Redes neuronales: Regresión lineal

¿Qué vamos a hacer?

- Usar el dataset de California Housing
- Entrenar una red neuronal multicapa
- Optimizar su regularización por validación cruzada
- Evaluar la red neuronal sobre el subset de test

Vamos a entrenar nuestra primera red neuronal (RNN). En este caso entrenaremos una RNN profunda o multicapa para regresión, con la función de activación RELU ("REctified Linear Unit function").

El dataset de regresión sobre el que entrenaremos el modelo será un dataset real: el dataset de predicción de precios inmobiliarios de California Housing, incluido en Scikit-learn.

Referencias:

- California Housing dataset
- sklearn.datasets.fetch_california_housing
- Neuronal network models: Regression
- sklearn.neural_network.MLPRegressor

In [ ]:
```
# TODO: Importa todos los módulos necesarios en esta celda
```

Descargar el dataset y analizarlo

Descarga el dataset de California Housing en formato *(data.data, data.target)* y analízalo para hacerte una idea de sus características, dimensionalidad, etc..

P. ej., *¿necesita ser normalizado? ¿Están ordenados aleatoriamente los ejemplos?*

In [ ]:
```
# TODO: Descarga el dataset y analiza algunos de sus ejemplos
```

Preprocesar los datos

De nuevo, vamos a preprocesar los datos siguiendo nuestros pasos habituales, siempre que sea necesario:

- Reordena los datos aleatoriamente.
- Normalízalos.
- Divídelos en subset de entrenamiento y test (usaremos validación cruzada por K-fold).

In [ ]:
```
# TODO: Reordena los datos aleatoriamente
```

In [ ]:
```
# TODO: Normaliza los datos
```

In [ ]:
```
# TODO: Divídelos en subset de entrenamiento y test
```

Entrena una RNN multicapa inicial

De nuevo, para comprobar nuestra implementación y que un modelo de regresión lineal por RNN multicapa podría resolver este dataset, vamos a entrenar una RNN inicial, sin regularización.

Entrénalo sobre el subset de entrenamiento y evalúalo con su método *score()* sobre el subset de test:

In [ ]:
```
# TODO: Entrena una RNN multicapa sin regularización
# Como topología, usa 2 capas intermedias de 25 nodos cada una
hidden_layer_sizes = (25, 25)
```

Optimiza la RNN por validación cruzada

Vamos a optimizar los diferentes hiper-parámetros de la RNN para nuestro modelo por CV, usando K-fold.

Vamos a usar sklearn.model_selection.GridSearchCV puesto que tenemos que optimizar 2 valores a la vez:

- *hidden_layer_sizes*: el nº de capas ocultas y nº de neuronas por capa, en el rango [1, 4] capas ocultas y [10, 50] neuronas por capa.
- *alpha*: parámetro de regularización L2, en el rango [10^0, 10^7].

Según los recursos de tu entorno de trabajo y el tiempo que le lleve entrenar los modelos, puedes evaluar cuantos valores consideres convenientes en dichos rangos:

In [ ]:
```
# TODO: Entrena una RNN multicapa optimizando sus hiper-parámetros hidden_layer_sizes y alpha por CV
```

In [ ]:
```
# TODO: Escoge el modelo más óptimo entre los entrenados
```

Evalúa la RNN sobre el subset de test

Finalmente, evalúa la RNN sobre el subset de test usando su método *score()*, el coeficiente de determinación R^2:

In [ ]:
```
# TODO: Evalúa la RNN sobre el subset de test
```