Backpropagation I

Liangjie Cao

5 May 2018

Today I learn the process of Backpropagation to train multilayer architectures. The key insight is that the derivative (or gradient) of the objective with respect to the input of a module can be computed by working backwards from the gradient with respect to the output of that module (or the input of the subsequent module) in Fig. 1.1

In the late 1990s, neural nets and backpropagation were largely forsaken by the machine-learning community and ignored by the computer vision and speech recognition communities. It was widely thought that learning useful, multistage, feature extractors with little prior knowledge was infeasible. In particular, it was commonly thought that simple gradient descent would get trapped in poor local minima —weight configurations for which no small change would reduce the average error. [1]And the paper says that is not a simple probelm.

Then around 2006, cifar brought together researchers and renewed interest in deep feedforward neural networks. Researchers propose an unsupervised learning1 approach that creates network layers to detect features without using tagged data. These network layers can be used to reconstruct or model the activity of feature detectors. Through the pre-training process. The weights of the deep network can be initialized to interesting values. An output layer is then added to the top of the network and finetuned using standard back propagation algorithms. I will continue learning in the following days.

1	Cluster (e. g., hybrid models)
2	, hierarchical cluster

Table 1: **Classifition of unsupervised learning**

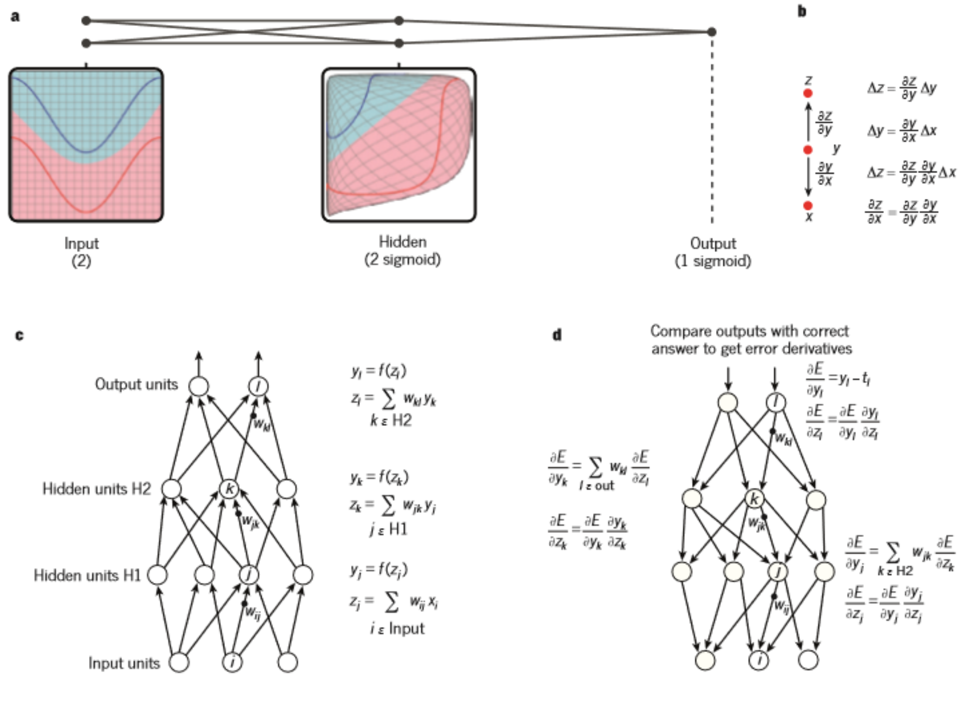
a
Input
(2)
Hidden
(2 sigmoid)
Output
(1 sigmoid)
b
z
y
x
$\frac{\partial z}{\partial y}$
$\frac{\partial y}{\partial x}$
$\Delta z = \frac{\partial z}{\partial y}\Delta y$
$\Delta y = \frac{\partial y}{\partial x}\Delta x$
$\Delta z = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}\Delta x$
$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$
c
Output units
Hidden units H2
Hidden units H1
Input units
w_{kl}
w_{jk}
w_{ij}
$y_l = f(z_l)$
$z_l = \sum_{k \in H2} w_{kl} y_k$
$y_k = f(z_k)$
$z_k = \sum_{j \in H1} w_{jk} y_j$
$y_j = f(z_j)$
$z_j = \sum_{i \in Input} w_{ij} x_i$
d
Compare outputs with correct answer to get error derivatives
$\frac{\partial E}{\partial y_l} = y_l - t_l$
$\frac{\partial E}{\partial z_l} = \frac{\partial E}{\partial y_l}\frac{\partial y_l}{\partial z_l}$
$\frac{\partial E}{\partial y_k} = \sum_{l \in out} w_{kl}\frac{\partial E}{\partial z_l}$
$\frac{\partial E}{\partial z_k} = \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial z_k}$
$\frac{\partial E}{\partial y_j} = \sum_{k \in H2} w_{jk}\frac{\partial E}{\partial z_k}$
$\frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial z_j}$

Figure 1: **Multilayer neural networks and backpropagation.**

References

[1] Yoshua Bengio& Geoffrey Hinton Yann LeCun. Deep learning. *Nature*, 521(28):9, 2015.