**Assignment**

*Backpropagation* is an efficient method of computing gradients in directed graphs of computations, such as neural networks.

There are some drawbacks to backpropagation. Backpropagation works well on simple training problems. Complex spaces have nearly global minima which are sparse among the local minima. *Gradient Descent* type search techniques (used in Neural networks) tend to get trapped at local minima. With a high enough gain (or momentum), backpropagation can escape these local minima. However, it leaves them without knowing whether the next one it finds will be better or worse. When the nearly global minima are well hidden among the local minima, backpropagation can end up bouncing between local minima without much overall improvement, thus making for very slow training.

In this assignment we are using a technique called Eagle Strategy, which involves switching between random walk and careful walk. The switch to careful walk is with some probability. We have used simulated annealing for the global optimization and hill climb for local search.

*Simulated annealing* is a well-known optimization technique. It is based on the analogy with annealing in physics, where the changes of the system state are seen as essentially random, but changes that reduce the energy are more likely than those that increase it.

In addition, changes that increase the energy are more likely while the temperature is high than later when the temperature is low.

Our main evaluation measure is the fitness of the weights i.e., energy function(accuracy) which is the most difficult to maximize.

Since simulated annealing is a global search it is recommended as an alternative training technique for backpropagation.

Drawbacks of simulated annealing is that it is very slow, especially if the function to maximize is expensive to compute.

For problems where there are few local minima, **SA is overkill**. Simpler, faster methods like gradient descent will work better. But we usually don't know what the energy landscape is.

In our case we are trying to maximize the function *f* which is the accuracy. Let *O* be the initial output of function *f* and *O*’ be the output of function f’.

If accuracy returned by the function *f*’ i.e., *X*’ is greater than *X* then we say *X*’ is the new solution. If *f*’ is greater than or equal to *f*, then we go to *X*’ with some probability defined in the below equation,

*P=e(f’-f)/T*

*Hill climbing* looks one step ahead to determine if any successor is better

than the current state; if there is, move to the best successor. It does not allow

backtracking or jumping to an alternative path since it doesn’t remember where it has been.

|  |
| --- |
| **Algorithm:** Eagle Strategy |

**Result:** Best Solution for *f*(*x*)

**START.**

**Input:** Model, X\_train, y\_train

**while**  !stop criteria **do**

Global exploration by Simulated Annealing

Evaluate the objectives and find a promising solution

If *pe* <rand,

switch to a Hill Climbing

Intensive Hill Climbing around a promising solution

if (a better solution is found)

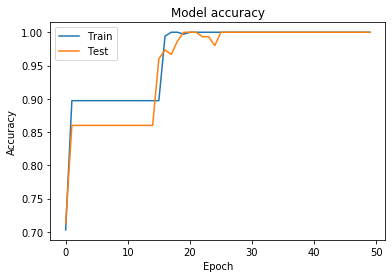
Update the current best

**Output:** Weights giving best accuracy.

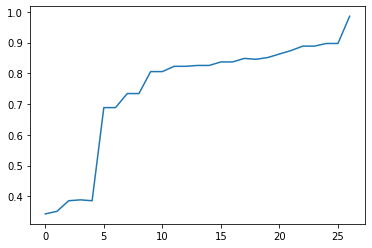
**STOP.**

**Keras Backpropagation Model Accuracy for given code:**

**Figure 1.**



**Model Accuracy of Simulated Annealing and Hill Climbing using eagle strategy: Figure 2.**

****

Accuracy

Epoch

**Explanation:**

Since backpropagation uses gradient descent type technique it tends to get stuck at local optima which can be seen in Figure 1. The Keras BP model gets stuck at 0.9 accuracy for approximately 15 epochs to come out of local optima using momentum whereas our model using eagle strategy has been able to take off from the local optima in less than 3 iterations.

**Conclusion**

Since Simulated annealing performs random walk it has better chances of convergence to the global maximum within iterations fewer than keras’s BP.