Inteligencia Artificial LFIS419

Clase 9: Regresión

Profesor: Jorge Arevalo (jorge.arevalo@uv.cl jab@meteo.uv.cl)

Miércoles 7 de junio de 2023

Objetivos de la sesión

Regresión

- El objetivo es estimar un valor ”real” a partir de las variables de entrada.
- Muchas técnicas de clasificación tienen su contraparte para regresión
  - Random forest, en vez de usar mayoría, usa promedio
  - kNN usa el promedio de los vecinos
- Regresión lineal simple y múltple son muy usadas
- Algunos métodos proveen una medida de la incertidumbre

Regresión lineal

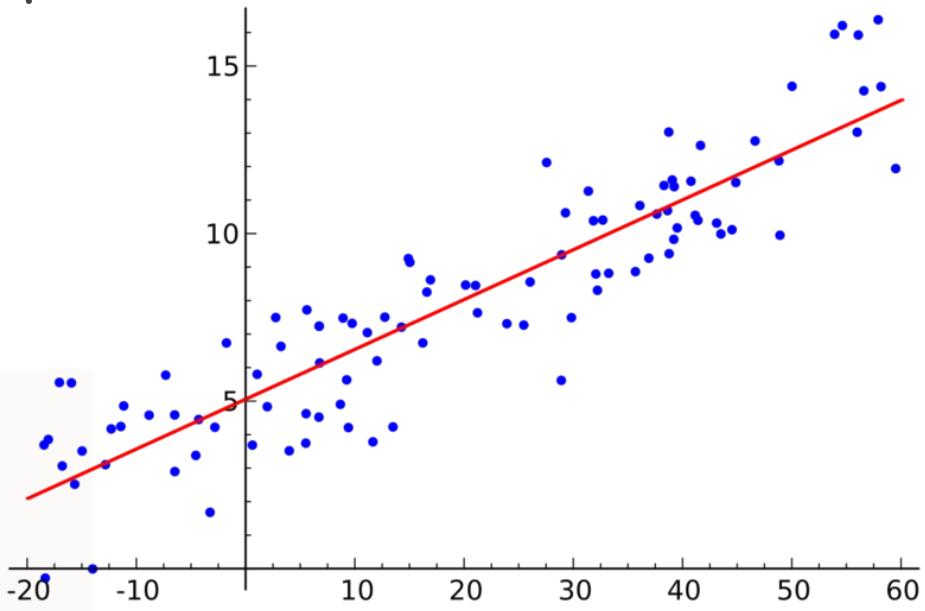
- Tiene como objetivo aproximar el valor de una función real de n variables, de la que sólo se conocen algunos valores, a través de una función lineal (hiperplano).
- Asume que los datos se pueden aproximar linealmente
- Correlación y coeficiente de determinación son medidas importantes
- Con la aproximación obtenida se puede estimar el valor en puntos que no son parte de los datos de entrenamiento.
- Existe método determinístico para encontrar la mejor aproximación al conjunto de entrenamiento.

Regresión lineal

1. Y es la variable dependiente.
2. X es la variable independiente
3. β_0 y β_1 son dos constantes desconocidas que representan el punto de intersección y la pendiente respectivamente.
4. ε (epsilon) es la función de pérdida.

$$Y = \beta_0 + \beta_1 X + \varepsilon,$$

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_a X_a + \varepsilon$$



Regresión lineal

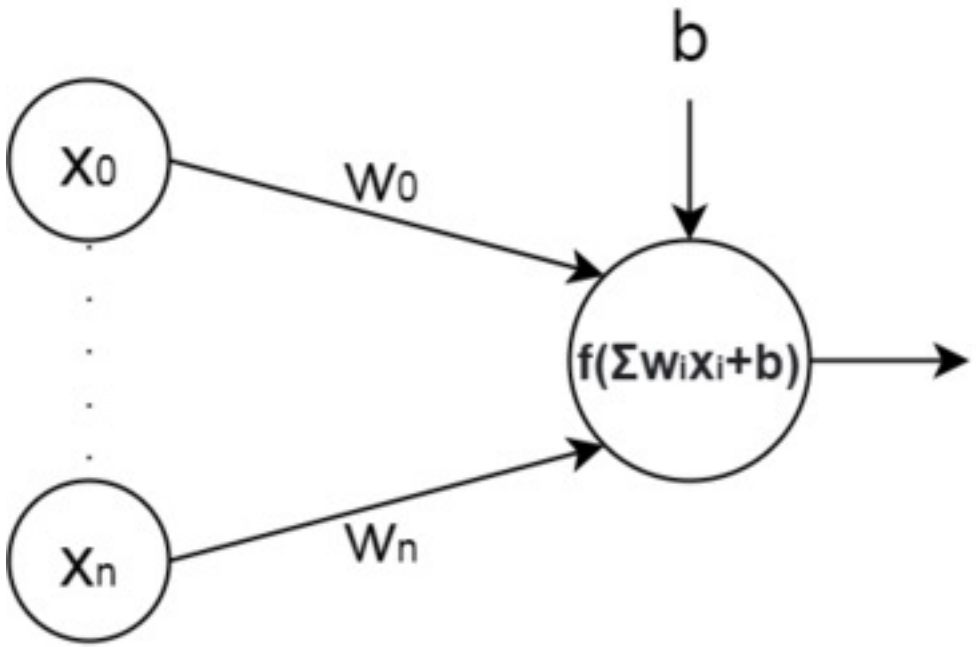
```
>>> import numpy as np
>>> from sklearn.linear_model import LinearRegression
>>> X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
>>> # y = 1 * x_0 + 2 * x_1 + 3
>>> y = np.dot(X, np.array([1, 2])) + 3
>>> reg = LinearRegression().fit(X, y)
>>> reg.score(X, y)
1.0
>>> reg.coef_
array([1., 2.])
>>> reg.intercept_
3.0...
>>> reg.predict(np.array([[3, 5]]))
array([16.])
```

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

Características deseables/requeridas

- Relación lineal
  - Los datos deben exhibir una relación lineal
  - Coeficiente de Correlación, scatter plot
- Independencia residual
  - No debe existir un patrón en los residuos (errores)
  - Test de Durbin-Watson
- Normalidad de los residuos
  - Los residuos (errores) deberían tener una distribución gaussiana
  - Q-Q plot, test de Kolmogorov-Smirnov
- Homocedasticidad
  - La varianza de los residuos se mantiene “constante”

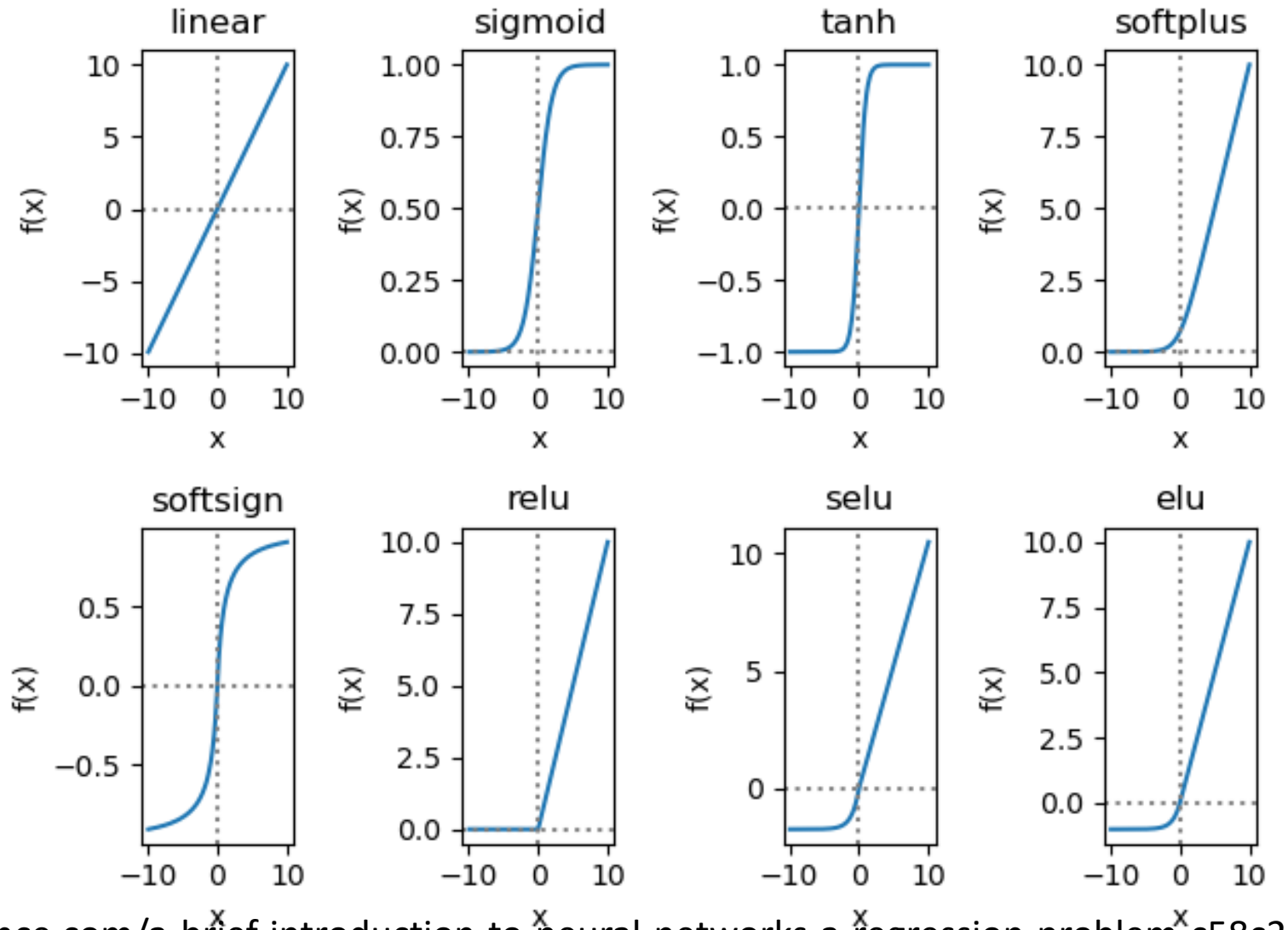
Redes Neuronales (neurona, nodo, unidad)



- **Pesos (Wi):** define el impacto de cada input en la salida de la neurona. Se inicializan antes del entrenamiento y se actualizan durante este.
- **Sesgo (b):** Intercepto, independiente de los inputs. Usado para desplazar la salida de la neurona. Se inicializan antes del entrenamiento y se actualizan durante este.
- **Función de Activación (f):** Define la salida de la neurona en función de la suma ponderada de los inputs y del sesgo. Varias elecciones posibles. Importante conocer el dominio y co-dominio de cada una.

https://towardsdatascience.com/a-brief-introduction-to-neural-networks-a-regression-problem-c58c26e18008

Redes neuronales (función de activación)



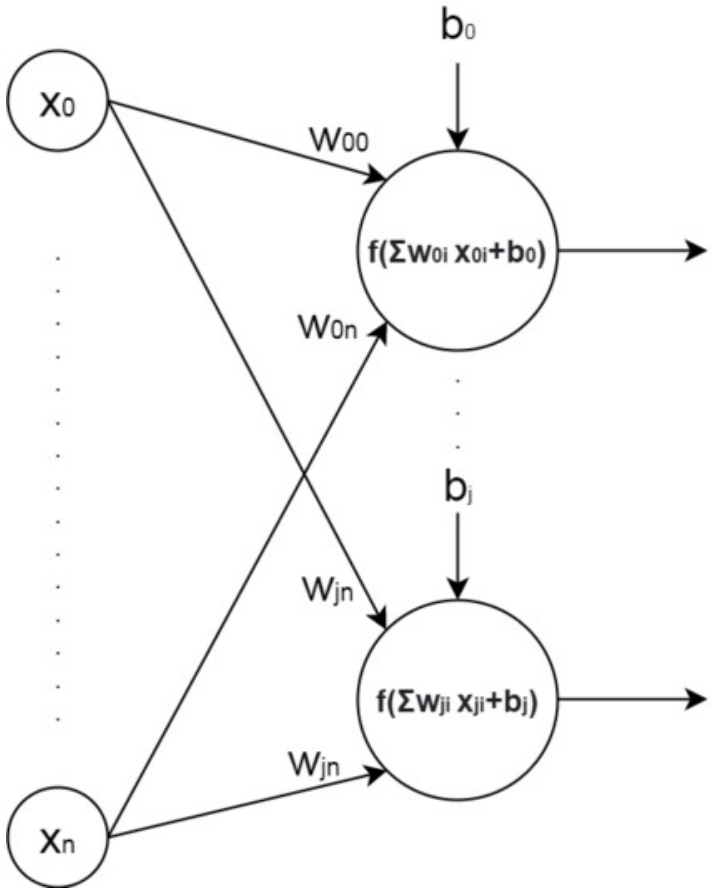
https://towardsdatascience.com/a-brief-introduction-to-neural-networks-a-regression-problem-c58c26e18008

Redes neuronales (capas)

Un conjunto de neuronas que tiene los mismos inputs, se llama capa.

Usualmente, todas las neuronas de una capa tienen la misma función de activación.

Hay distintos tipos de capas. En keras: convolución, pooling, normalización, regularización, input, dense y output.



https://towardsdatascience.com/a-brief-introduction-to-neural-networks-a-regression-problem-c58c26e18008

Redes neuronales (capas en keras)

- Input, usada para definir el número de features de entrada (shape).

```
keras.Input(
    shape=None,
    **kwargs
)
```

- Dense, las capas internas del modelo.
  - Units: número de neuronas en esta capa
  - Activation: función de activación
  - use_bias: booleano para indicar si usará sesgo
  - kernal_initializer y bias_initializer: definen cómo los pesos y el sesgo son inicializados.

```
keras.layers.Dense(
    units,
    activation=None,
    use_bias=True,
    kernel_initializer="glorot_uniform",
    bias_initializer="zeros",
    **kwargs
)
```

https://towardsdatascience.com/a-brief-introduction-to-neural-networks-a-regression-problem-c58c26e18008

Redes neuronales (multicapa)

- Input: Recibe las entradas del modelo
- Output: Es la última capa del modelo. La función de activación tiene que ser elegida cuidadosamente, dependiendo del tipo de la salida.
- Capas ocultas: Capas entre el input y el output.

Redes neuronales (keras, imports y reproductibilidad)

```python
from tensorflow.python.keras import Input
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dense
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import median_absolute_error, mean_absolute_percentage_error
from numpy import arange
```

import_libraries.py hosted with ❤ by GitHub view raw

```python
from numpy.random import seed
seed(1)
from tensorflow import random, config
random.set_seed(1)
config.experimental.enable_op_determinism()
import random
random.seed(2)
```

results_reproduction.py hosted with ❤ by GitHub view raw

Redes neuronales (keras: creación del modelo)

```python
# Create model:
model = Sequential()
model.add(Input(shape=(1,)))
model.add(Dense(200, activation='sigmoid'))
model.add(Dense(200, activation='sigmoid'))
model.add(Dense(200, activation='sigmoid'))
model.add(Dense(1, activation='linear'))
print(model.summary())
```

regression_model.py hosted with ❤ by GitHub view raw

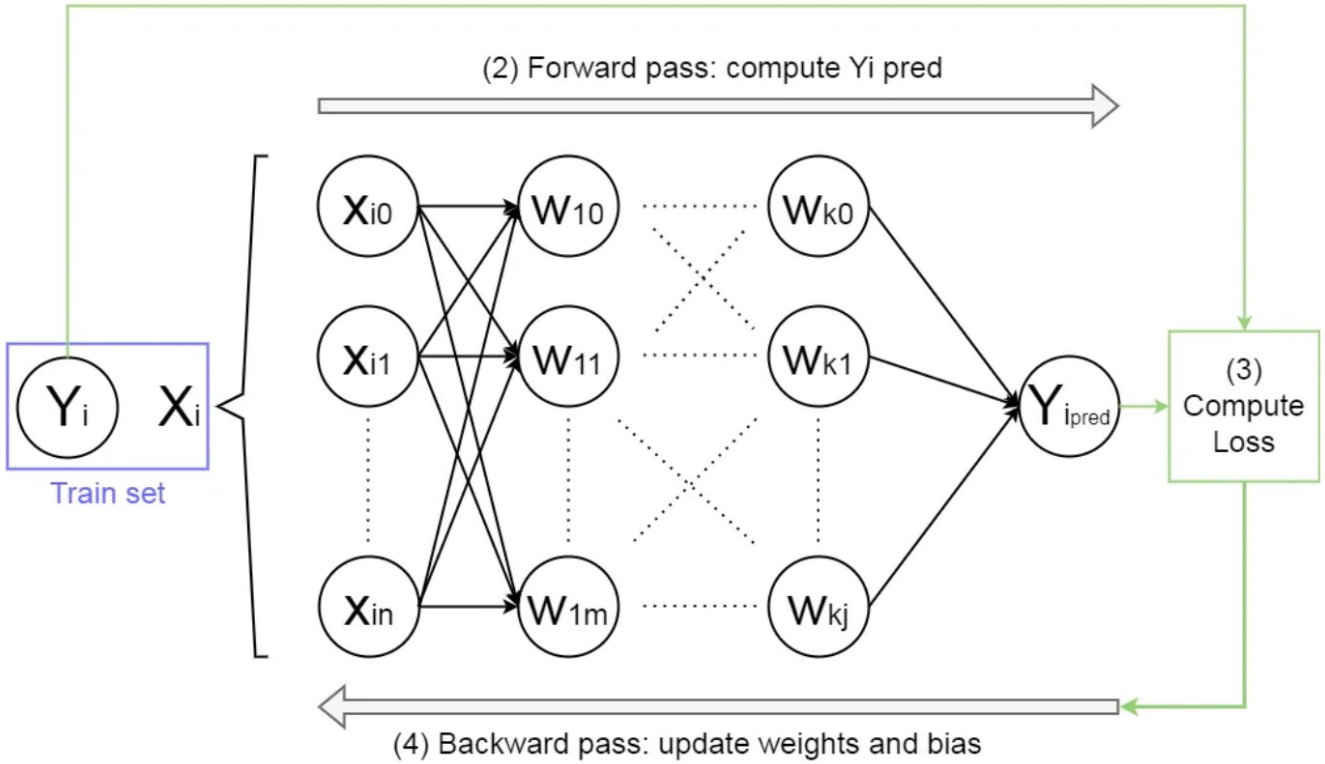
```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 dense (Dense)                (None, 200)               400

 dense_1 (Dense)              (None, 200)               40200

 dense_2 (Dense)              (None, 200)               40200

 dense_3 (Dense)              (None, 1)                 201
```

Redes neuronales (entrenamiento)



Training algorithm for feed forward network

Redes neuronales (entrenamiento)

- **Optimizador:** método para actualizar los pesos y sesgos.
- **Función de pérdida:** medida del “error” del modelo.
- **Batch:** número de muestras usadas para cada iteración de la actualización de pesos.
- **Epochs:** Cuántas repeticiones del entrenamiento se realizan con todos los datos de entrenamiento.
- **Tasa de aprendizaje:** Controla cuánto afecta la propagación de errores los pesos.

Redes neuronales (keras, compilación)

```
# Train:
epochs = 1750
x_train, y_train = train['x'], train['y']
model.compile(loss='mse', optimizer='adam', metrics=['mae'])
history = model.fit(x_train, y_train, epochs=epochs, batch_size=64, verbose=1, validation
```

training_regression.py hosted with ❤ by GitHub view raw

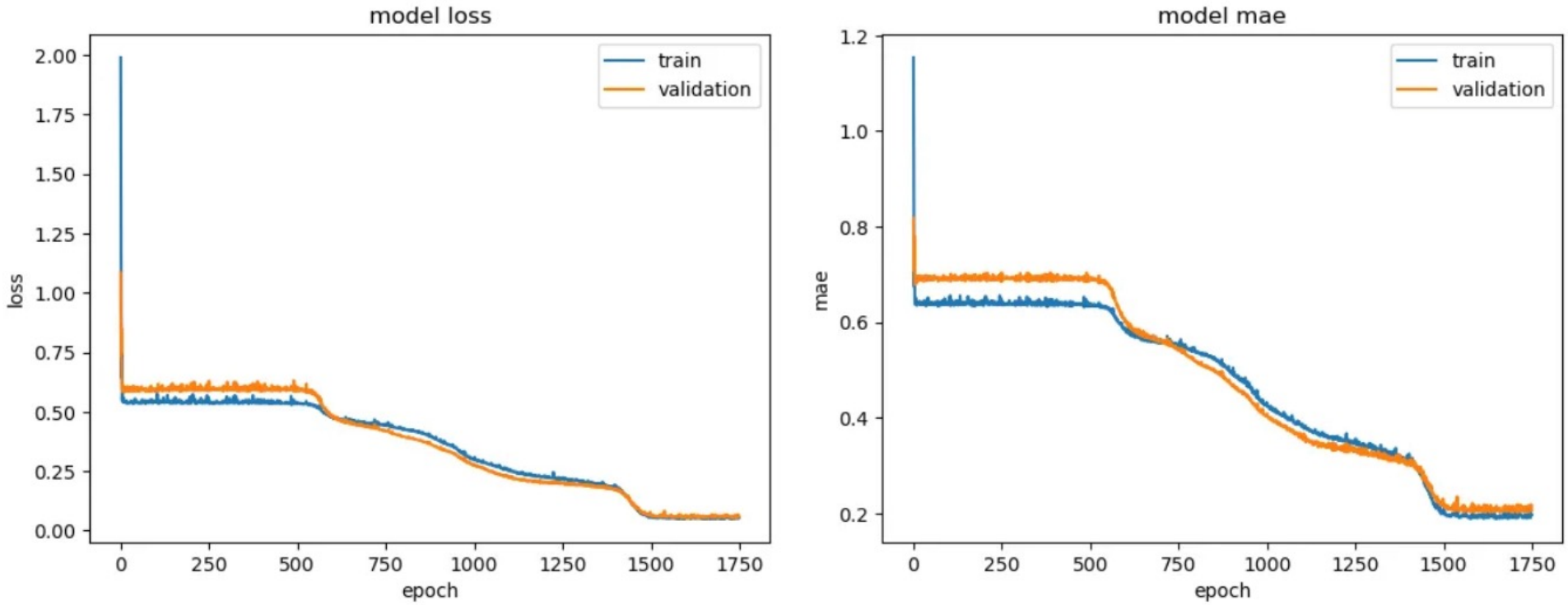
Redes neuronales (curvas de aprendizaje)

```python
# Display loss:
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'])
plt.show()

# Display metric:
plt.plot(history.history['mae'])
plt.plot(history.history['val_mae'])
plt.title('model mean absolute error (mae)')
plt.ylabel('mae')
plt.xlabel('epoch')
plt.legend(['train', 'test'])
plt.show()
```

learning_curves.py hosted with ❤ by GitHub view raw

Redes neuronales (curvas de aprendizaje)



Learning curves