Documentación Tarea Adaline

adalyne.py

En la línea 1 se añade la clase AdalineGD(), que contiene Adaline con descenso de gradiente. Se compone de los parámetros eta (ratio de aprendizaje), n\_iter (numero de iteraciones, épocas) y random\_state (semilla para generar valores aleatorios). Compuesto de los atributos w\_ (pesos del modelo) y cost\_ (función de coste)

Constructor

En la línea 22 hay un constructor que guarda los parámetros de entrada (los mencionados en el párrafo anterior) en variables.

Pantalla de computadora con letras

El contenido generado por IA puede ser incorrecto.

* eta es el ratio de aprendizaje, que vale 0,01
* n\_iter es el número de épocas, vale 50
* random\_state fija una semilla aleatoria, para tener los mismos números aleatorios

Cada uno de los parámetros se guardan como atributos del objeto

Función fit()

En la línea 27 hay una función llamada fit(), que es para entrenar el modelo.

Una captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

* X es la matriz de datos de entrada
* Y el vector con la etiquetas asignadas
* rgen genera números aleatorios usando la semilla generada en el constructor, para usar los pesos aleatoriamente
* self.w\_ son los pesos del modelo. Se crean con valores cercanos a 0
* rgen.normal() genera números aleatorios de una distribución normal
* self.cost\_ guarda el coste de cada época. Con esto se sabe si el modelo está aprendiendo

En la misma función se puede observar un bucle, para entrenar el modelo. El bucle itera el número de epocas que le hemos asignado con n\_iter

Pantalla de computadora con letras

El contenido generado por IA puede ser incorrecto.

* net\_input = self.net\_input(X) calcula la entrada neta, que son la combinacion lineal de entradas y pesos
* output pasa la entrada neta a la funcion de activacion
* errors = y – output calcula los errores. Lo que predijo el modelo menos el resultado esperado
* self.w\_[1:] += self.eta \* X.T.dot(errors) actualiza los pesos menos el bias.
* cost Calcula el error cuadrático medio
* self.cost\_.append(cost) guarda el coste en la lista

La función devuelve el modelo entrenado

Función net\_input()

En la línea 51 hay una función que calcula la entrada neta.

Texto

El contenido generado por IA puede ser incorrecto.

* Calcula el resultado de multiplicar los pesos por las entradas
* Np.dot(X, self.w\_[1:]) multiplica las entradas por sus pesos
* + self.w\_[0]: suma el bias.

Función activation()

En la línea 54 hay una función que hace de activación lineal

Texto

El contenido generado por IA puede ser incorrecto.

Función predict()

En la línea 58 está la función predict(), para hacer la predicción final



* net\_input(x) calcula la entrada neta
* El valor anterior se pasa a la función de activacion (activation())
* np.where(>= 0.0, 1, -1) Indica que si la salida es mayor o igual que 0, la prediccion vale 1. Si es menor que 0, valdrá -1
* Al final devuele un array con las etiquetas predichas.