Neural Networks
A simple example

Henrik Lund Mortensen

September 12, 2017

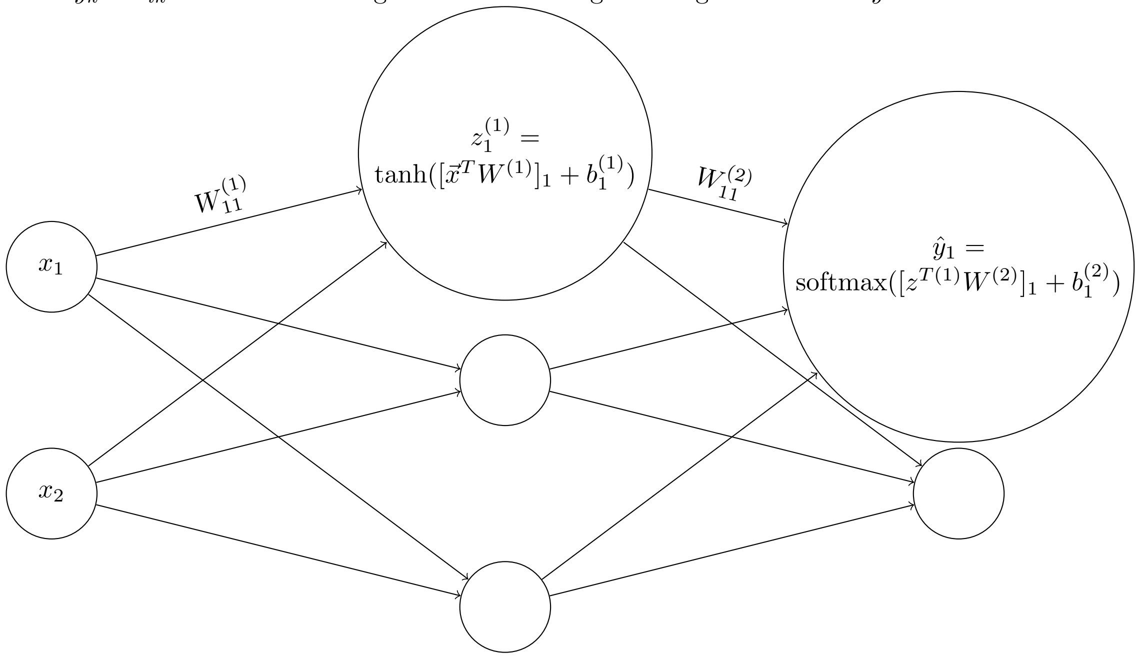
Neural network as a classifier

The neural network (NN) presented in this note solves a classification problem; Given $\vec{x}$ and a set of classes Y, determine which class $\vec{x}$ belongs to.

The NN is a nonlinear function with a lot of parameters, or weights $\{W, b\}$.

$$\hat{\vec{y}} = f_{NN}(\vec{x}, W, b) \tag{1}$$

The output is a vector, $\hat{\vec{y}}$, where the i'th entry is the estimated probability that the input belongs to the class $\vec{y}_k = \delta_{ik} \in Y$. The training consists of finding the weigths such that $\hat{\vec{y}}$.


x_1
x_2
$W_{11}^{(1)}$
$z_1^{(1)} = \tanh([\vec{x}^T W^{(1)}]_1 + b_1^{(1)})$
$W_{11}^{(2)}$
$\hat{y}_1 = \text{softmax}([z^{T(1)} W^{(2)}]_1 + b_1^{(2)})$

$$\text{Activation function} = \tanh(\vec{x}) \tag{2}$$

$$\text{softmax}(\vec{x})_i = \frac{e^{\vec{x}_i}}{\sum_k e^{\vec{x}_k}} \tag{3}$$

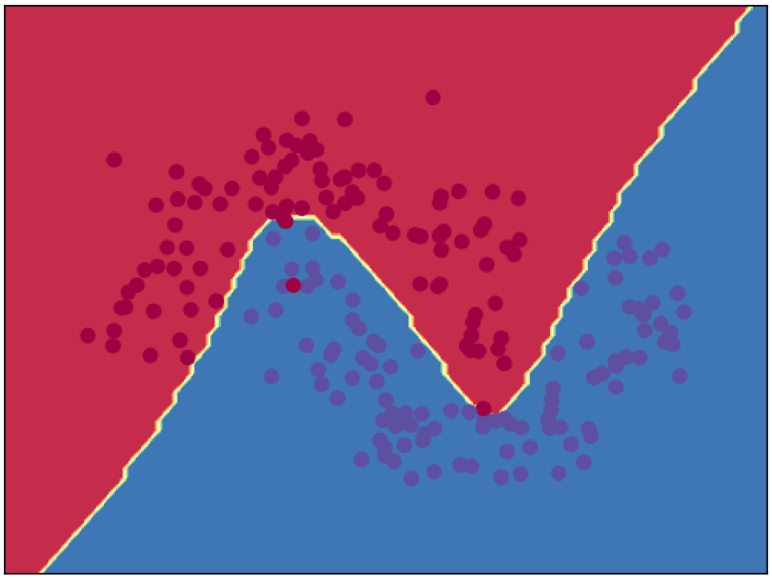
$$\text{Err}(\hat{\vec{y}}, \vec{y}) = \frac{-1}{N} \sum_{n \in N} \sum_{i \in C} \vec{y}_{n,i} \log \hat{\vec{y}}_{n,i} \tag{4}$$

Backpropagation

The weigths are updated by calculating the gradient of the loss function w.r.t. the individual weigths. These gradients can be efficiently calculated by using the backpropagation algorithm.

At the heart of the backpropagation algorithm is the chain rule,

$$\frac{\partial x}{\partial y} = \frac{\partial y}{\partial z} \frac{\partial z}{\partial x}. \tag{5}$$



Local gradients

$$\frac{\partial L}{\partial \hat{\vec{y}}_i} = \frac{-1}{N} \frac{\vec{y}_i}{\hat{\vec{y}}_i} \tag{6}$$

$$\frac{\partial \hat{\vec{y}}_i}{\partial \vec{a}_j^{(N_{\text{layer}})}} = \hat{\vec{y}}_i(\delta_{ij} - \hat{\vec{y}}_i) \tag{7}$$

so that (assuming $\vec{y}_i = \delta_{is}$ for some s)

$$\frac{\partial L}{\partial \hat{\vec{y}}} \cdot \frac{\partial \hat{\vec{y}}}{\partial \vec{a}^{(N_{\text{layer}})}} = \frac{\hat{\vec{y}} - \vec{y}}{N} \tag{8}$$

$$\frac{\partial \vec{a}_i^{(p)}}{\partial W_{jk}^{(p)}} = \begin{cases} \delta_{ik} \vec{z}_j^{(p-1)} & \text{if} p > 1 \\ \delta_{ik} \vec{x}_j & \text{if} p = 1 \end{cases} \tag{9}$$

$$\frac{\partial \vec{a}_i^{(p)}}{\partial b_j^{(p)}} = \delta_{ij} \tag{10}$$

$$\frac{\partial \vec{a}_i^{(p)}}{\partial \vec{z}_j^{(p-1)}} = W_{j,i}^{(p)} \tag{11}$$

The recursive formula for calculating the gradient of all weights and biases is given by

$$\frac{\partial L}{\partial \vec{b}^{(i)}} = \frac{\partial L}{\partial \vec{b}^{(i+1)}} \cdot W^{T(i+1)} \circ (1 - \vec{z^{(i)}} \circ \vec{z^{(i)}}) \tag{12}$$

$$\frac{\partial L}{\partial \vec{b}^{(N_{\text{layer}})}} = \frac{\hat{\vec{y}} - \vec{y}}{N_{\text{samples}}} \tag{13}$$

$$\frac{\partial L}{\partial \vec{W}^{(i)}} = \begin{cases} \vec{z}^{(i-1)} \otimes \frac{dL}{db^{(i)}} & \text{if } i \geq 2 \\ \vec{x} \otimes \frac{dL}{db^{(i)}} & \text{if } i = 1 \end{cases} \tag{14}$$