On Gradient

Giulio Galvan

16 marzo 2015

Sommario

1 How to compute gradient

1.1 Backpropagation

First of all we need to define a loss function over the training data, so we define a dataset as

$$D = \{x^{(i)} \in \mathbb{R}^p, y^{(i)} \in \mathbb{R}^q, i \in [1, N]\} \tag{1}$$

and the loss function as

$$L(W) = \frac{1}{N} \sum_{i=1}^{N} L^{(i)}(W) \tag{2}$$

where W is represents all the weights of the net. The network is defined by

$$a_l = \sum_j w_{lj} \phi_j \tag{3}$$

$$\phi_l = \sigma(a_l) \tag{4}$$

where w_{lj} is the weight of the connection between neuron j and neuron l and σ is the non linear activation function

So we can compute partial derivatives with respect to a single weight w_{lj}, using simply the chain ruly, as

$$\frac{\partial L^{(i)}}{\partial w_{lj}} = \frac{\partial L^{(i)}}{\partial a_l} \cdot \frac{\partial a_l}{\partial w_{lj}} = \delta_l \cdot \phi_j$$

where we put

$$\delta_l \triangleq \frac{\partial L^{(i)}}{\partial a_l}$$

So we can easily compute $\delta_u = \frac{\partial L^{(i)}}{\partial a_u}$ for each output unit u once we choose a differtiable loss function; note that we dont need the weights for such a computation.

Let $P(l)$ be the set of parents of neuron l, formally:

$$P(l) = \{k : \exists \text{ a link between } l \text{ and } k \text{ with weight } w_{lk}\} \tag{5}$$

Again, simply using the chain rule, we can write:

$$\delta_l = \sum_{k \in P(l)} \frac{\partial L^{(i)}}{\partial a_k} \cdot \frac{\partial a_k}{\partial a_l} = \sum_{k \in P(l)} \delta_k \cdot \frac{\partial a_k}{\partial \phi_l} \cdot \frac{\partial \phi_l}{\partial a_l} = \sum_{k \in P(l)} \delta_k \cdot w_{kl} \cdot \sigma'(a_l)$$

Riferimenti bibliografici

[1] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory, 1995.