Sheet 5

November 23, 2015

Assignment 24

The figure 1 represents the typical learning curve for a MLP trained with Backpropagation of Error using the single step learning strategy. As we can see there is a first stage where the error decreases a little bit, and after that we can see that this change happen faster. After this, the change will be slow. And finally, we will reach a plateu.

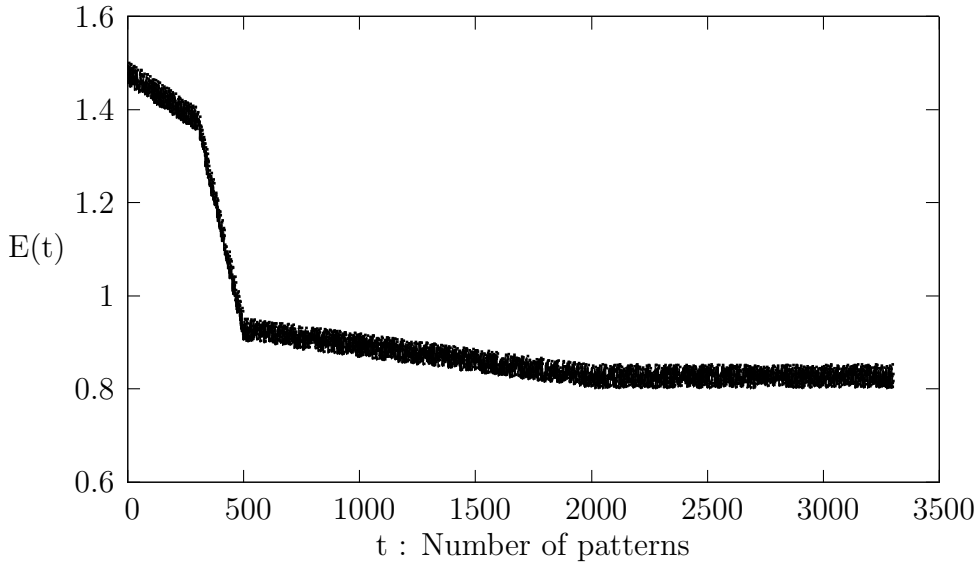

Figure 1: Typical learning curve for an MLP

Assignment 25

The differences in each of these graphs show us that speed and accuracy of learning can be controlled by scaling the axes. In the figure 3 we can observe that when the axe X is logarithimic scaled won't be easy to reach the negative gradient.

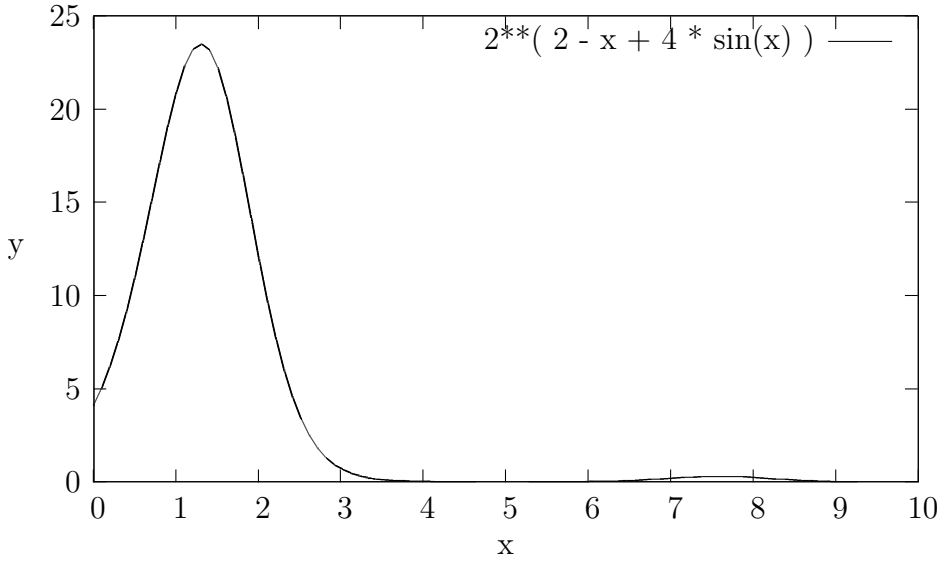

Figure 2: X and Y axes scaled linear

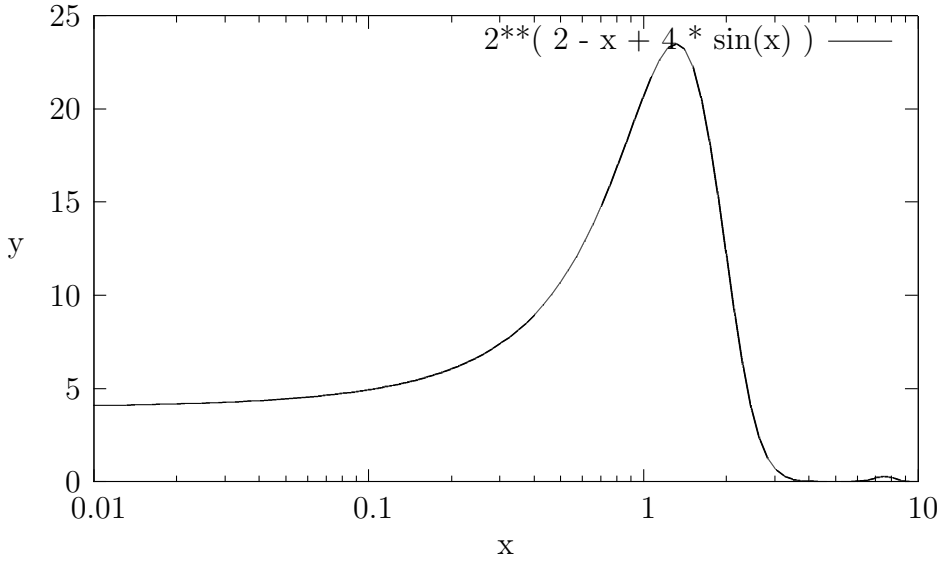

Figure 3: X axe scaled logarithmic

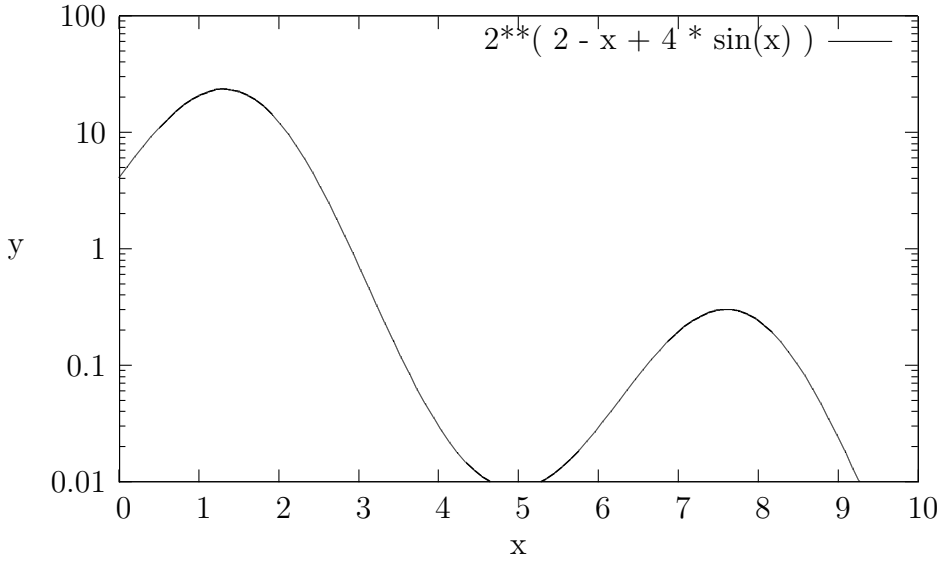

Figure 4: Y axe scaled logarithmic

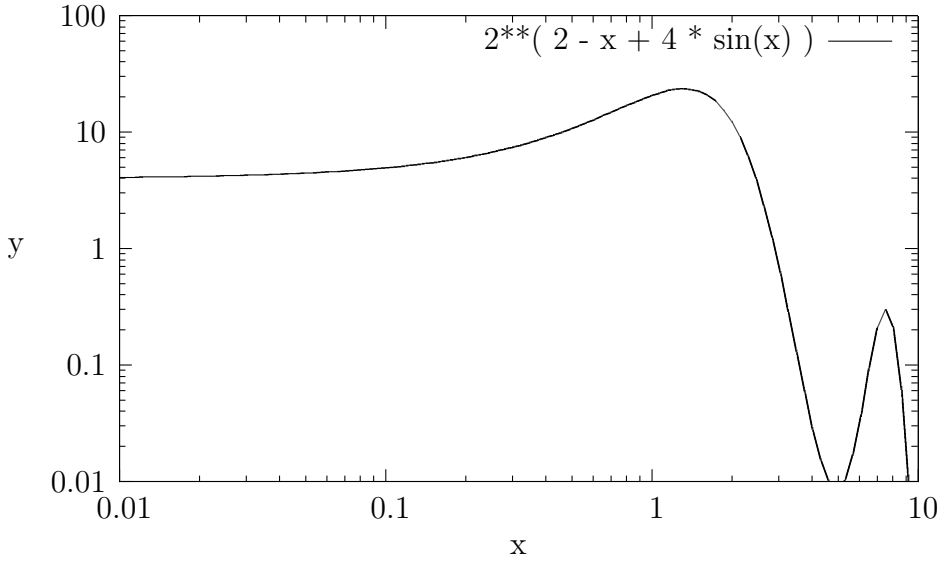

Figure 5: X and Y axes scaled logarithmic

Assignment 26: Resilient-PROP algorithm

The idea of RPROP algorithm [1] is to change the size of the weight-update Δw_{ij} without considering the size of the partial derivative. The algorithms which are proposed to deal with the problem of appropiate weight-update by doing some sort of parameters adaptation during learning still have some failures since that the size of the actually taken weight-step Δw_{ij} is not only depending on the learning rate, but also on the partial derivative $\frac{\partial E}{\partial w_{ij}}$. The above problem can lead to oscillation, preventing the error to fall below a certain value.

Detail,

$$ {}^{p}\Delta_{ij} = \begin{cases} \eta^{+} \cdot {}^{p-1}\Delta_{ij}, & if\ {}^{p-1}(\frac{\partial E}{\partial w_{ij}}) \cdot {}^{p}(\frac{\partial E}{\partial w_{ij}}) > 0 \\ \eta^{-} \cdot {}^{p-1}\Delta_{ij}, & if\ {}^{p-1}(\frac{\partial E}{\partial w_{ij}}) \cdot {}^{p}(\frac{\partial E}{\partial w_{ij}}) < 0 \\ {}^{p-1}\Delta_{ij}, & otherwise \end{cases} \tag{1} $$

Where, $0 < \eta^{-} < 1 < \eta^{+}$, here η^{-} is the decrease factor and η^{+} is the increase factor.

$$ {}^{p}\Delta w_{ij} = \begin{cases} -\,{}^{p}\Delta_{ij}, & if\ {}^{p}(\frac{\partial E}{\partial w_{ij}}) > 0 \\ +\,{}^{p}\Delta_{ij}, & if\ {}^{p}(\frac{\partial e}{\partial w_{ij}}) < 0 \\ 0, & otherwise \\ -\,{}^{p-1}\Delta w_{ij}, & if\ {}^{p-1}(\frac{\partial E}{\partial w_{ij}}) \cdot {}^{p}(\frac{\partial E}{\partial w_{ij}}) < 0 \end{cases} \tag{2} $$

$$ if\ {}^{p-1}(\frac{\partial E}{\partial w_{ij}}) \cdot {}^{p}(\frac{\partial E}{\partial w_{ij}}) < 0, {}^{p}(\frac{\partial E}{\partial w_{ij}}) = 0 $$

$$ {}^{p-1}w_{ij} = {}^{p}w_{ij} + {}^{p}\Delta w_{ij} $$

If gradient $\frac{\partial E}{\partial w_{ij}}$ retains its sign, the update-value is slightly increased in order to accelerate convergence in shallow regions.

$$ \Rightarrow {}^{p}\Delta_{ij} = \eta^{+} \cdot {}^{p-1}\Delta_{ij},\ if\ {}^{p-1}(\frac{\partial E}{\partial w_{ij}}) \cdot {}^{p}(\frac{\partial E}{\partial w_{ij}}) > 0 \tag{3} $$

$$ \Rightarrow {}^{p}\Delta w_{ij} = \begin{cases} -\,{}^{p}\Delta_{ij},\ if\ derivative > 0 \\ +\,{}^{p}\Delta_{ij},\ if\ derivative < 0 \end{cases} \tag{4} $$

If gradient changes its sign, that indicates that the last update was too big and the algorithm has jumped over a local minimun, the update-value Δ_{ij} is decreased by the factor η^{-}.

$$ \Rightarrow {}^{p}\Delta_{ij} = \eta^{-} \cdot {}^{p-1}\Delta_{ij} $$

and the previous weight-update is reverted which is denoted "back-tracking" weight-step.

$$ {}^{p}\Delta w_{ij} = -\,{}^{p-1}\Delta w_{ij} $$

$$ {}^{p}(\frac{\partial E}{\partial w_{ij}}) = 0 $$

References

[1] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The rprop algorithm," in *Neural Networks, 1993., IEEE International Conference on*, pp. 586–591, IEEE, 1993.

Assignment 27

Advantages of momentum

- Avoid oscillation: Single step learning can lead to oscillation when in steep valleys, because of the large gradient. In contrast, cumulative learning can reduce the step size, by adding the weight change of previous step.
- Accelerate on plateaus: Single step learning keeps its speed of covergence even if the step is on flat plateaus, but cumulative learning increases the step size on flat plateaus by adding the weight change of previous step to the new one.

Disadvantages of momentum

- Setting the momentum parameter *too high* can create a risk of overshooting the minimum.
- Setting the momentum parameter *too low* can not reliably avoid local minima, and slow down the speed of covergence.

0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Table 1: Different outputs for a layer with three neurons and binary output

Assignment 28

MLP 8-3-8 task

$$X \in 1 - out - of - 8\, binary\, coding$$

$$Y = X$$

A three hidden layer whose neurons have binary output can represent 8 combinations. See table 1.

So, it is enough to have three neurons in the hidden layer to emulate a encoder-decoder.

MLP 8-2-8 task

Given a hidden layer h with two neurons:

$$net_h = w_0 + w_1 \cdot h_1 + w_2 \cdot h_2$$

We can draw a line to separate eight points using a function like this:

$$h_2 = \begin{array}{c} -\frac{w_0+w_1}{w_2} \\ -\frac{w_1}{w_2} \cdot h_1 - \frac{w_0}{w_2} \end{array} \tag{5}$$

MLP 8-1-8 task

Eight different states are eight different positions in (out h). The net between hidden layer and output layer can implement a linear separation. However it does not assure that is linear separable from all others.

Assignment 29

$$
\begin{aligned}
\frac{\partial E}{\partial x_n} &= \frac{\partial E}{\partial net_n} \cdot && \frac{\partial \mathbf{net_n}}{\partial \mathbf{x_n}} \\
&\quad \vdots && \frac{\partial}{\partial x_n} \sum_{i=0}^{n} x_i \cdot w_{ih} \\
&\quad \vdots && w_{nh} \ (when\ i = n) \\
&= \frac{\partial E}{\partial net_n} && \cdot w_{nh} \\
&= \frac{\partial E}{\partial out_n} && \cdot \frac{\partial out_h}{\partial net_h} \cdot w_{nh} \\
&\quad \vdots && \frac{\partial f(net_h)}{\partial net_h} = f'(net_h) \\
&= \frac{\partial E}{\partial out_n} && \cdot f'(net_h) \cdot w_{nh} \\
&= \sum_{k=1}^{K} \frac{\partial E}{\partial net_k} \cdot \frac{\partial net_k}{\partial out_n} && (6) \\
&= \ \vdots && \frac{\partial}{\partial out_h} \cdot \sum_{j=0}^{H} out_j \cdot w_{jk} = w_{hk} \\
&= \frac{\partial E}{\partial y_m} \cdot \frac{y_m}{\partial net_m} \\
&= \frac{\partial}{\partial y_m} \cdot \frac{1}{2} \sum_{j=1}^{M} (\hat{y}_j - y_j)^2 \\
&= \frac{1}{2} \frac{\partial}{\partial y_m} (\hat{y}_m - y_m)^2 \\
&= \frac{1}{2} \cdot 2 \cdot (\hat{y}_m - y_m) \cdot \frac{\partial}{\partial y_m} \cdot (-y_m) \\
&= -(\hat{y}_m - y_m)
\end{aligned}
$$

Conclusion,

$$\sum_{h=1}^{M} -(\hat{y}_m - y_m) \cdot f'(net_m) \cdot w_{hm} \cdot f'(net_h) \cdot w_{nh}.$$