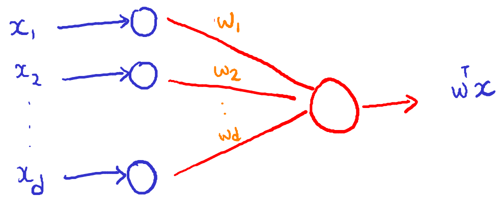
$$\min_{w} \underbrace{\sum_{i=1}^{n} L(w^T x_i, y_i)}_{\text{Loss}} + R(w)$$

$\hookrightarrow$ Regularizer

NEURAL NETWORKS

$x \in \mathbb{R}^d$ $\qquad$ $\text{sign}(w^T x)$

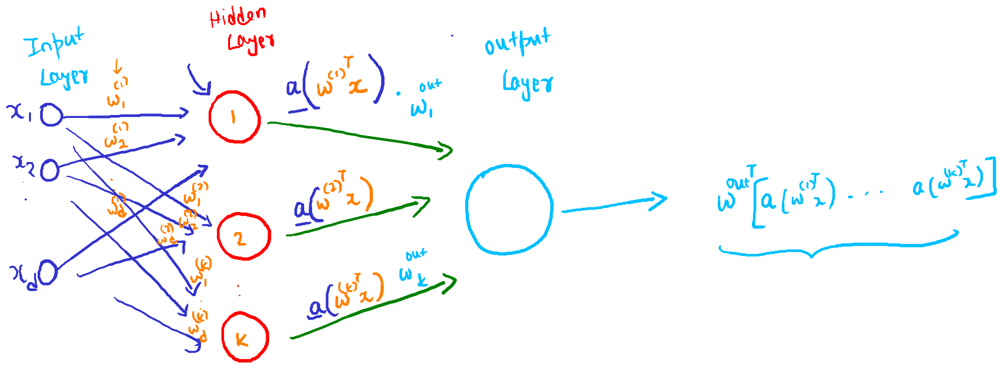
$[x_1, x_2 \cdots x_d]$



PARAMETERS

$\{ \underline{w^{(1)}}, \dots, \underline{w^{(k)}} \}$ $\quad w^{(i)} \in \mathbb{R}^d$

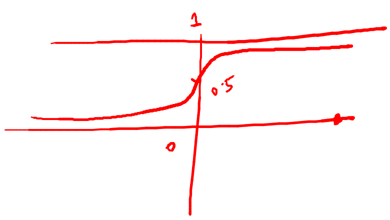
$\underline{w^{out}} \in \mathbb{R}^k$



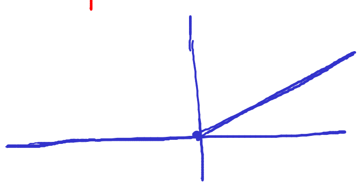
\# hidden layer nodes $\qquad$ activation function

$$\hat{y} = \underbrace{\sum_{i=1}^{k} w_i^{out}}\ \underbrace{a(w^{(i)^T} x)}$$

Examples of activation functions / non-linearities



- $a(z) = \frac{1}{1+e^{-z}}$ $\quad$ [SIGMOID]



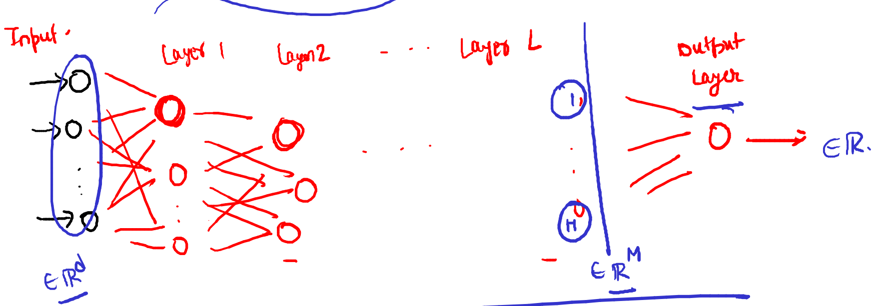
- $a(z) = \max(0, z)$ $\quad$ [Rectified Linear unit]

Regression

$$\mathcal{L}\big(NN(x_i,\underset{\{w^{(1)},\dots,w^{(L)},w^{out}\}}{\theta}),\ y_i\big)$$

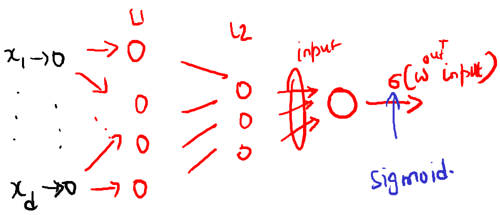
$$= \sum_{i=1}^{n} \big(\underbrace{NN(x_i;\theta)}_{= w^T x_i} - y_i\big)^2$$

- Learn θ^* using Gradient descent



- Gradient computed taking advantage of Chain rule → BACK-PROPAGATION

- Converges to local minima!



$P(y/x)$ $\quad$ y.

CROSS-ENTROPY LOSS.

Conclusion

CNN, RNN, LSTM, Attentions, Transformers,

- NN Learns local minima of non-concave functions
- Typically works very well in practice.
  ↳ especially for unstructured data.