PSO-NeuralNet Document

Nafis Neehal

6 August 2018

1 Network Description

Artificial Neural Network: Dataset- MNIST, Training Data - 42,000, Test Data-10,000

Layer	Number of Nodes	Layer Dim	Weight Dim	Bias Dim	Activation
Input	784	784x1	None	None	None
Hidden 1	16	16x1	16x784	16x1	ReLU
Hidden 2	16	16x1	16x16	16x1	ReLU
Output	10	10x1	10x16	10x1	Sigmoid

Table 1: Artificial Neural Network Structure

2 Forward Propagation

Algorithm 1 Forward Propagation in Neural Network

procedure $\textsc{ForwardPropagation}(X)$ ▷ Forward Propagation
  $Z^{[1]} = W^{[1]}.X + b^{[1]}$
  $A^{[1]} = \text{ReLU}(Z^{[1]})$
  $Z^{[2]} = W^{[2]}.A^{[1]} + b^{[2]}$
  $A^{[2]} = \text{ReLU}(Z^{[2]})$
  $Z^{[3]} = W^{[3]}.A^{[2]} + b^{[3]}$
  $A^{[3]} = \text{Sigmoid}(Z^{[3]})$
end procedure

The notations that have been used in forward propagation is described below:

- $Z^{[i]}$ = Output of i^{th} Layer of Neural Network
- $A^{[i]}$ = Activation of i^{th} Layer of Neural Network
- X = input

- $W^{[i]}$ = Weight from (i-1)th to ith Layer of Neural Network
- $b^{[i]}$ = Biases for ith Layer of Neural Network

3 Back Propagation

Algorithm 2 Back Propagation in Neural Network

procedure BACKPROPAGATION(X, Y)
  $dZ^{[3]} = A^{[3]}$ - Y ▷ Derivatives Calculation
  $dW^{[3]}$ = (1/m) $dZ^{[3]}.A^{[2].T}$
  $dB^{[3]}$ = (1/m) np.sum($dZ^{[3]}$, axis=1, Keepdims = True)
  $dZ^{[2]} = W^{[3].T}.dZ^{[3]}$ * DReLU($Z^{[2]}$)
  $dW^{[2]}$ = (1/m) $dZ^{[2]}.A^{[1].T}$
  $dB^{[2]}$ = (1/m) np.sum($dZ^{[2]}$, axis=1, Keepdims = True)
  $dZ^{[1]} = W^{[2].T}.dZ^{[2]}$ * DReLU($Z^{[1]}$)
  $dW^{[1]}$ = (1/m) $dZ^{[1]}.X^{T}$
  $dB^{[1]}$ = (1/m) np.sum($dZ^{[1]}$, axis=1, Keepdims = True)

  $W^{[1]} = W^{[1]}$ - 0.01 * $dW^{[1]}$ ▷ Parameter Update
  $b^{[1]} = b^{[1]}$ - 0.001 * $dB^{[1]}$
  $W^{[2]} = W^{[1]}$ - 0.01 * $dW^{[2]}$
  $b^{[2]} = b^{[1]}$ - 0.001 * $dB^{[2]}$
  $W^{[3]} = W^{[1]}$ - 0.01 * $dW^{[3]}$
  $b^{[3]} = b^{[1]}$ - 0.001 * $dB^{[3]}$
end procedure

The notations that have been used in back propagation is described below:

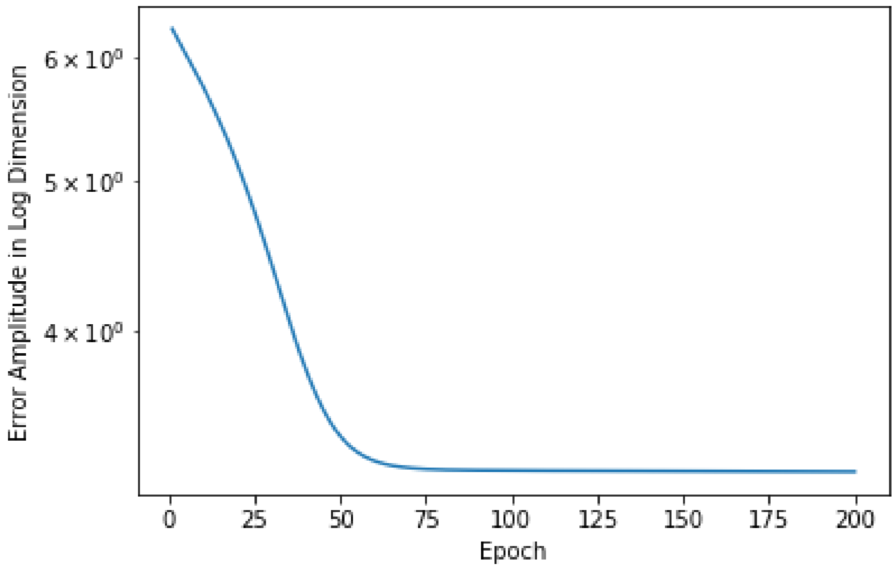
- $dZ^{[i]}$ = Derivative of Z of ith Layer of Neural Network
- $A^{[i].T}$ = Transpose of $A^{[i]}$
- $dW^{[i]}$ = Derivative of W of ith Layer of Neural Network
- $dB^{[i]}$ = Derivative of B of ith Layer of Neural Network

4 Loss Function for Neural Network

Algorithm 3 Loss Function in Neural Network

procedure LOSSFUNCTION(Y)
  RETURN $\frac{1}{m}\sum_{i=1}^{m}(-Y^{(i)}\log(\hat{Y^{(i)}}) - (1-Y^{(i)})\log(1-\hat{Y^{(i)}})$
end procedure

5 Performance of Back Propagation



6 Particle Swarm Optimization as a replacement of Backpropagation in Neural Network

Main Equation of Particle Swarm Optimization -

$\mathrm{WV_{ij}(t)}$ = Intertia Component
$\mathrm{r_1c_1(P_{ij}(t) - x_{ij}(t))}$ = Cognitive Component
$\mathrm{r_2c_2(g_j(t) - x_{ij}(t))}$ = Social Component
$\mathrm{c_1}$, $\mathrm{c_2}$ = Acceleration Coefficients
W = Inertia Coefficient
t, t(t+1) = timestamps
$\mathrm{V_{ij}(t)}$ = $\mathrm{j^{th}}$ velocity component of $\mathrm{i^{th}}$ particle
$\mathrm{P_{ij}(t)}$ = $\mathrm{j^{th}}$ component of $\mathrm{i^{th}}$ particle's best position
$\mathrm{x_{ij}(t)}$ = $\mathrm{j^{th}}$ component of $\mathrm{i^{th}}$ particle's current position
$\mathrm{r_1}$, $\mathrm{r_2}$ = $N(\mu, \sigma^2)$

For determining the value of W, $\mathrm{c_1}$ and $\mathrm{c_2}$, I've used Constriction coefficients proposed by Clerc and Kennedy, 2002.

Let,

$$\phi = \phi_1 + \phi_2 \geq 4$$

$$0 \leq \kappa \leq 1$$

$$\chi = \frac{2\kappa}{\left|2 - \phi - \sqrt{\phi^2 - 4\phi}\right|}$$

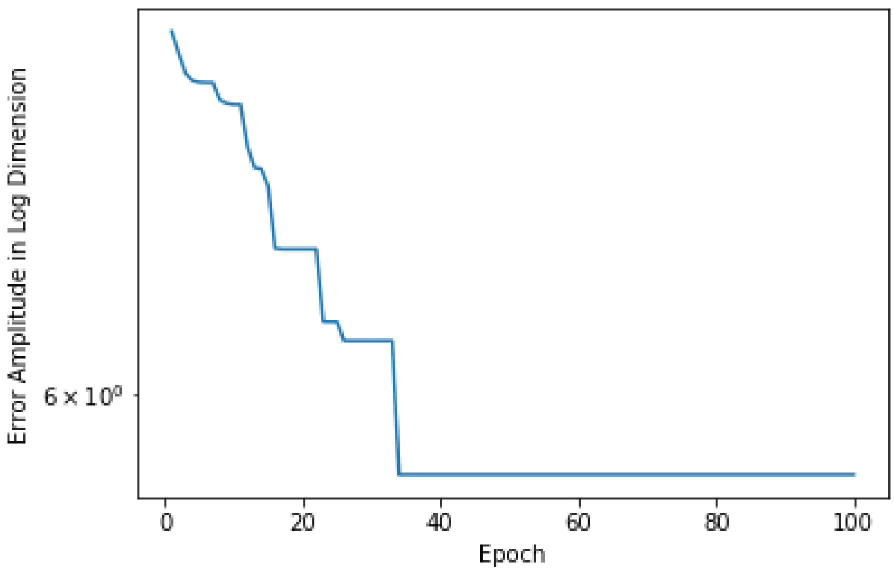
Commonly, $\kappa = 1$, $\phi_1 = 2.05$, $\phi_2 = 2.05$

So, Finally,
W = χ, $C_1 = \kappa\phi_1$, $C_2 = \kappa\phi_2$

And for Velocity and Position Update,

$$V_{ij}(t+1) = WV_{ij}(t) + r_1c_1(P_{ij}(t) - x_{ij}(t)) + r_2c_2(g_j(t) - x_{ij}(t))$$
$$x_{ij}(t+1) = x_{ij}(t) + V_{ij}(t+1)$$

Performance of PSO is shown in the following graph. Swarm Population Size = 50, Total Epochs = 100.



Algorithm 4 Particle Swarm Optimization in Neural Network

procedure InitializeParticle(nVar, nPop)
    globalBestPosition $\leftarrow \vec{0}$ of dimension (nVar x 1)
    globalBestCost $\leftarrow$ None
    for each particle$_{(i)}$ in range(1, nPop) **do**
        particle$_{(i)}$.position $\leftarrow N(\mu, \sigma^2)$ of dimension (nVar x 1)
        particle$_{(i)}$.velocity $\leftarrow \vec{0}$ of dimension (nVar x 1)
        particle$_{(i)}$.cost $\leftarrow$ None
        particle$_{(i)}$.bestPosition $\leftarrow$ particle$_{(i)}$.position
        particle$_{(i)}$.bestCost $\leftarrow$ particle$_{(i)}$.cost
    end for
end procedure
procedure InitializeParameters($NeuralNetwork, X, Y, maxIt, nPop$)
    nVar $\leftarrow$ NeuralNetwork.numWeights + NeuralNetwork.numBiases
    $\kappa \leftarrow 1$
    $\phi_1 \leftarrow 2.05$
    $\phi_2 \leftarrow 2.05$
    $\phi \leftarrow \phi_1 + \phi_2$
    $\chi \leftarrow \frac{2\kappa}{\left|2-\phi-\sqrt{\phi^2-4\phi}\right|}$
    maxIt $\leftarrow$ 100
    nPop $\leftarrow$ 50
    w $\leftarrow \chi$
    wdamp $\leftarrow$ 0.99
    $c_1 \leftarrow \chi.\phi_1$
    $c_2 \leftarrow \chi.\phi_2$
end procedure
procedure COSTPSO($NeuralNetwork, particle$)
    NeuralNetwork.weights $\leftarrow$ particle.position.slice
    NeuralNetwork.biases $\leftarrow$ particle.position.slice
    NeuralNetwork.ForwardPropagation(X)
    RETURN NeuralNetwork.Loss(Y)
end procedure
procedure InitializeSwarm($NeuralNetwork$)
    for each particle$_{(i)}$ in range(1, nPop) **do**
        particle$_{(i)}$.cost $\leftarrow$ COSTPSO(NeuralNetwork, particle$_{(i)}$)
        if particle$_{(i)}$.bestCost $< globalBestCost$ **then**
            globalBestPosition $\leftarrow$ particle$_{(i)}$.bestPosition
            globalBestCost $\leftarrow$ particle$_{(i)}$.bestCost
        end if
    end for
end procedure

Algorithm 5 Particle Swarm Optimization in Neural Network

procedure PSOLOOP($NeuralNetwork$)
    for each epoch$_{(\mathrm{i})}$ in range(1, maxIt) **do**
        for each particle$_{(\mathrm{j})}$ in range(1, nPop) **do**
            $V_{\mathrm{j}}(t+1) \leftarrow WV_{\mathrm{j}}(t) + r_1 c_1 (P_{\mathrm{j}}(t) - x_{\mathrm{j}}(t)) + r_2 c_2 (g(t) - x_{\mathrm{j}}(t))$
            $x_{\mathrm{j}}(t+1) \leftarrow x_{\mathrm{j}}(t) + V_{\mathrm{j}}(t+1)$
            particle$_{(\mathrm{j})}$.cost $\leftarrow$ COSTPSO(NeuralNetwork,particle$_{\mathrm{j}}$)
            if particle$_{(\mathrm{j})}$.cost <particle$_{(\mathrm{j})}$.bestCost **then**
                particle$_{(\mathrm{j})}$.bestPoition $\leftarrow$ particle$_{(\mathrm{j})}$.position
                particle$_{(\mathrm{j})}$.besCost $\leftarrow$ particle$_{(\mathrm{j})}$.cost
                if particle$_{(\mathrm{j})}$.bestCost <globalBestCost **then**
                    globalBestPosition $\leftarrow$ particle$_{(\mathrm{j})}$.bestPosition
                    globalBestCost $\leftarrow$ particle$_{(\mathrm{j})}$.cost.bestCost
                end if
            end if
        end for
    end for
end procedure