Reconocimiento de Patrones
Regresión polinomial y validación cruzada

Nicolás San Martín
Sebastián Sujarchuk
Eric Brandwein

May 2020

1 Introducción

En este taller, implementaremos algunas funciones para llevar a cabo una regresión polinomial sobre un set de datos. En una regresión polinomial, partimos de un conjunto de observaciones (x_i, t_i), donde según nuestro modelo, t_i depende de x_i. Que sea polinomial significa que en nuestro modelo vamos a intentar predecir el valor de t_i como un polinomio evaluado en x_i. Para esto, partimos de un conjunto llamado de *entrenamiento*, en el cual tenemos valores de distintos x_i con sus correspondientes t_i observados (o simulados). Usamos este conjunto para obtener los valores correspondientes a los coeficientes del polinomio, y una vez obtenidos, utilizamos este modelo para predecir valores de t_i para x_i nuevos. Para esto, utilizamos el método de cuadrados mínimos lineales, con el cual obtenemos los w que minimizan las diferencias cuadráticas entre los valores observados y los predichos de los t_i.

Usaremos Python para implementar los procedimientos de generación de datos y de ajuste polinomial, valiéndonos de las bibliotecas numpy, pandas, y seaborn.

2 Ejercicio 1: Generación de Datos

Generamos 200 simulaciones de muestreos con 25 puntos cada uno. Los x de los puntos fueron generados con una variable aleatoria de distribución uniforme en el intervalo [0, 1]. Para calcular el t para cada punto, utilizamos una función seno, y agregamos un error de distribución normal con parámetros $\mu = 0$ y $\sigma^2 = 0.3$ a cada punto.

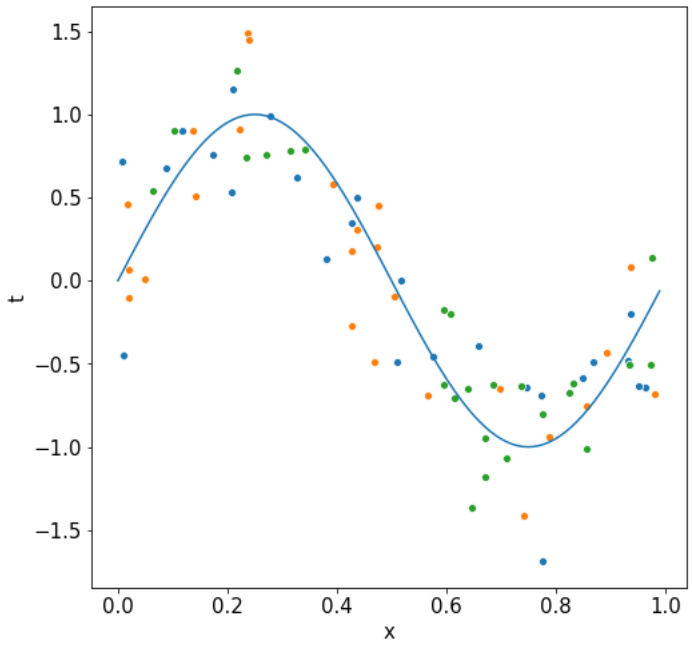

Figure 1: Pares (x, t) correspondientes a tres de los 200 datasets generados.

3 Ejercicio 2: Cálculo del error

Para este ejercicio implementamos la función `solve_weights` que recibe tres parámetros: un data set (que es un dataframe de pandas), M, que es la cantidad de coordenadas en w, y λ, el término de regularización. Esta función devuelve el vector w^* de pesos que minimizan el error cuadrático dados los hiperparámetros M, λ.

Para encontrar w^* utilizamos las ecuaciones normales, i.e.

$$w^* = (\lambda \mathbb{I} + \Phi^t \Phi)^{-1} \Phi^t t$$

donde la matriz Φ es

$$\begin{bmatrix} x_1^0 & x_1^1 & ... & x_1^m \\ x_2^0 & x_2^1 & ... & x_1^m \\ \vdots & \vdots & \ddots & \vdots \\ x_n^0 & x_n^1 & ... & x_n^m \end{bmatrix}$$

y $t^t = [t_1, ..., t_n]$

Si usamos $\lambda = 0$ entonces nuestra ecuación corresponde a la solución de cuadrados mínimos sin término de regularización.

Para encontrar w^*, no calculamos la inversa de $\lambda\mathbb{I} + \Phi^t\Phi$ sino que resolvemos

$$(\lambda\mathbb{I} + \Phi^t\Phi)w = \Phi^t t$$

con la función `linalg.solve` de numpy.

Cada modelo (con w fijo) tendrá un error, que es el promedio de la suma de los cuadrados de las diferencias entre los valores predichos y los valores observados (o simulados) en el data set. Podemos realizar una estimación para el valor esperado del error, y su desviación standard, calculando el promedio de dicho error,

$$\hat{\mu}_E = \frac{1}{L}\sum_{k=1}^{L} E_{D_k}(w)$$

donde

$$E_{D_i} = \frac{1}{N}\sum_{n=1}^{N}(y(x_n, w) - t_n)^2$$

,

para los puntos (x_n, t_n) pertenecientes el dataset D_k, así como su desvío estándard empírico.

$$\hat{\sigma}_E = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(E_{D_i} - \mu_E)^2}$$

Mostramos el promedio del error y el desvío estándard para valores de hiperparámetros $M = [4, 6, 10]$ y $\lambda = [0, 0.005, 0.000001]$.

M	λ	Promedio del error	Desvío estándard
4	0	1.39	0.38
4	1e-06	1.39	0.38
4	0.005	2.50	0.61
6	0	1.38	0.38
6	1e-06	1.39	0.38
6	0.005	1.76	0.46
10	0	2.75	1.96
10	1e-06	1.62	0.55
10	0.005	1.72	0.45

Table 1: Promedio y Desvío estándard de polinomios variando M y λ.

El valor del término de regularización λ permite contrarrestar el *overfitting*, que ocurre por ejemplo cuando el modelo tiene un M demasiado grande y por lo tanto se ajusta al ruido, el cual no es generalizable. Eso se ve para el par de

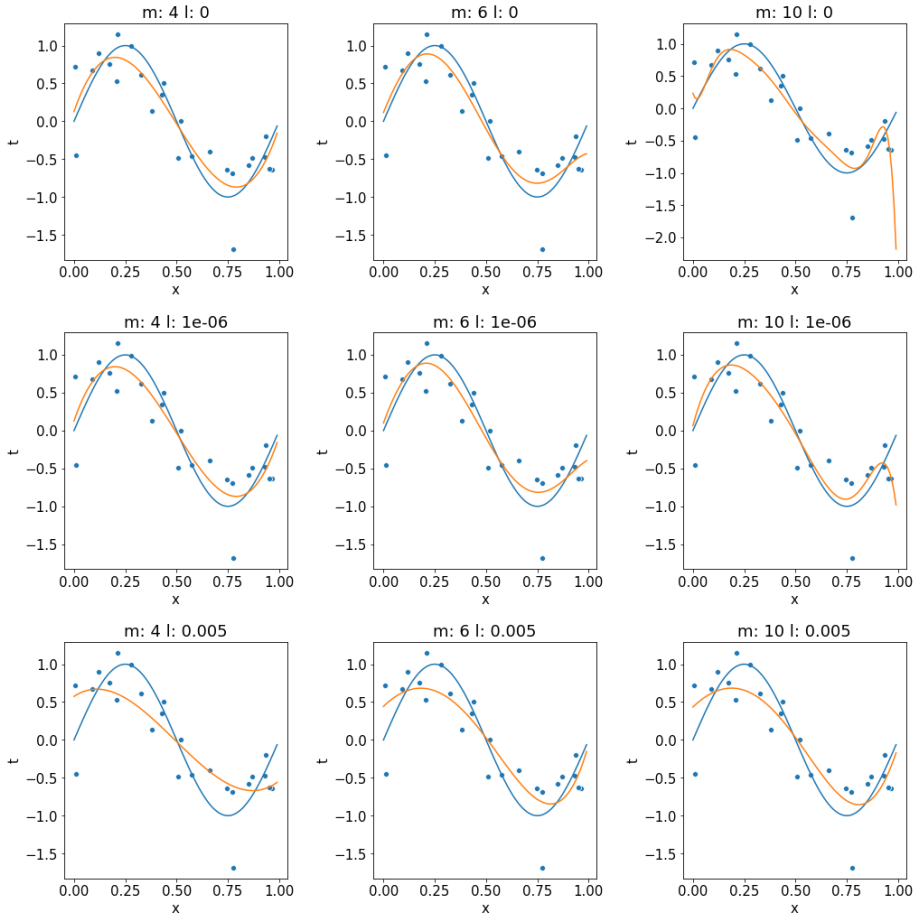

Figure 2: Funciones generadas variando hiperparámetros para un set de datos.

parámetros ($\lambda = 0, M = 10$), cuyo error medio es mayor al de los otros modelos con mismo M, y además en el gráfico se ve claramente cómo el polinomio correspondiente ajusta bien para los puntos de entrenamiento, pero se aleja en otros valores no pertenecientes al set de entrenamiento.

Por otra parte, vemos que con un valor de $\lambda = 0.005$, pese a ser mayor que $1e$-06, produce un error también mayor. Esto se debe, en cambio, al *underfitting*: al ser muy grande el término de regularización, se "castigan" valores de w que no son demasiado grandes. Esto se observa también en el gráfico en la convexidad de la curva, que es claramente menor a la de la curva usada en la generación de los datos.

4 Ejercicio 3: Validación cruzada

Para realizar la validación cruzada generamos un conjunto de training de 200 datasets, y 40 datasets de testing. Cada dataset tiene 10 puntos.

Dado un dataset de validación elegido de entre los del conjunto de training, hacer la validación para un par de hiperparámetros M y λ quiso decir, para nosotros, calcular un vector de pesos por cada uno de los otros datasets de training, calcular el vector promedio de todos estos vectores de pesos, y luego calcular el error de la predicción del modelo usando este vector contra el dataset de validación. Por cada par M y λ de entre los que elegimos en el punto 2, realizamos una validación por cada dataset en el conjunto de training, tomando cada uno como el dataset de validación cada vez. Luego, calculamos el error promedio de entre todos los errores resultantes de las validaciones, realizamos el mismo procedimiento para todos los otros pares de hiperparámetros, y comparamos los errores promedios para encontrar el mejor par.

M	λ	Error promedio
4	0	0.458
4	1e-06	0.458
4	0.005	1.320
6	0	2.119
6	1e-06	0.511
6	0.005	0.853
10	0	655367.209
10	1e-06	0.539
10	0.005	0.725

Table 2: Error promedio de validación de las curvas generadas con distintos hiperparámetros.

Vemos que las combinaciones de menor error promedio son las dos con $M = 4$ y con $\lambda = 0$ o $\lambda = 1e - 06$, con un error promedio de 0.485 contra el conjunto de verificación. En la figura 3 se puede ver la curva resultado de utilizar los hiperparámetros $M = 4$ y $\lambda = 1e - 06$ para calcular los pesos con los datasets de training y promediar los vectores resultantes para conseguir los coeficientes w.

Una cosa a notar es el alto error promedio que resulta de utilizar $M = 10$ y $\lambda = 0$, que es 6 órdenes de magnitud mayor que la mayoría de los errores del resto de los pares. Esto es probable que sea ocasionado por overfitting, ya que usar un polinomio de orden 10 permite directamente generar un polinomio interpolador para cada dataset de testing de 10 puntos.

Para calcular el error contra los conjuntos de testing, usamos los coeficientes w obtenidos de promediar cada uno de los vectores obtenidos con cada uno de los datasets de training para $M = 4$ y $\lambda = 1e - 06$. Luego, calculamos el error cuadrático medio de la predicción usando estos w para cada conjunto de testing, y posteriormente los promediamos. El resultado fue de 0.463. Creemos que es

una muy buena aproximación, tomando en cuenta que el error alcanzado por la función seno, que fue la función que utilizamos para generar los datos, fue de 0.449.

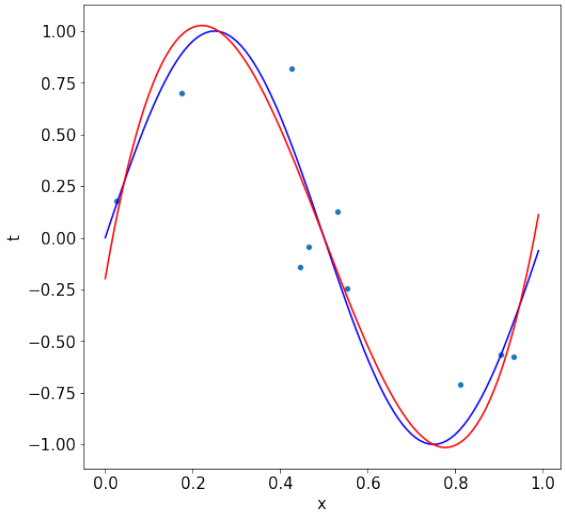

Figure 3: Curva en rojo generada con los hiperparámetros $M = 4, \lambda = 1\text{e} - 06$ usando los conjuntos de training, junto a puntos de un conjunto de testing y la curva de la función seno.