On recurrent neural networks

Giulio Galvan

20th March 2015

Contents

Chapter 1

Artificial neural networks model

1.1 Feed foward neural networks

Definition 1. RNN
Here is a new definition.

1.1.1 Gradient

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

$$D \triangleq \{x^{(i)} \in \mathbb{R}^p, y^{(i)} \in \mathbb{R}^q, i \in [1, N]\} \tag{1.1}$$

and the loss function $L_D : \mathbb{R}^U \to \mathbb{R}_{\geq 0}$ over D as

$$L_D(w) \triangleq \frac{1}{N}\sum_{i=1}^{N} L_i(w) \tag{1.2}$$

where $w \in \mathbb{R}^U$ represents all the weights of the net and L_i is an arbitrary loss function for the i^{th} example.

The network is composed of several layers as show in figure 1.1, each layer is composed of several neuron defined, as show in figure 1.2, by

$$a_l \triangleq \sum_j w_{lj}\phi_j \tag{1.3}$$

$$\phi_l \triangleq \sigma(a_l) \tag{1.4}$$

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 rule, as

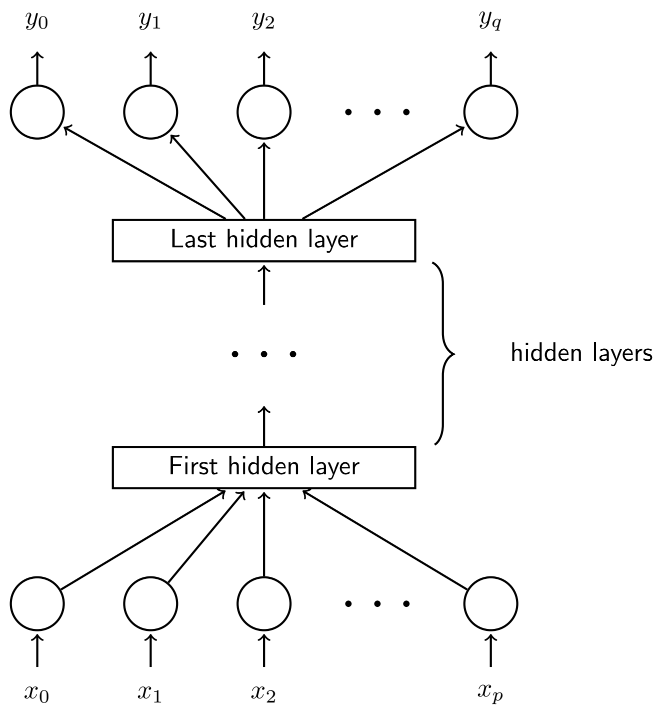

Figure 1.1: Feed forward neural network model

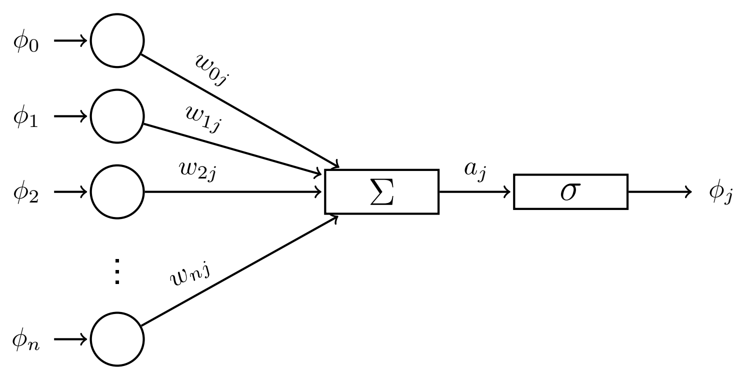

Figure 1.2: Neuron model

$$\frac{\partial L}{\partial w_{lj}} = \frac{\partial L}{\partial a_l} \cdot \frac{\partial a_l}{\partial w_{lj}} = \delta_l \cdot \phi_j$$

where we put

$$\delta_l \triangleq \frac{\partial L}{\partial a_l} \tag{1.5}$$

So we can easily compute $\delta_u = \frac{\partial L^{(i)}}{\partial a_u}$ for each output unit u once we choose a differentiable loss function; note that we don't need the weights for such a computation.

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

$$P(l) = \{k : \exists \text{ a link between } l \text{ and } k \text{ with weight } w_{lk}\} \tag{1.6}$$

Again, simply using the chain rule, we can write, for each non output unit l:

$$\delta_l = \sum_{k \in P(l)} \frac{\partial L^{(i)}}{\partial a_k} \cdot \frac{\partial a_k}{\partial a_l} = \sum_{k \in P(l)} \delta_k \cdot \frac{\partial a_k}{\partial \phi_l} \cdot \frac{\partial \phi_l}{\partial a_l} = \sum_{k \in P(l)} \delta_k \cdot w_{kl} \cdot \sigma'(a_l) \tag{1.7}$$

For output units instead we can compute $\delta_u = \frac{\partial L^{(i)}}{\partial a_u}$ directly once we define the loss function.

1.1.2 Backpropagation matrix notation

Here we rewrite the previously derived equations in matrix notation. Let us define the weight matrix for the i^{th} layer as the $p(i) \times p(i-1)$ matrix whose elements $w_{l,k}$ are the weights of the arcs which link neuron k from level $i-1$ to neuron l from level i, where $p(i)$ is the number of neuron layer i is composed of.

$$\boldsymbol{a}_{i+1} \triangleq W_{i+1} \cdot \boldsymbol{\phi}_i \tag{1.8}$$

$$\boldsymbol{\phi}_{i+1} \triangleq \sigma(\boldsymbol{a}_{i+1}) \tag{1.9}$$

$$\boldsymbol{\phi}_1 \triangleq \boldsymbol{x} \tag{1.10}$$

where $\sigma(\cdot)$ is the non-linear activation function and it's applied element by element. We can rewrite equation 1.7 in matrix notation as:

$$\frac{\partial L}{\partial W_i} = \frac{\partial L}{\partial \boldsymbol{a}_i} \cdot \frac{\partial \boldsymbol{a}_i}{\partial W_i}^T = \Delta_i \cdot \boldsymbol{\phi}_{i-1}^T \tag{1.11}$$

where

$$\Delta_i \triangleq \frac{\partial L}{\partial \boldsymbol{a}_i} \tag{1.12}$$

$$\Delta_i = W_{i+1}^T \cdot \Delta_{i+1} \circ \sigma(\Delta_i) \tag{1.13}$$

1.2 Recursive neural networks

Bibliography