Límites Teóricos del Aprendizaje

Análisis de la Cota Superior VC Clásico

Considera la cota superior clásica derivada de la dimensión VC con probabilidad al menos $1 - \delta$:

$$E_{out}(h^*) \leq E_{in}(h^*) + \sqrt{\frac{d(log(2m/d) + 1) - log(\delta/d)}{m}}$$

donde:

- d es la dimensión VC.
- m es el tamaño del dataset.

Clasificador lineal en $\mathbb{R}^{10} \implies d = 11$

Dimensión VC teórica

Aquí aprovechando que es un clasificador lineal, la dimension vc es el número de parámetros más uno, lo cual da que $d = 11$

Cota superior para $m \in \{100, 1000, 10000\}$

In [35]:
```
import numpy as np

def upper_bound(d: int, m: int, delta: float = 0.05) -> float:
    term = (d * (np.log(2*m/d) + 1) - np.log(delta/d)) / m
    if term < 0:
        print(f"Warning: término negativo dentro de la raíz: {term}. Cota no
        return np.nan
    return np.sqrt(term)

# Ahora calculamos las cotas superiores.
M = [100, 1000, 10000]

for m in M:
    print(upper_bound(d=11, m=m))
```

```
0.6949695718346142
0.27134241577853835
0.0994761994767933
```

Con lo cual tenemos que con al menos una probabilidad del 95%:

- Para $m = 100$:

$E_{out}(h^*) \leq E_{in}(h^*) + 0.695$

- Para $m = 1000$:
  $E_{out}(h^*) \leq E_{in}(h^*) + 0.2712$
- Para $m = 10000$:
  $E_{out}(h^*) \leq E_{in}(h^*) + 0.0995$

Clasificador polinomial de grado en 3 en $\mathbb{R}^5$

Investigando se encontró la siguiente formula:

$$d = C_g^{g+e}$$

donde:

- g es el grado del polinomio.
- e es la dimensión del espacio

Para nuestro caso particular $g = 3$ y $e = 5$, entonces remplazando:

$$d = C_3^{5+3} = 56$$

In [36]:
```
for m in M:
    print(upper_bound(d=56, m=m))
```

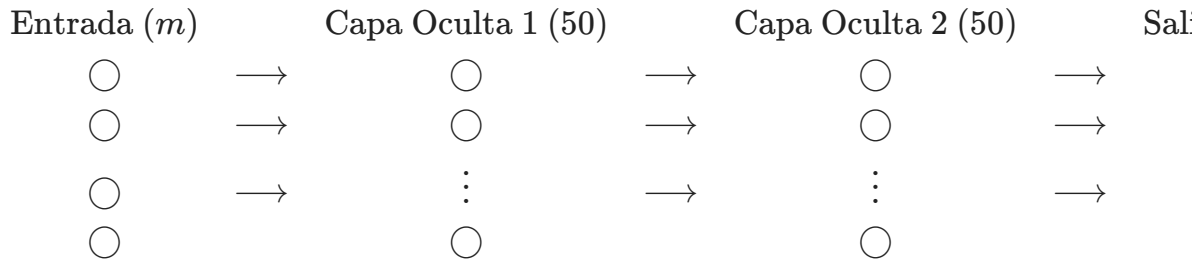
```
1.158909667790423
0.5130808191868775
0.19803956479076193
```

Con lo cual tenemos que con al menos una probabilidad del 95%:

- Para $m = 100$:
  $E_{out}(h^*) \leq E_{in}(h^*) + 1.159$
- Para $m = 1000$:
  $E_{out}(h^*) \leq E_{in}(h^*) + 0.513$
- Para $m = 10000$:
  $E_{out}(h^*) \leq E_{in}(h^*) + 0.198$

Red Neuronal con dos Capas Ocultas con 50 Neuronas cada Una.

[



]

Calculamos los pesos entre capas junto con sus sesgos.

- Entre la capa de entrada y la primera capa oculta:

$$P = m * 50 + 50 = 50 * (m + 1)$$

- Entre las dos capas ocultas:

$$P = 50 * 50 + 50 = 50 * (50 + 1)$$

- Entre la última capa oculta y la salida:

$$P = 50 + 1$$

Juntando todo nos queda:

$$P = 50 * (m + 1) + 50 * (50 + 1) + 50 + 1 = 50(m + 1 + 50 + 1) + 1 = 2651 + 5$$

donde P es el total de parámetros.

Según la bibliografía la dimension vc de una red neuronal densa esta acotada por

$$d = O(P\,log(P))$$

Remplazando:

$$d = O(2651 + 50 * m\,log(2651 + 50 * m))$$

Y agarrando la peor cota nos quedamos con que

$$d = (2651 + 50 * m)\,log(2651 + 50 * m)$$

In [37]:
```
for m in M:
    d = round((2651 + (50 * m)) * np.log10(2651 + (50 * m)))
    print(upper_bound(d=d, m=m))
```

```
Warning: término negativo dentro de la raíz: -1188.740896425024. Cota no válida.
nan
Warning: término negativo dentro de la raíz: -950.2446084167344. Cota no válida.
nan
Warning: término negativo dentro de la raíz: -1136.2245016825298. Cota no válida.
nan
```

Para las redes neuronales debido a que tienen una dimension vc altísima se hace díficil calcular la dimensión vc, vamos a aumentar el número de datos a ver si funciona mejor. Usemos la regla de 10 veces la dimensión vc

In [38]:
```
for i in [10, 100, 1000]:
    d = round((2651 + (50 * m)) * np.log10(2651 + (50 * m)))
    print(upper_bound(d=d, m=i * d))
```

```
0.6321185416681602
0.2509646110550422
0.0927410841709051
```

Vemos que para esta cantidad de datos en función de la dimension vc es:

- Para $m = 10 * d$:
  $E_{out}(h^*) \leq E_{in}(h^*) + 0.632$
- Para $m = 100 * d$:
  $E_{out}(h^*) \leq E_{in}(h^*) + 0.251$
- Para $m = 1000 * d$:
  $E_{out}(h^*) \leq E_{in}(h^*) + 0.093$