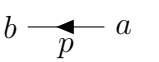
Appendix

A.1 QCD Feynman rules

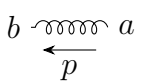
Here are reported the Feynman rules for QCD and in general a non-Abelian Gauge theory[**Peskin**]:

Fermion propagator:



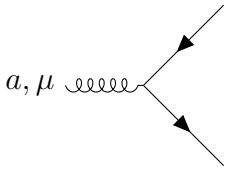
$$= \frac{i(\not{p}+m)}{p^2-m^2+i\epsilon}\delta^{a,b}$$

Gluon propagator:



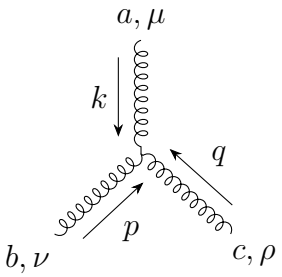
$$= \frac{-ig^{\mu\nu}}{p^2+i\epsilon}\delta^{a,b}$$

Fermion vertex:



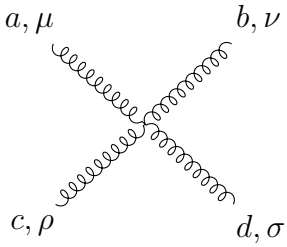
$$= ig\gamma^{\mu}T^{a}$$

3-boson vertex:

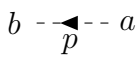


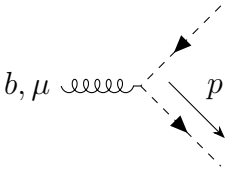
$$= \begin{aligned} & gf^{abc}[g^{\mu\nu}(k-p)^{\rho} \\ & \quad +g^{\nu\rho}(p-q)^{\mu} \\ & \quad +g^{\rho\mu}(q-k)^{\nu}] \end{aligned}$$

4-boson vertex:



$$= \begin{aligned} & -ig^2[f^{abc}f^{cde}(g^{\mu\rho}g^{\nu\sigma}-g^{\mu\sigma}g^{\nu\rho}) \\ & \quad +f^{ace}f^{bde}(g^{\mu\nu}g^{\rho\sigma}-g^{\mu\sigma}g^{\nu\rho}) \\ & \quad +f^{ade}f^{bce}(g^{\mu\nu}g^{\rho\sigma}-g^{\mu\rho}g^{\nu\sigma})] \end{aligned}$$

Ghost propagator:  $= \frac{-i\delta^{ab}}{p^2+i\epsilon}$

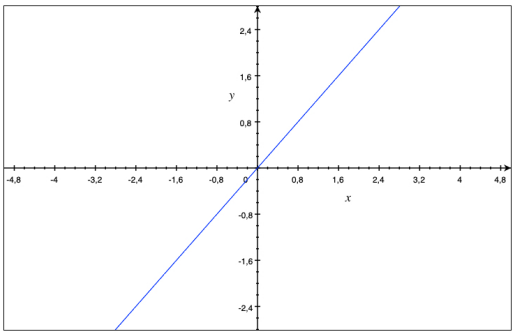
Ghost vertex:  $= -gf^{abc}p^{\mu}$

B.1 Activation functions

Here are reported the most commonly used activation functions for an artificial neuron in a neural network.

Linear function

Simple non-saturating linear function:

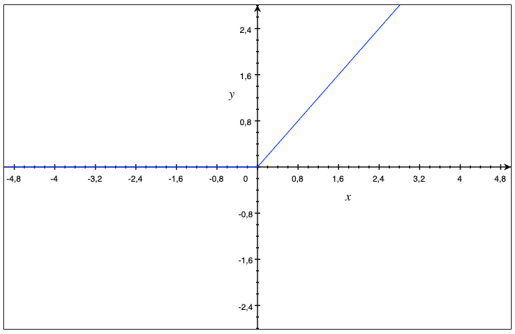


$$f(x) = x \tag{5.1}$$

Figure 5.1: Linear activation function

ReLU

Rectified Linear Unit (ReLU) introduces non linearities in the output. It is a non-saturating function with the disadvantage of having dead neurons, stopping the gradient flow, for negative inputs.

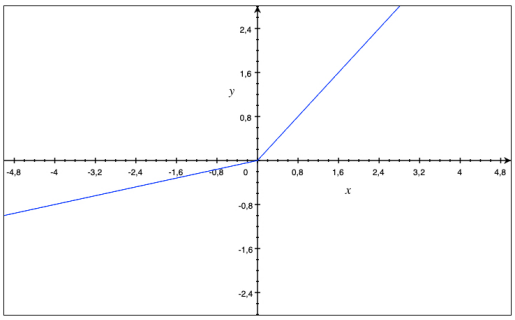


$$f(x) = max(0, x) \tag{5.2}$$

Figure 5.2: ReLU activation function

Leaky ReLU

Resolve the problem of dead neurons adding a small slope for negative inputs which ensure a gradient flow in this range.

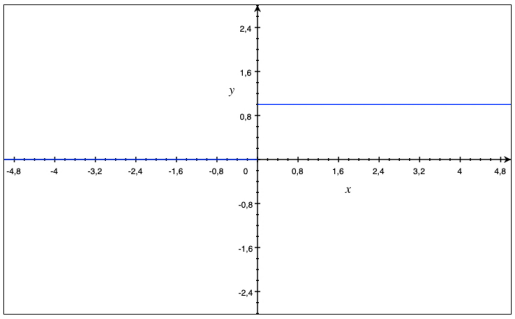


$$f(x) = \begin{cases} x & x > 0 \\ \alpha x & x < 0 \end{cases} \tag{5.3}$$

Figure 5.3: Leaky ReLU activation function

Binary

Standard limited step function where the output of the neuron is 0 or 1. In this case a small change in the weights can cause the output of the neuron to completely flip. That flip may then cause the behaviour of the rest of the network to change in an unpredicted way. Therefore, it is preferable to use a smooth activation function which imply small variations of the output for small variations of the weights.

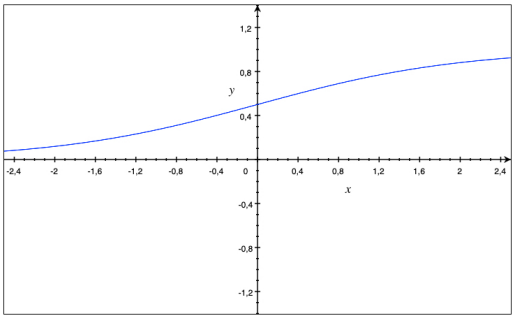


$$f(x) = \Theta(x) \tag{5.4}$$

Figure 5.4: Step activation function

Sigmoid

Smooth limited non-linear function which maps in the range $[0, 1]$. The drawbacks of this function are that the gradients near the tails are almost zero causing the problem of vanishing gradients in these regions and that the outputs are not zero-centered.

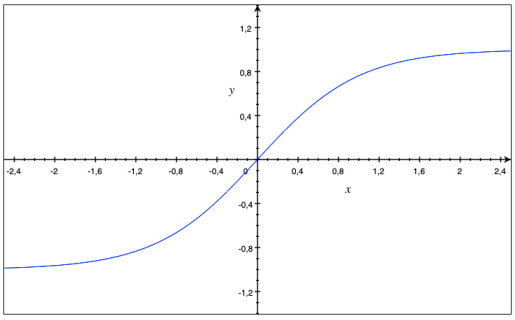


$$f(x) = \frac{1}{1 + e^{-x}} \tag{5.5}$$

Figure 5.5: Sigmoid activation function

Tanh

Smooth non-linear limited function which maps in the range $[-1, 1]$. Like the Sigmoid function vanishing gradients are possible in the tails but in this case the outputs are zero-centered.



$$f(x) = tanh(x) \tag{5.6}$$

Figure 5.6: Tanh activation function

Softmax

The Softmax activation functions is used in multinomial logistic regression in the last layer. Each output lies in the range $[0, 1]$ and it's normalized such that it sums to 1. This allow to interpret the output of a Softmax layer as a probability distribution.

$$\begin{pmatrix} 3 \\ 1 \\ 0.2 \end{pmatrix} \xrightarrow{\bar{f}(\bar{x})} \begin{pmatrix} 0.84 \\ 0.11 \\ 0.05 \end{pmatrix}$$

$$f(\bar{x})_i = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{5.7}$$

Figure 5.6: Softmax activation function