Introduction to Deep Learning

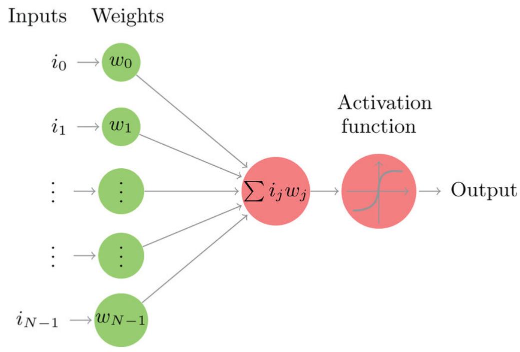
Himanshu Singh

Github

Perceptron

$$\hat{y} = g(w_o + X^T W)$$

Common activation Functions:



1) Sigmoid

$$g(z) = \frac{1}{1+e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

2) Hyperbolic Tangent

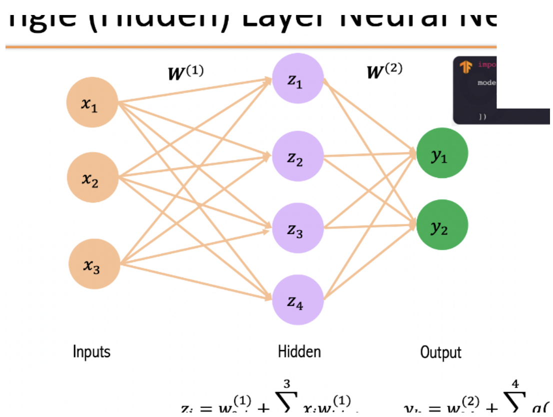
$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

3) Rectified Linear Unit (ReLU)

$$g(z) = max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$



MultiLayer Perceptron

$$z_j = \sum_{i=1}^{3} x_i w_{ij}^{(1)} + w_{0j}^{(1)}$$

$$y_k = \sum_{j=1}^{4} g(z_j) w_{jk}^{(2)} + w_{0k}^{(2)}$$

Emperical Loss

$$J(W) = \frac{1}{n} \sum_{i=1}^{n} L(\underbrace{f(x^{(i)}; W)}_{Predicted}, \underbrace{y^{(i)}}_{Actual})$$

Binary Cross Entropy Loss

$$J(W) = -\frac{1}{n} \sum_{i=1}^{n} \underbrace{y^{(i)}}_{Actual} \log(\underbrace{f(x^{(i)}; W)}_{Predicted}) + (1 - \underbrace{y^{(i)}}_{Actual}) \log(1 - \underbrace{f(x^{(i)}; W)}_{Predicted})$$

Mean Square Error Loss

$$J(W) = \frac{1}{n} \sum_{i=1}^{n} (\underbrace{f(x^{(i)}; W)}_{Predicted} - \underbrace{y^{(i)}}_{Actual})^2$$

Loss Optimization

$$W^* = \underset{W}{argmin} \frac{1}{n} \sum_{i=1}^{n} L(f(x^{(i)}; W), y^{(i)})$$

$$W^* = \underset{W}{argmin}\, J(W)$$

Gradient Descent Algorithm:

1. Initialize weight randomly $\sim N(0, \sigma^2)$
2. Loop until convergence
3. Compute gradient, $\frac{\partial J(w)}{\partial W}$
4. Update weights, $W \leftarrow w - \eta \frac{\partial J(w)}{\partial W}$
5. Return weights

#BACKPROPAGATION ALGORITHM

Recurrent Neural Network

Sequence Modelling Applications:

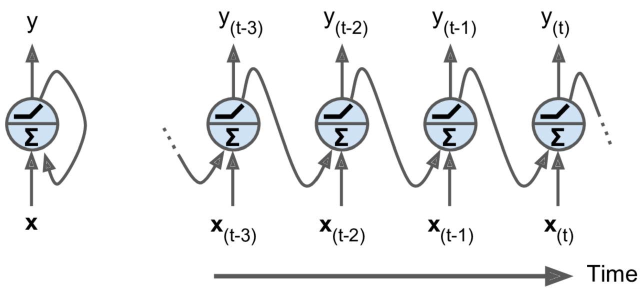
- One to One (Binary Classification)
- Many to One (Sentiment Classification)
- One to Many (Image Captioning)
- Many to Many (Machine Translation)

Design criteria for sequence Modelling:

- Handle variable-length sequences
- Track long-term dependencies
- Maintain information about order
- Share parameters across the sequence

Neurons with recurrence

$$\underbrace{\hat{y}_t}_{\text{Output}} = f(\underbrace{x_t}_{\text{input}}, \underbrace{h_{t-1}}_{\text{past memory}})$$



Recurrent Neural Networks

$$h_t = f_W(x_t, h_{t-1})$$

h_t: cell state

f_W: function with weights W

x_t: Input

h_{t-1}: previous state

$$h_t = \tanh(W_{hh}^T h_{t-1} + W_{xh}^T x_t)$$

$$\hat{y}_t = W_{hy}^T h_t$$

Reuse the same weight matrices at every time step

Encoding Language for Neural Network: Embedding (transform indexes into a vector of fixed size)

#Backpropagation Through Time

The problem of Long-Term dependencies: Vanishing gradients problem

Gating Mechanism in Neurons: Use gates to selectively add or remove information within each recurrent unit (LSTMs, GRUs)

Limitation of RNNs:

- Encoding Bottleneck
- Slow, no parallelization
- Not long Memory

Attention: Identify and attend to most important feature in the input (Query, Key, Value)

1. Encode positions information (Add positional embedding to tokens)
2. Extract Query, Key and Value for search
3. Compute attention weighting
4. Extract features with high attention

Attention is the foundational block of transformer.