2. 3. 2. Neural Networks

(This section is based on Nielsen)

Neural networks are a class of machine learning methods with many variations. Some popular variants of this include convolutional neural networks, generally used for image analysis, recurrent neural networks for tasks such as speech recognition. However, for the problem presented in this work we use a multilayer perceptron model for regression, a type of deep neural networks.

Neural networks are structured in layers, the layers consist of input layer, one or more hidden layers, and an output layer. The hidden layers consist of “neurons”. In a feed forward neural network a neuron in one layer is connected to all neurons in the next layer. Each neuron has weights and biases that dictate their influence on the neurons in the next layer. The biases restrict the minimum weight needed for a neuron to influence the next layer. Information is processed through the different layers by going through each layer’s activation function. Activation functions are often non-linear functions like the commonly used sigmoid $1/(1+e^{\mathrm{x}})$ activation function. When we let the network complete the first forward pass, we get an output which usually will just be a random guess. For the network to start “learning” we use gradient decent on the loss function of the network. With this information we can adjust the weights and biases in the network through backward propagation. The process of feed forward and backward propagation is repeated under the assumption that we iteratively minimize the loss through each cycle.

The activation function in each layer takes a weighted sum (or linear combination) z_i^l of the neurons in the previous layer. The weighted sum is given by $z_i^{(0)} = \Sigma_j^M w_{ij}^{(0)} x_j + b_j^{(0)}$ in the input layer and $z_i^{(l)} = \Sigma_j^M w_{ij}^{(l)} a_j^{(l-1)} + b_j^{(l)}$ in layers after the input layer. Here (l) represents the layer, w represents the weights and b represents the biases and $a_j^{(l)} = f\left(z_j^{(l)}\right)$ where f is the activation function contributing to the weighted sum in the layers after. All neurons in one layer use the same activation function but from layer to layer there may be different activation functions.

Computing the activation function of the weighted sum for each subsequent layer is called the feed forward pass. After a feed forward pass, we use the result to get a correction from our optimizer (gradient decent) algorithm. With this we can use backprop to correct all the weights and biases throughout our network. Backprop starts in the output layer where the error is computed as:

$$\delta_j^{(L)} = f'\left(z_j^{(L)}\right)\frac{\partial \mathcal{L}}{\partial a_j^{(L)}}$$

Where $\mathcal{L}$ is the loss function. For regression we use the identity function as activation function, with the identity derivative being given by $f\left(z_j^{(L)}\right) = z_j^{(L)}$ and $f'\left(z_j^{(L)}\right) = 1$. With the mean square error (divided by 2) $\mathcal{L}\left(a_{\mathrm{i}}^{(L)}\right) = \frac{1}{2n}\Sigma_{i=1}^{n}(y_i - \hat{y_\imath})^2$ as the loss function this derivative should be $\frac{\partial \mathcal{L}\left(a_{\mathrm{i}}^{(L)}\right)}{\partial a_{\mathrm{i}}^{(L)}} = a_{\mathrm{i}}^{(L)} - y_i$. With this we see that the error for the output layer is then given by:

$$\delta_{\mathrm{i}}^{(L)} = a_{\mathrm{i}}^{(L)} - y_{\mathrm{i}}$$

With this output error we can go through each layer in the neural network backwards and compute the error and correction to the gradients. For the layers $\mathrm{l} = \text{L-1}, \text{L-2}, \ldots, 1$ we compute the corrections as:

$$\delta_j^{(l)} = \Sigma_k \delta_k^{(l+1)} w_{kj}^{(l+1)} f'\left(z_j^{(l)}\right)$$

And with these error corrections we update the weights according to our ADAM optimizer.

Kommentert [SF1]: Add short description of adavantages of ADAM

ADAMs parameter update is described by this set of equations:

$$m_w^{(t+1)} \leftarrow \beta_1 m_w^{(t)} + (1 - \beta_1)\nabla_w \mathcal{L}^{(\mathrm{t})}$$

$$\mathrm{v}_w^{(t+1)} \leftarrow \beta_2 \mathrm{v}_w^{(t)} + \left((1 - \beta_2)\nabla_w \mathcal{L}^{(\mathrm{t})}\right)^2$$

$$\widehat{m_w} = \frac{m_w^{(t+1)}}{1 + \beta_1^{(t+1)}}$$

$$\widehat{\mathrm{v}_w} = \frac{\mathrm{v}_w^{(t+1)}}{1 + \beta_2^{(t+1)}}$$

$$w^{(t+1)} \leftarrow w^{(t)} - \eta \frac{\widehat{m_w}}{\sqrt{\widehat{v_w}} + \epsilon}$$

Where β_1 and β_2 are decay factors, typically set to 0.9 and 0.999 respectively. For the other parameters (t) is the training iteration, η is the learning rate and ϵ is some small number to prevent zero division.

With this we know the most important algorithmic details to understand a feed forward neural network. Because of success in many machine learning problems and the popularity of deep learning they are widely used.