Chapter 1

Activation Function

Activation functions are needed in order to increase the complexity of the NN, without them it would just be a linear sandwich of linear functions (multiplying linear functions = linear function)

1.1 Sigmoid

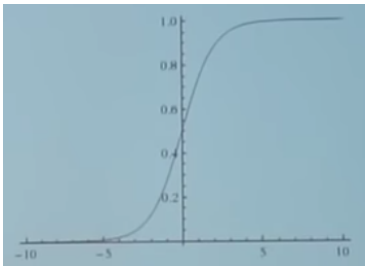

FIGURE 1.1: Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$

- Saturated neurons "kill" the gradient
- No zero-centered output
- exp() is computationaly expensive

1.2 Tanh

$\tanh(x) = \frac{2}{e^{-2x}+1} - 1$

- Zero centered output

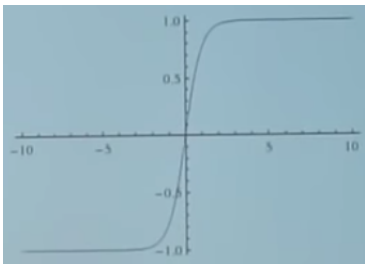

FIGURE 1.2: Tanh

- Saturated neurons "kill" the gradient

1.3 ReLU

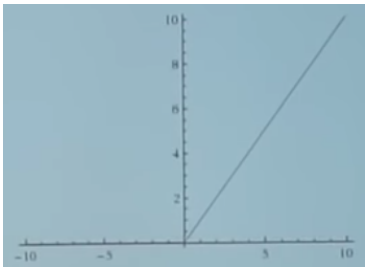

FIGURE 1.3: ReLU

$f(x) = \max(0, x)$ If a neuron has $x < 0$ the ReLU kills it in backprop. The gradient will be 0 so that neuron will not backprop downwards and its weights will not be updated. Its like a boolean gate. The gradient in the case of $x = 0$ is undefined (unlike case).

- Does not saturate
- Computationally efficient
- Converges mch faster than sigmoid/tanh in practive ($6x$)
- Not zero centred output
- If your unlucky a neuron may be never active because the initialization has put it outside the manifold.
- When the learning rate is high is easy to kill a lot of neurons. Imagine the activation function as a threshold which is moving while training. If the learning rate is to high it may move out of the data manifold. In that case the neuron dies and will never be able to recover because will never update again.

It is a good practice to initialize them with a slightly positive initial bias to avoid "dead neurons"

1.4 Leaky ReLU

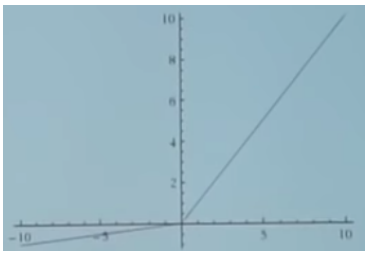

FIGURE 1.4: Leaky ReLU

$f(x) = \max(\alpha x, x) = \begin{cases} \alpha x & \text{if}(x \leq 0) \\ x & \text{if}(x > 0) \end{cases}$ where α is a very small number 0.01 which can be a hyperparameter or learned through .

- Does not saturate
- Computationally efficient
- Converges much faster than sigmoid/tanh in practice ($6x$)
- Does not die
- Not zero centred output
- Consistency of the benefits across tasks not clear

1.5 ELU - Exponential Linear Units

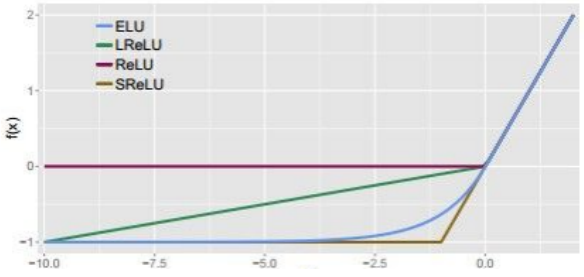

FIGURE 1.5: ELU

$f(x) = \begin{cases} \alpha(e^x - 1) & \text{if}(x \leq 0) \\ x & \text{if}(x > 0) \end{cases}$

- All benefits of ReLu
- Does not die
- Closer to zero mean outputs

- exp() is computationally expensive
- There is controversy if it actually trains better with this than with ReLU

1.6 Maxout

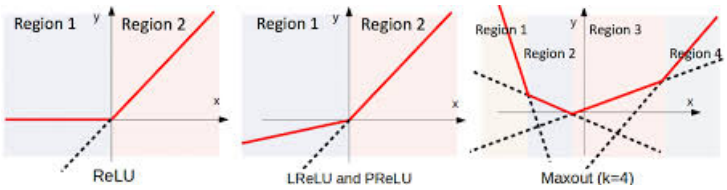

FIGURE 1.6: Maxout

$f(x) = \max(w_1^T x + b_1, w_2^T x + b_2)$

- Generalized ReLU and Leaky ReLU
- Does not saturate
- Does not die
- Linear Regime
- Doubles the number of parameters/neuros

Conclusion

- **Use ReLU** but be careful with your learning rates. So for example a 3-layer neural network will look like this:
- Try out Leaky ReLU/Maxout/ELU
- Try out tanh but don't expect much
- Don't use sigmoid