Focus on L_k Regularization

$$X = (x_1, \dots, x_n), \text{ for } \begin{cases} q \in \mathbb{N}^*, & \|x\|_q = \sqrt[q]{\sum |x_i|^q} \\ q = 0, & \|x\|_0 = \sum \underbrace{\mathbb{1}(x_i \neq 0)} \end{cases}$$

INDICATOR FUNCTION : $\mathbb{1}_{x_i} \begin{cases} 0, x_i = 0 \\ 1, x_i \neq 0 \end{cases}$

I - Focus on L_0 Regularization

L_0 Regularization refers to minimizing the number of nonzero weights in a model

→ goal : remove unnecessary parameters and make the network sparse (sparse model = ↗ interpretable ↗ efficient)

→ effect : pruning = forces some weights to be exactly zero

⇒ Because of its discontinuous nature : Standard Gradient Descent for Regularization UNUSABLE : X $\underbrace{\mathcal{L}}_{\text{total loss}} = \underbrace{Loss(W)}_{\text{regular loss from our NN (like MSE)}} + \underbrace{\lambda \|W\|_0}_{L_0 \text{ penalty}}$

⇒ We need a continuous approximation

→ We use the Hard Concrete Distribution : (relaxed version of discrete 0/1 values)

1. Instead of directly pruning weights (0 or 1), we assign a probability that a weight is active.
2. We sample values from a stretched sigmoid function ($\in [0;1]$) to approximate binary decisions.
3. This allows gradient-based learning, making L_0 Regularization trainable.

- Probabilistic Relaxation : a mathematical trick used to approximate non-differentiable discrete decisions (e.g., selecting whether a weight should be active or zero) with continuous and differentiable approximations.

- Why Concrete in the Hard Concrete distribution ?
  - The Concrete Distribution (or Relaxed Bernoulli Distribution) is a continuous approximation of a discrete Bernoulli (0/1) Distribution.

- Why Hard in the Hard Concrete Distribution ? (mask output nearly binary (0/1))
  - Even though the function is continuous, the output strongly resembles a binary decision
  ⇒ The relaxation still maintains a "hard" selection because most values get pushed close to 0 or 1.

The Hard Concrete Distribution modifies the Concrete distribution by adding additional stretching or clipping.

Mathematical Trick : Hard Concrete Distribution

$$z = \sigma\left(\frac{\log(u) - \log(1-u) + \log(\alpha)}{\beta}\right)(\xi - \gamma) + \gamma$$

- $u \sim U(0,1)$, a sample from a Uniform (0,1) Distribution

→ introduces stochasticity, allowing the model to explore different sparsity patterns
→ this randomness ensures that weight selection is a smooth process rather than an abrupt, discrete change

- $\log(u) - \log(1-u)$, (Logistic Noise), a transformation of u that maps it to a range from $-\infty$ to $+\infty$

→ transforms Uniform D into a Logistic D, which helps in approximating binary choices in a smooth way
→ ensures that small differences in u leads to smooth changes in the final output z, making the function differentiable

- $\log(\alpha)$, a trainable parameter controlling how likely a weight is to be zeroed out (sparsity)

→ allows the model to learn which connections to remove dynamically during training
→ a large $\alpha \Rightarrow$ the weight is more likely to be pruned, a small $\alpha \Rightarrow$ the weight is more likely to stay

- β → Temperature Parameter (Softening the Decision), a scaling factor that controls the sharpness of a transition

→ controls how smooth or sharp the transition is from keeping a weight to pruning it
→ a large $\beta \Rightarrow$ smoother transitions (gradually learning sparsity), small $\beta \Rightarrow$ harder thresholding (closer to pure 1/0)

- $\sigma(\cdot)$ → Sigmoid Activation : $x \mapsto \frac{1}{1+e^{-x}}$

→ smoothly maps the transformed value into the range [0;1]
→ ensures that the function is differentiable so gradient-based optimization works

- $(\xi - \gamma) + \gamma$ → Stretching & Shifting, scales and shift the sigmoid output

→ ensures that the mask z can be in the range $[\gamma;\xi]$ instead of just [0;1]
→ γ (negative shift) : prevent weights from being too close to zero during training
→ ξ (positive stretch) : ensures that the function can reach exactly 1 when needed

Clamping z between 0 and 1 : $z = \text{clamp}(z, 0, 1)$

→ a final step to force z to stay between 0 and 1
→ even with smooth transformations, z could sometimes exceed these limits, so we clip it
→ ensures z behaves like a proper probability mask

- Mask : a matrix M with values typically between [0;1] or {0,1}, applied element-wise to another tensor X : $X' = M \odot X$

- $M_{ij} = 1$: the corresponding element X_{ij} is kept
- $M_{ij} = 0$: the corresponding element X_{ij} is zeroed out (ignored)
- $0 < M_{ij} < 1$: represents a probabilistic mask allowing gradient-based learning