

Regresión Lineal con **Keras**

- Librería de Redes Neuronales Profundas
 - Objetivo: Facilidad de uso
 - Prototipado rápido de modelos
 - Modelos pre-entrenados
 - Python
 - Alto nivel
 - Programar a nivel de **capas**
 - **Capa ~= Transformación Entrenable**
 - Regresión Lineal = 1 capa
 - Backend
 - Implementa operaciones eficientes
 - TensorFlow (defecto)
 - Theano o CNTK (ya no se usan)

Definición de un modelo

```
from keras.models import Sequential
from keras.layers import Dense
#Crear un modelo de capas secuenciales
model = Sequential()
```


```
#Agregar capa Densa
model.add(Dense(1, input_shape=[4]))
```

- Dense: $f(x) = wx+b$
- 4 variables de entrada
- 1 variables de salida

```
# compilar el modelo
model.compile(optimizer= 'sgd', loss= 'mse')
```

- 'sgd': Descenso de Gradiente Estocástico
- 'mse': Error Cuadrático Medio

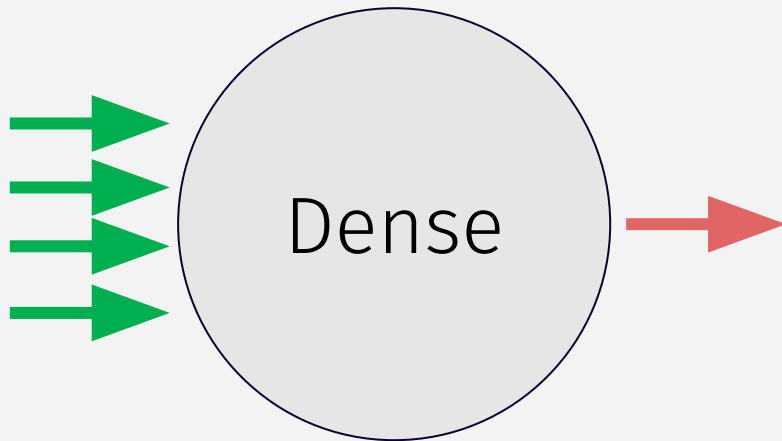
```
# Imprimir resumen del modelo
model.summary()
```



Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 1)	5
Total params: 5		
Trainable params: 5		
Non-trainable params: 0		

Capa Dense

```
from keras.models import Sequential
from keras.layers import Dense
model = Sequential()
model.add(Dense(1, input_shape=[4]))
```



Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 1)	5
Total params: 5		
Trainable params: 5		
Non-trainable params: 0		

- 5 parámetros
 - 4 de w
 - 4 variables de entrada
 - 1 de salida
 - $w: 4 \times 1$
 - 1 de b
 - 1 de salida
 - $b: 1 \times 1$

```
# cargar datos de alguna forma
# X e Y deben ser matrices de NumPy
X,Y = cargar_dataset()

# entrenar el modelo con 50 épocas
# tamaño de lote 32 y datos X e Y
history = model.fit(X, Y, epochs=50,
batch_size=32)

# Predecir la salida de X
Y_predicted = model.predict(X)

# Calcula el score del modelo
E = model.evaluate(X, Y)
print(f"El error del modelo es {E}")
```

- Métodos de model
 - fit: entrena el modelo
 - predict: calcula la salida del modelo $f(x)$
 - evaluate: calcula el error del modelo (y otras métricas)
- history: error al final de cada época

Calculando las dimensiones a partir de X e Y



```
# Datos
X,Y = cargar_dataset()
nx,d_in = X.shape # X tiene tamaño n x d_in
ny,d_out = Y.shape # Y tiene tamaño n x d_out
assert(nx=ny) # misma cantidad de ejemplos en ambos vectores
```

```
#Modelo
import keras
model=keras.Sequential()
# Capa lineal que convierte vector de d_in a d_out dims
model.add(keras.Dense(d_out,input_shape=[d_in]))
model.compile(loss='mse', optimizer='sgd')
```

```
# Entrenamiento y evaluación
model.fit(X,Y,epochs=100,batch_size=32,shuffle=True)
E=model.evaluate(X,Y)
```