Aprendizaje Automático Profundo (Deep Learning)

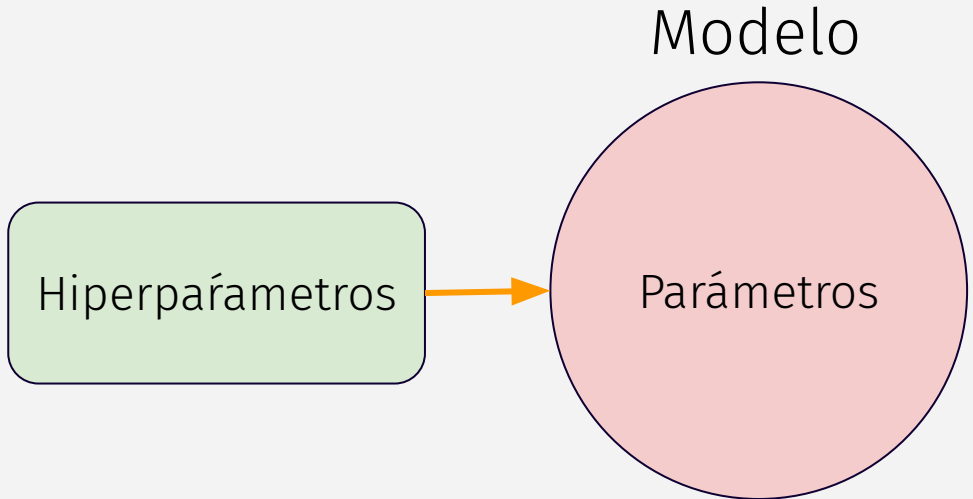
Dr. Facundo Quiroga - Dr. Franco Ronchetti

Optimización de hiperparámetros

Parámetros vs Hiperparámetros

- Hiperparámetros: están “afuera” del modelo
  - Cantidad de capas
  - Tipo de capas
  - Optimizador
    - Tasa de aprendizaje
    - Regularización
    - Función de error
  - Cantidad de épocas
  - Data augmentation
  - Inicialización de parámetros
- Se optimizan “a mano”

- Parámetros: están “dentro” del modelo
  - Pesos de filtros convolucionales
  - Sesgos (bias)
  - Pesos de capas lineales
- Se optimizan mediante descenso de gradiente



Fuerza bruta

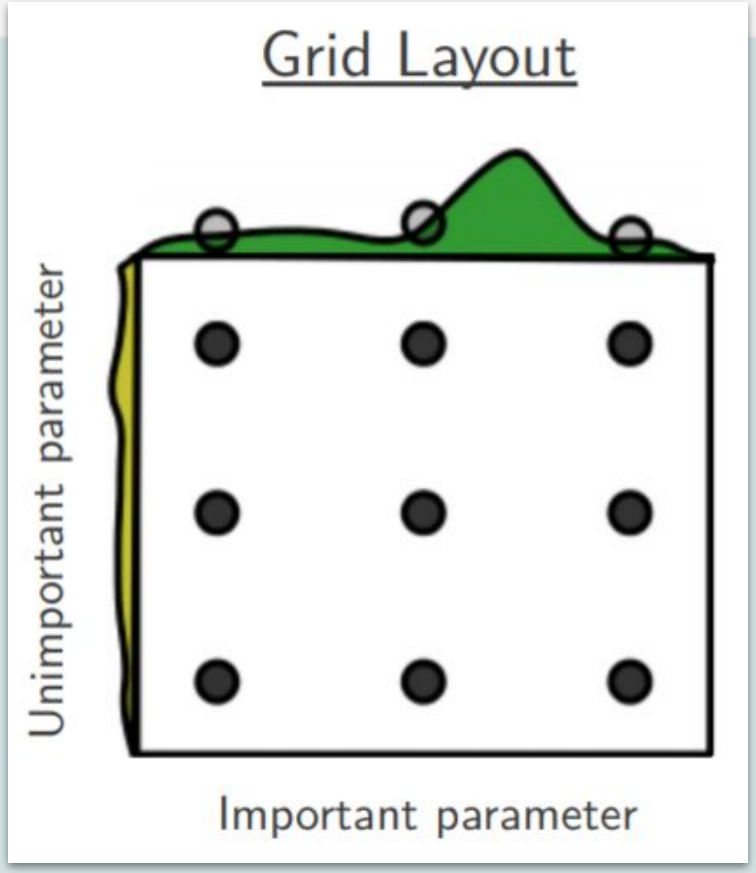
- Fuerza bruta: probar todas las posibilidades
  - Explosión combinatoria
  - No es práctico con muchos parámetros/valores

```
mejor_modelo=None
for epocas in range(500):
   for capas in range(500):
      for tipo_capa in [“conv3x3”,”conv5x5”,...”lineal”]:
         … (más for-loops con más hiperparámetros)
         modelo=entrenar_modelo(epocas,capas,tipo_capa)
         if mejor(modelo, mejor_modelo):
            mejor_modelo = modelo
```

Búsqueda en Grilla

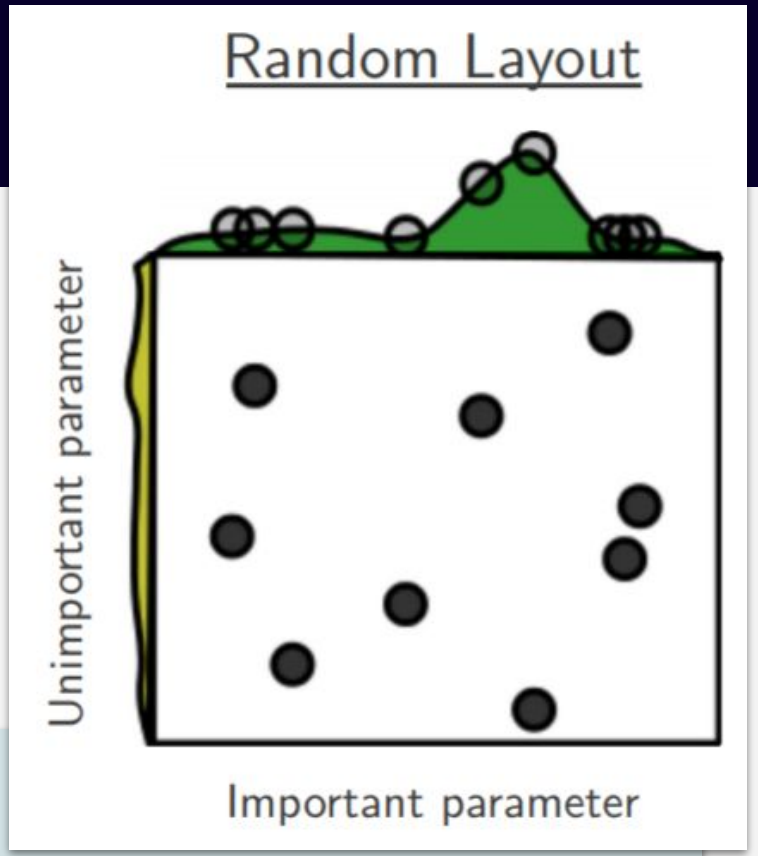
- Búsqueda en grilla
  - como fuerza bruta
  - pero presupuesto finito de K combinaciones

```
mejor_modelo=None
for epocas in range(500,step=100):
   for capas in range(500,step=100):
      … (más fors con más hiperparámetros)
         modelo,score=entrenar_modelo(epocas
         print(modelo,score)
         if mejor(modelo, mejor_modelo):
            mejor_modelo = modelo
```



Búsqueda Aleatoria

- Búsqueda aleatoria:
  - elegir K combinaciones aleatorias de hiperparámetros
- Casi siempre mejor que Grilla o Fuerza Bruta




```
mejor_modelo=None
for i in range(K):
   epocas, capas, ... = hiperparametros_aleatorios()
   modelo, score = entrenar_modelo(epocas, capas, ...)
   print(modelo, score)
   if mejor(modelo, mejor_modelo):
      mejor_modelo = modelo
```

Optimizar hiperparámetros con menos datos

```
x, y = load_samples()
x_subset,y_subset = make_subset(x,y,0.1) #10%
best_model = hiperparameters_search(x_subset,y_subset)
best_model.fit(x,y)
```