1.72 and ϵ at ±0.72. This setting would conveniently make the value of d_j=±1

vi) **Normalize the input patterns**. This should be done such that the mean (averaged over the entire training set) of each input variable is as close to zero or it's smaller than its standard deviation. Moreover, the input variables should be decorrelated and should be scaled such that their covariances are equal. This preprocessing should help in faster parameter convergence.

vii) **Parameter initialization should be done carefully**. Avoid using very large or small values for initialization of parameters. The weights and biases should be initialized from a uniform distribution with

---

[1] Adapted from Simon Haykin's book. Neural Networks: A Comprehensive Foundation, 2nd Ed.

[2] There is a common variant called 'mini-batch' gradient descent, which provides a compromise between stochastic and full-batch gradient descent.

zero mean and variance equal to the reciprocal of the number of synaptic connections 'm' of a neuron, i.e., $\sigma_w = \frac{1}{\sqrt{m}}$

viii) **Learning rate**. The learning rate 'c' should vary across layers and within the layer at the level of a neuron. It should be set smaller in the last layers (due to larger error gradients) and smaller in the front layers. Within a layer, it should be inversely proportional to the square root of the number of weights.