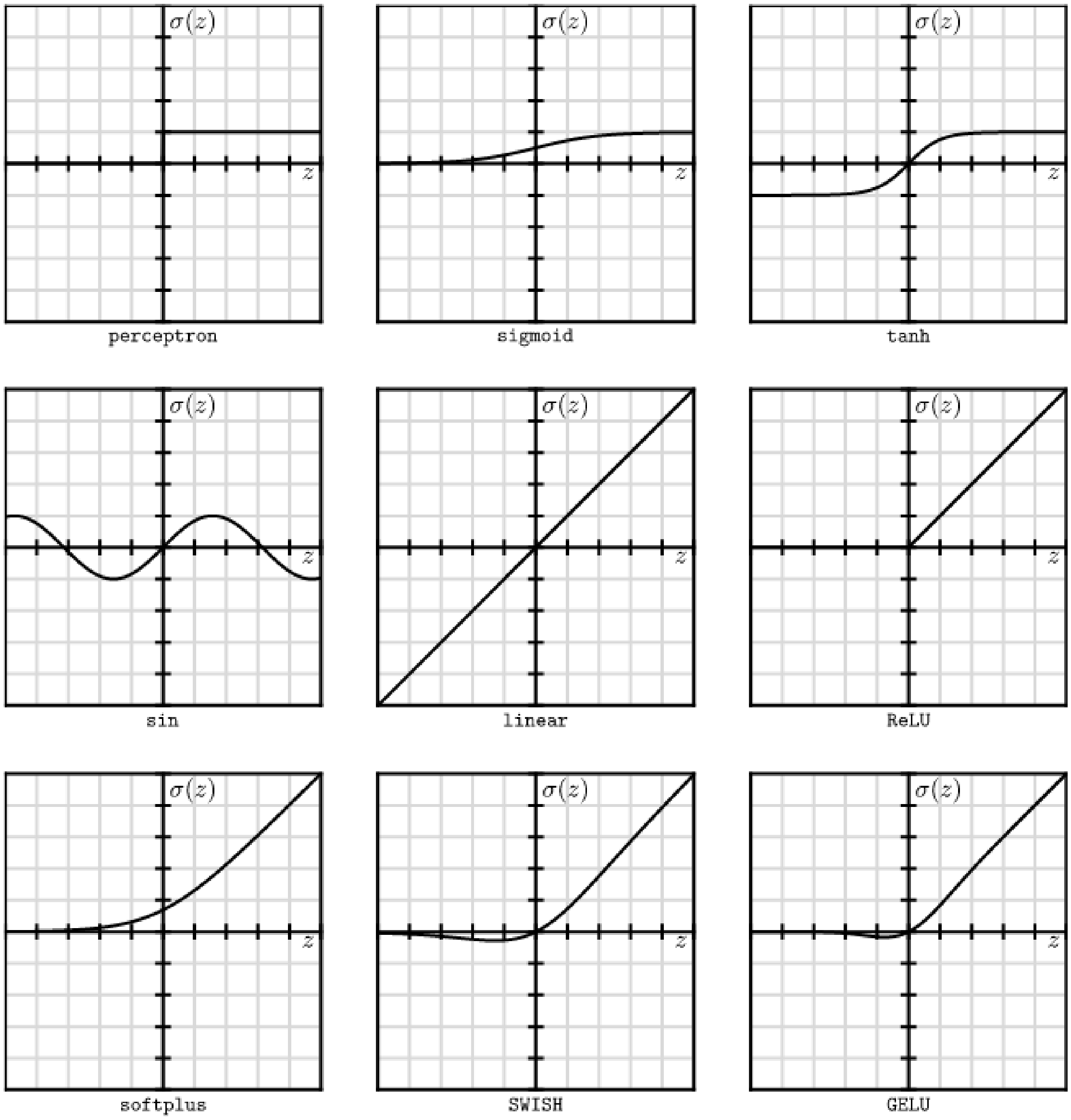
Neural Networks

Function approximation and activation functions

The authors introduce the basics of function approximation (what NNs do) and then cover some of the more common activation functions (perceptron, linear, RELU etc...). The big thing to take away from these sections is that neural networks aim to approximate general functions through learning weights in paramterised models, can be induced to learn particular functions by particular structures on the parameters, and that some activations functions (like RELU) are scale free (i.e $\sigma(kx) = k\sigma(x)$).



perceptron

sigmoid

tanh

sin

linear

ReLU

softplus

SWISH

GELU

Figure 2.2: Commonly-used activation functions $\sigma(z)$. Grids are in units of one for both the preactivation z and activation σ. (The `leaky ReLU` is not shown.)

Ensembles

By defining the initialisation regime of a neural network in terms of probability distributions we constrain the resulting ensemble of networks to have particular properties depending on the probability distributions chosen. In particular Gaussian initialistion of the weights and biases are easy to work with in practice and in theory.

$$\begin{aligned} p\left(b_i^{(\ell)}\right) &= \frac{1}{\sqrt{2\pi C_b^{(\ell)}}} \exp\left[-\frac{1}{2C_b^{(\ell)}}\left(b_i^{(\ell)}\right)^2\right], \\ p\left(W_{ij}^{(\ell)}\right) &= \sqrt{\frac{n_{\ell-1}}{2\pi C_W^{(\ell)}}} \exp\left[-\frac{n_{\ell-1}}{2C_W^{(\ell)}}\left(W_{ij}^{(\ell)}\right)^2\right]. \end{aligned}$$

By theoretically analysing the resulting outputs from NN's initialised with weights and biases sampled from these distributions, they can give recommendations for the setting of the hyperparameter variances $C_b^{(l)}$ and $C_W^{(l)}$.

These randomly sampled W&Bs induce a probability distribution over the outputs $z^{(L)}$ of the NN.

$$p\left(z^{(L)}\middle|\mathcal{D}\right) = \int \left[\prod_{\mu=1}^{P} d\theta_\mu\right] p\left(z^{(L)}\middle|\theta, \mathcal{D}\right) p(\theta).$$

The function $p(z^{(L)}|\theta, \mathcal{D})$ is actually deterministic, since it is defined by the NN which we know how to evaluate if we know the weights. This results in putting down a dirac delta function on the outputs of each layer:

$$\begin{aligned} p\left(z^{(1)}\middle|\mathcal{D}\right) = \int &\left[\prod_{i=1}^{n_1} db_i^{(1)}\; p\left(b_i^{(1)}\right)\right]\left[\prod_{i=1}^{n_1}\prod_{j=1}^{n_0} dW_{ij}^{(1)}\; p\left(W_{ij}^{(1)}\right)\right] \\ &\times \left[\prod_{i=1}^{n_1}\prod_{\alpha\in\mathcal{D}} \delta\left(z_{i;\alpha}^{(1)} - b_i^{(1)} - \sum_{j=1}^{n_0} W_{ij}^{(1)} x_{j;\alpha}\right)\right]. \end{aligned}$$

$$p\Big(z^{(\ell+1)}\Big|z^{(\ell)}\Big) = \int \left[\prod_{i=1}^{n_{\ell+1}} db_i^{(\ell+1)}\ p\Big(b_i^{(\ell+1)}\Big)\right] \left[\prod_{i=1}^{n_{\ell+1}} \prod_{j=1}^{n_\ell} dW_{ij}^{(\ell+1)}\ p\Big(W_{ij}^{(\ell+1)}\Big)\right]$$

$$\times \left[\prod_{i=1}^{n_{\ell+1}} \prod_{\alpha\in\mathcal{D}} \delta\left(z_{i;\alpha}^{(\ell+1)} - b_i^{(\ell+1)} - \sum_{j=1}^{n_\ell} W_{ij}^{(\ell+1)} \sigma\Big(z_{j;\alpha}^{(\ell)}\Big)\right)\right],$$

$$p\Big(z^{\text{out}}\Big|\mathcal{D}\Big) = \int \left[\prod_{\mu=1}^{P} d\theta_\mu\right] p(\theta) \left[\prod_{i=1}^{n_{\text{out}}} \prod_{\alpha\in\mathcal{D}} \delta\Big(z_{i;\alpha}^{\text{out}} - f_i(x_\alpha;\theta)\Big)\right].$$