Redes de neuronas artificiales

Rafael Jordá Muñoz

En este PDF se resumen las tres actividades que se van a realizar para comprender el algoritmo de ML basado en vecinos más cercanos.

1. Ejercicio 1

Se tiene una red neuronal con una capa oculta, siendo $s_l = 2, 3, 1$ la longitud de cada capa, W(1)= (-2, 1; 1, -1; 3, -1) y W(2)= (2, 3, 1) los vectores de peso de las dos primeras capas, y b(1)= (0; -1; 1), y b(2)= (0) los vectores con el sesgo que proporcionan cada una de las dos primeras capas a la siguiente. La red utiliza la función de activación sigmoide en todas las neuronas salvo la de salida, cuya función de activación es la identidad. Además, la función de coste es el error cuadrático. Dado el ejemplo (1, 1, 1):

a) Calcula z_i(l) y a_i(l) para todas las neuronas.

b) Usando el algoritmo de retro-propagación del error, calcula $\delta_i(l)$ para todas las neuronas.

c) Usando el algoritmo de retro-propagación del error, determina los valores finales de cada peso $W_{ij}(l)$ y bias $b_i(l)$ de la red neuronal tras finalizar la primera iteración del algoritmo, asumiendo un valor de λ=1, y una tasa de aprendizaje α=0.5.

Este ejercicio debe hacerse SIN usar scikit-learn, para aplicar directamente los conceptos teóricos del método.

2. Ejercicio 2

Dado el problema de clasificación Blood Transfusion Service Center:

a) La clase que implementa el Perceptrón multicapa (MLP) en problemas de clasificación en scikit-learn es sklearn.neural_network.MLPClassifier. Revisa los parámetros y métodos que tiene. Dado que en determinadas operaciones (división del conjunto de datos en entrenamiento y test, validación cruzada, entrenamiento) el resultado obtenido depende de números aleatorios, y se quiere que el resultado sea repetible en otras máquinas, es necesario que utilicéis la función np.random.seed(1) al comienzo del fichero .ipynb.

b) Divide los datos en entrenamiento (80 %) y test (20 %).

c) Realiza la experimentación con MLPClassifier usando los valores por defecto de los parámetros, excepto para activation=tanh y solver=lbfgs.Además, utiliza los siguientes hiper-parámetros:

- *hidden_layer_sizes*: prueba entre 1 y 3 capas ocultas, y varía el número de neuronas en dichas capas (el número de neuronas por capa debe ser el mismo en todas las capas ocultas)
- *alpha*: parámetro de regularización.

Comprueba que el número de iteraciones (max_iter) es suficiente para aprender correctamente. Muestra las gráficas del error de entrenamiento con validación cruzada (5-CV) frente al valor de los hiper-parámetros, y justifica la elección de los valores más apropiados. Para cada combinación de valor del número de capas ocultas y parámetro de regularización se debe generar una gráfica donde se represente en el eje horizontal el número de neuronas en la capa oculta. ¿Cuál es el menor error de validación cruzada, su desviación estándar y el valor del hiper-parámetro para el que se consigue?

d) Muestra la gráfica del error de test frente al valor de los hiper-parámetros (siguiendo el mismo esquema que en el apartado anterior), y valora si la gráfica del error de entrenamiento con validación cruzada ha hecho una buena estimación del error de test. ¿Cuál es el menor error de test y el valor del hiper-parámetro para el que se consigue? ¿Cuál es el error de test para el valor del hiper-parámetro seleccionado por la validación cruzada?

3. Ejercicio 3

Repite el ejercicio 2 pero para el problema de regresión Energy Efficiency con la variable de salida cooling load mediante la función sklearn.neural_network.MLPRegressor.