

FeedForward Neural Networks e introducción a Las Librerías de Deep Learning

Física Computacional 2
Ph.D. Santiago Echeverri Arteaga

Feedforward neural network

- Una sola neurona no es suficiente para modelar un sistema arbitrario ni da fronteras de clasificación más allá de lo que podría hacer una regresión logística
- El modelo de feedforward neural network consiste en capas totalmente conectadas entre sí con pesos W
- Se usa gradient descent (mini batch) para entrenarlas
- En cada nodo se hace una combinación lineal de los tensores que ingresan y a la salida se aplica una función activadora.
- La predicción se le suele llamar Forward propagation y al entrenamiento Backpropagation

Estructura	Regiones de Desición	Problema de la XOR	Clases con Regiones Mezcladas	Formas de Regiones más Generales
1 Capa	Medio Plano Limitado por un Hiperplano			
2 Capas	Regiones Cerradas o Convexas			
3 Capas	Complejidad Arbitraria Limitada por el Número de Neuronas			

Backpropagation

- Considere un perceptrón \mathcal{N} de L capas con función de activación a y mapeos afines $z_i(x) = w_i x + b_i$ para $i = 1, \dots, L$. Dado un conjunto de datos de entrenamiento $\{(x_i, y_i)\}_{i=1}^n$ donde $x_i \in \mathbb{R}^p$ y $y_i \in \mathbb{R}^q$. Para cada $i \in \{1, \dots, n\}$ se define la función de pérdida

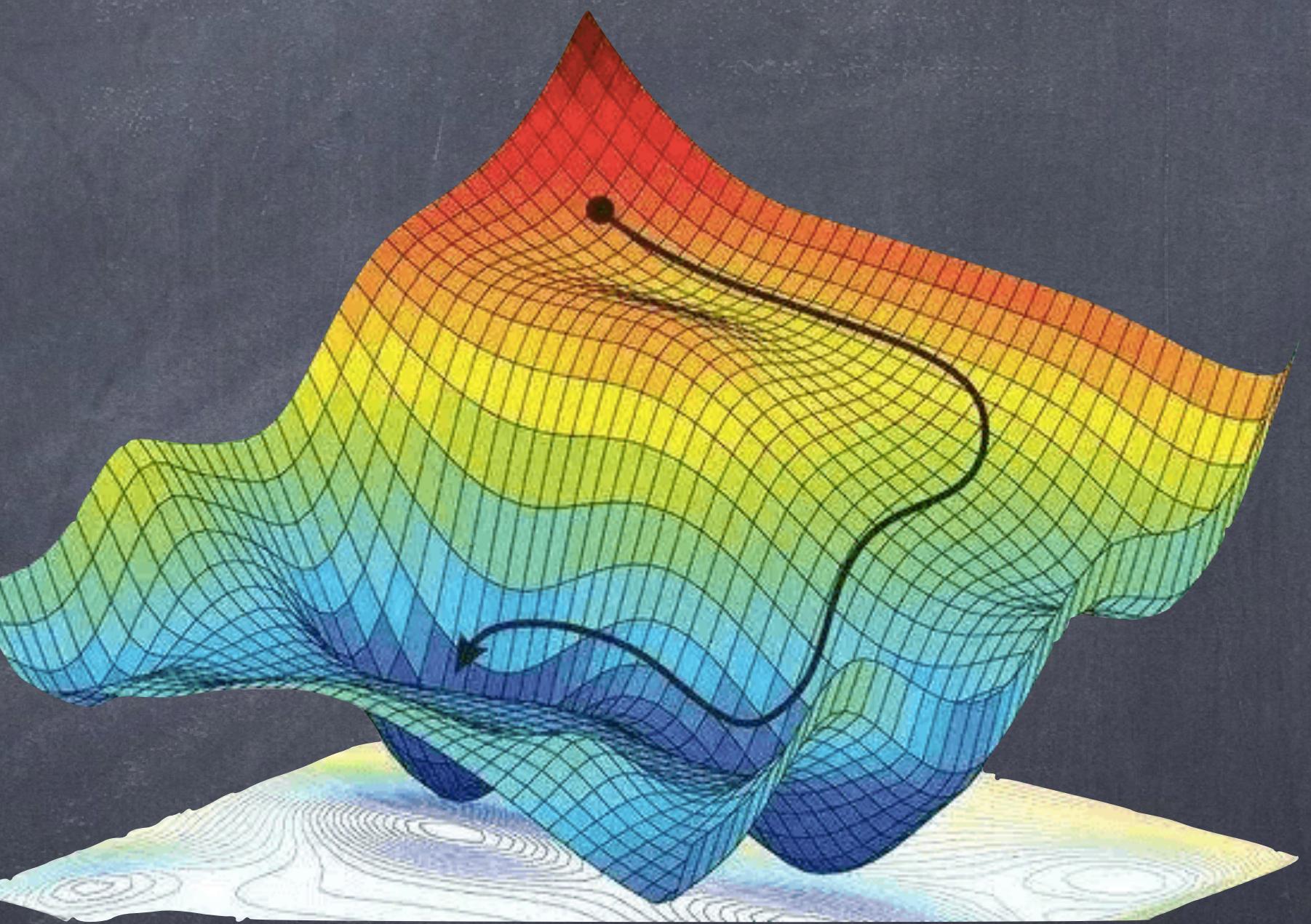
$$\mathcal{L}(\mathcal{N}, \{(x_i, y_i)_{i=1}^n\}) = -\frac{1}{n} \sum_{i=1}^n J(y_i, \mathcal{N}(x_i))$$
 con J una función que penaliza

la discrepancia entre un dato observado y_i y el valor $\mathcal{N}(x_i)$ predicho por el modelo para el input x_i . El algoritmo de backpropagation consiste en minimizar esa función ajustando los coeficientes de mapeo afín

Gradient Descent

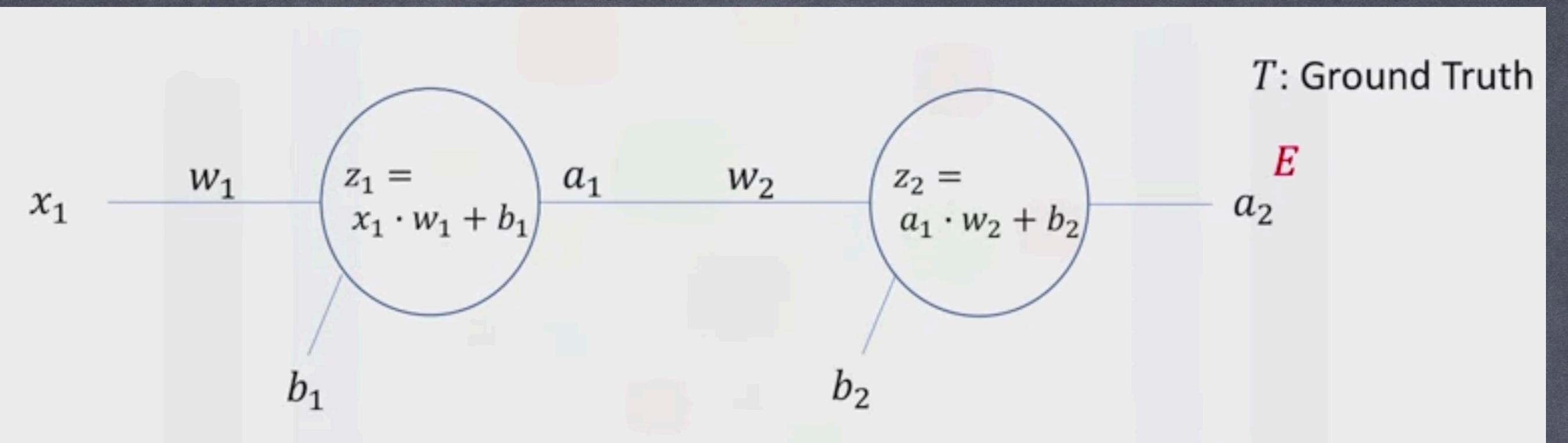
- Iniciar con una función de costo $J(\beta)$
- Calcular el gradiente del costo respecto a los parámetros.
- Actualizar los parámetros como $-\eta \nabla J(\beta)$ donde η es un hiperparámetro que determina el tamaño del paso. Se le conoce el Learning Rate.
- Hay variantes que toman menos puntos como el Mini-Batch y el Stochastic Gradient Descent
- Ejemplo en regresión lineal:

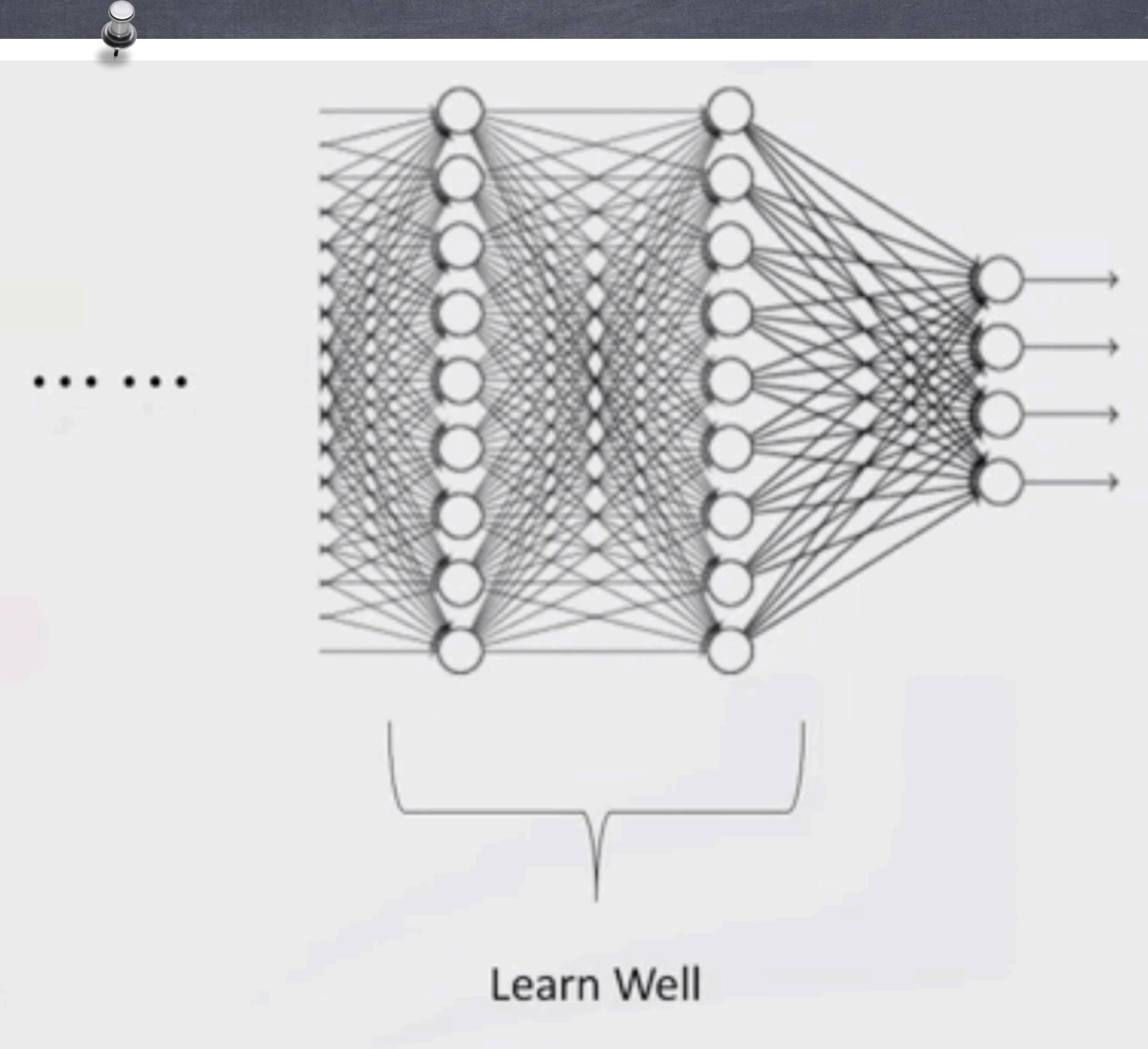
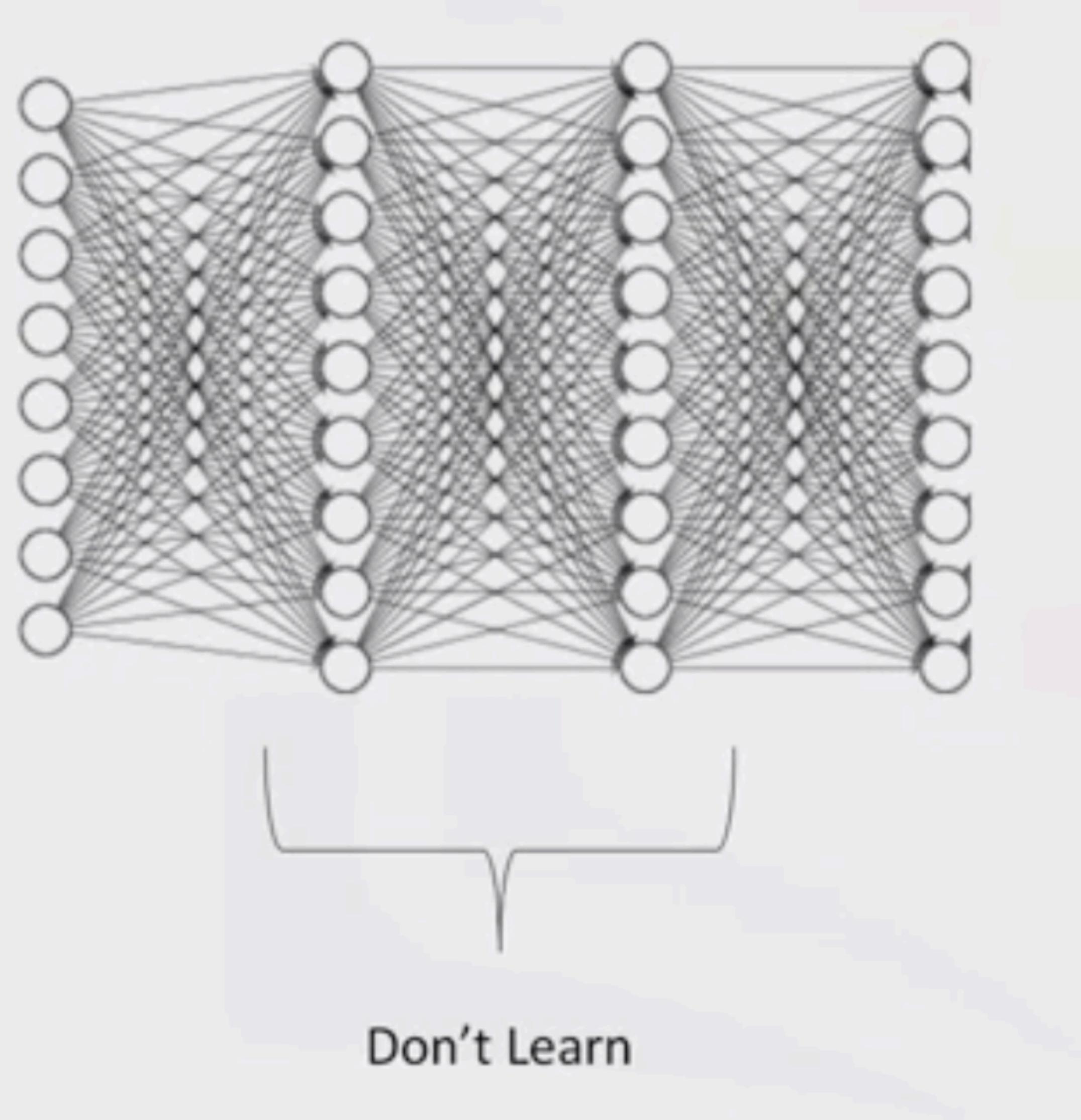
$$\omega_1 = \omega_0 - \eta \nabla \frac{1}{2} \sum_{i=1}^N \left((\beta_0 + \beta_1 x^{(i)}) - y^{(i)} \right)^2$$



Gradient Descent

- $w_2 \rightarrow w_2 - \eta \frac{\partial J}{\partial w_2}$
- Donde se sabe que $J(\omega) = \frac{1}{2}(T - a_2)^2$, $a_2 = f(z_2)$, $z_2 = a_1 w_2 + b_2$
- Para actualizar w_2 se calcula $\frac{\partial J}{\partial w_2} = \frac{\partial J}{\partial a_2} \frac{\partial a_2}{\partial z_2} \frac{\partial z_2}{\partial w_2}$; que tomando una función de activación sigmoide $\frac{\partial J}{\partial w_2} = (- (T - a_2))(a_2(1 - a_2)a_1)$
- Para actualizar b_2 es análogo a w_2 pero para w_1 se hace: $\frac{\partial J}{\partial w_1} = \frac{\partial J}{\partial a_2} \frac{\partial a_2}{\partial z_2} \frac{\partial z_2}{\partial a_1} \frac{\partial a_1}{\partial z_1} \frac{\partial z_1}{\partial w_1}$





Gradient descent

- Stochastic: No se hace la sumatoria sobre todos los puntos sino que se calcula en un solo valor tomado aleatoriamente
- Mini-Batch: No se hace la sumatoria sobre todos los puntos sino sobre un conjunto de tamaño reducido.
- Número de epochs: Número de veces que se itera sobre todo el dataset (1 paso por epoch para Full, N pasos por epoch en Stochastic y $\frac{N}{\text{Batch-size}}$ para mini-batch)

Librerías de Deep Learning

- TensorFlow by Google



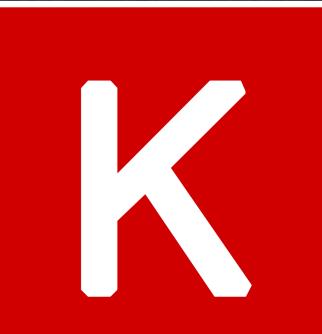
- Theano RIP (Padre de TensorFlow)



- PyTorch by Facebook (Torch para 🐍) Orientado a Investigación



- Keras (Librería de alto nivel que corre en Theano o TF)



- SparkML by Apache





Dask

Scalable Data Science

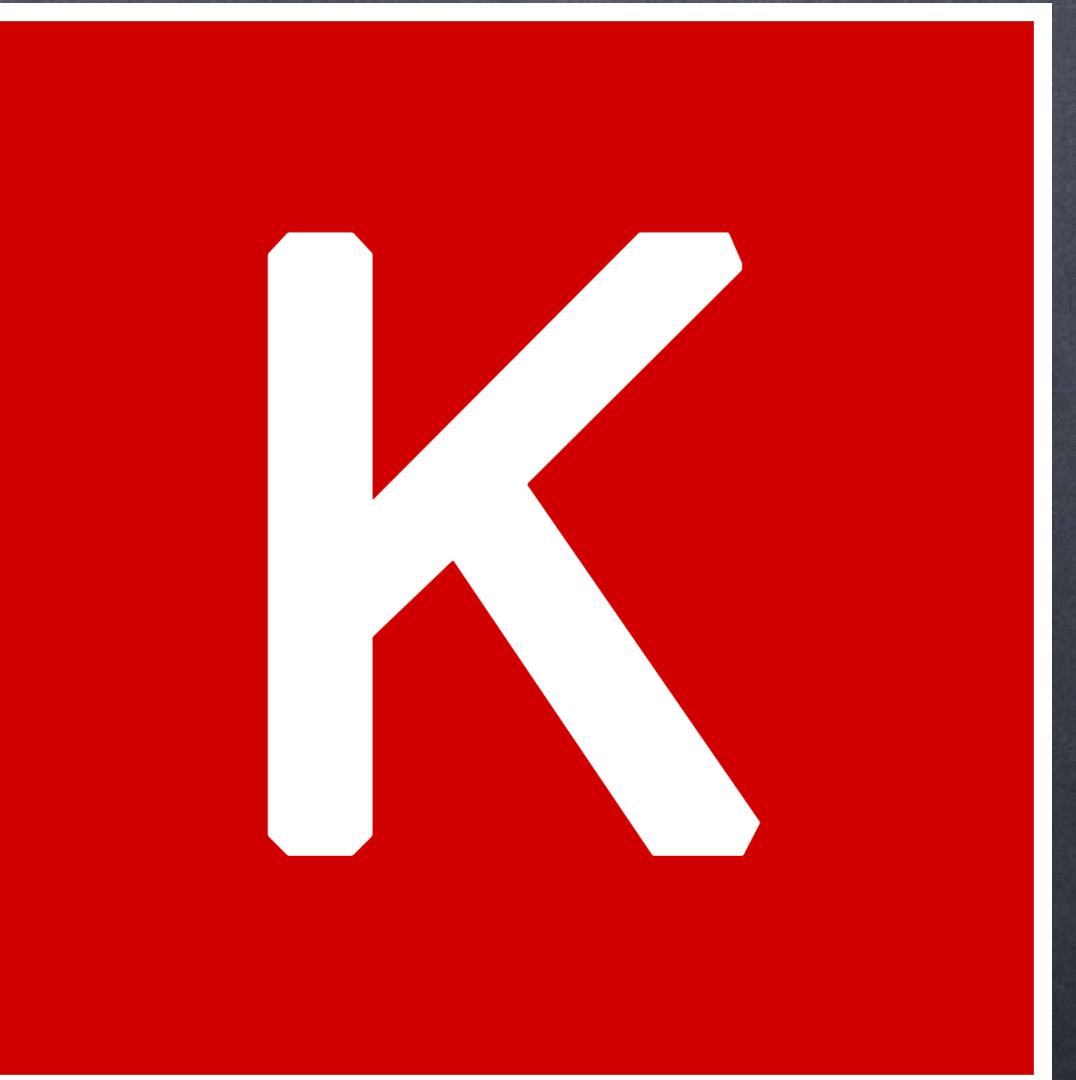


A tener en cuenta

- Para clasificación binaria la última capa se deja con un solo nodo y activación sigmoide
- Para clasificación multilase se dejan N nodos con función de activación softmax (Espacio N -dimensional donde cada eje corresponde a una clase)
- ReLu permite tener redes más dispersas por anularse para valores menores a cero, lo que ayuda a la regularización
- Después de cada epoch se recomienda hacer shuffling al dataset
- En mini batch el tamaño del batch suele ser 16 – 32. Entre mayor más lento pero paso más acertado
- Tener en cuenta los optimizadores y la regularización

Keras

- Decir cual es el tipo de layer, su función de pérdida, métrica, optimizados, taza de aprendizaje
- Entrenar el modelo especificando tamaño de batch y número de epochs
- Predecir
- Evaluar resultados
- Se pueden construir modelos
- Secuenciales: Stack de layers
- Functional API: Más compleja y detallada

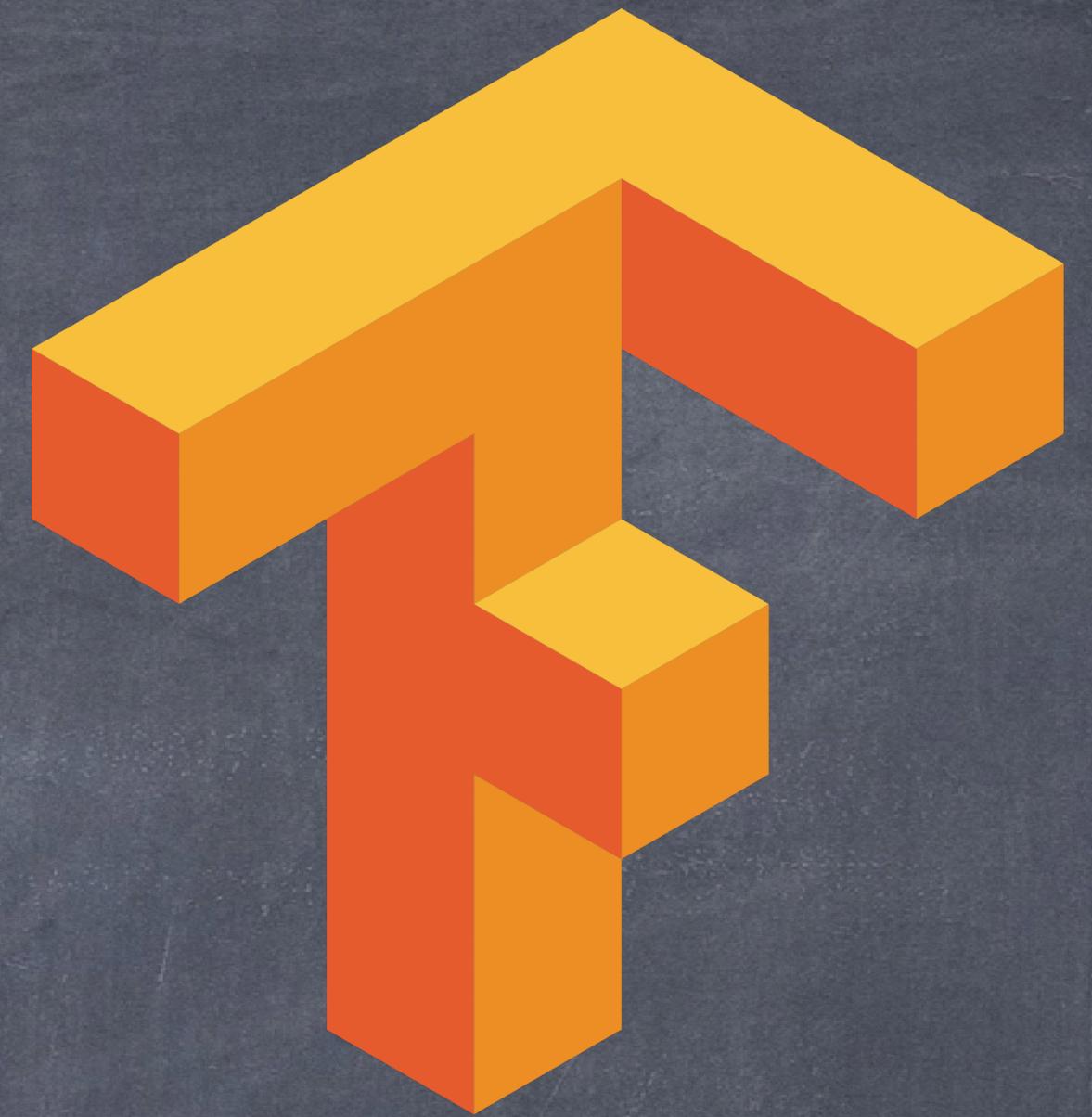


PyTorch



- Elementos son tensores
- .view(-1,1) cambia a tensor fila
- .numpy() para pasar a numpy (y graficar)
- .item() deja ver los tensores como números
- Métodos análogos a numpy
- requires_grad le dice que ese valor se usará para calcular derivadas
- is_leaf dice si es el último valor para el cálculo de la derivada por regla de la cadena
- grad almacena el gradiente del tensor cuando se calcula una derivada respecto a él
- detach() lo excluye de futuro seguimiento y permite usar numpy()
- Los datasets son clases que se construyen, al igual que las transformaciones y las redes

TensorFlow



- La estructura de TF se basa en la ejecución de un grafo de flujo, en donde los nodos son las operaciones matemáticas (datos que se consumen) y los enlaces tensores multidimensionales (Unidades de computo)
- TF bajo nivel (Sin Keras) admite Eager Execution (Evaluación instantánea sin requerir una sesión para ser ejecutado)

