Aprendizaje automático

Práctica 4: Regresión Logística Multi-Clase

GAJAN Antoine (894825)

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Índice

Introducción

Como parte de la asignatura .ªprendizaje automático", los estudiantes se familiarizarán con los conceptos de inteligencia artificial. Más concretamente, durante el cuatrimestre se estudiarán las bases del aprendizaje supervisado y no supervisado.

En esta cuarta práctica, vamos a ampliar los conocimientos adquiridos en las prácticas anteriores a un caso multi-clase. En otras palabras, después de haber trabajado con el reconocimiento de un determinado tipo de vino en la práctica 3, ahora podremos establecer una regresión logística para reconocer una cifra de 0 a 9.

Esta memoria pondrá de relieve el proceso emprendido para responder a las preguntas, y producirá una reflexión personal sobre los resultados obtenidos.

1. Estudio previo

Durante el estudio antes de asistir a la práctica, me tomé el tiempo para revisar los conceptos y funciones mencionados en el curso. En particular, he reescrito el algoritmo de entrenamiento y clasificación multiclase .ºne vs all.ªplicando la regresión logística regularizada.

Algorithm 1 one_vs_all(K, X, y)

{Numero de clases diferentes}

$clases \leftarrow unique(y)$

$theta \leftarrow []$

{Para cada clase}

for $clase \in clases$ **do**

$y\_clase = y == clase$

$theta\_clase \leftarrow regularizacion(K, X, y\_clase)$

$theta \leftarrow [theta\ theta\_clase]$

end for

return theta

2. Memoria de la practica

2.1. Cuestión 1 : Regresión logística regularizada

Primero implementamos el algoritmo del estudio previo. Vamos a entrenar un clasificador por regresión logística para cada clase j, que estime la probabilidad de y = j.

Vamos a realizar 10 entrenamientos distintos, cada uno modelando la probabilidad de pertenecer a la clase j (j entre 1 y 10) del número estudiado.

Para cada entrenamiento realizado one vs all diferente, practicaremos la regresión logística regularizada como en la práctica anterior.

Sin embargo, para mejorar la velocidad de ejecución del código, he modificado la función de aprendizaje llamada LearnerRegularizado. En lugar de utilizar el método de Newton para minimizar la función de coste logístico regularizado, opté por el método LBFGS. En efecto, esta última realiza una aproximación de la segunda derivada, lo que permite ahorrar un tiempo precioso. Invertir una matriz de 400 filas y 10 columnas (4000 valores) es un proceso costoso.

Con el fin de no sobrecargar el informe, presentaremos los resultados obtenidos para el aprendizaje one vs all de la clase 1 solamente. En efecto, el proceso empleado se repetirá para los aprendizajes de las otras clases.

Con un método similar al utilizado en la práctica anterior, obtenemos el siguiente gráfico para la clase 1 :

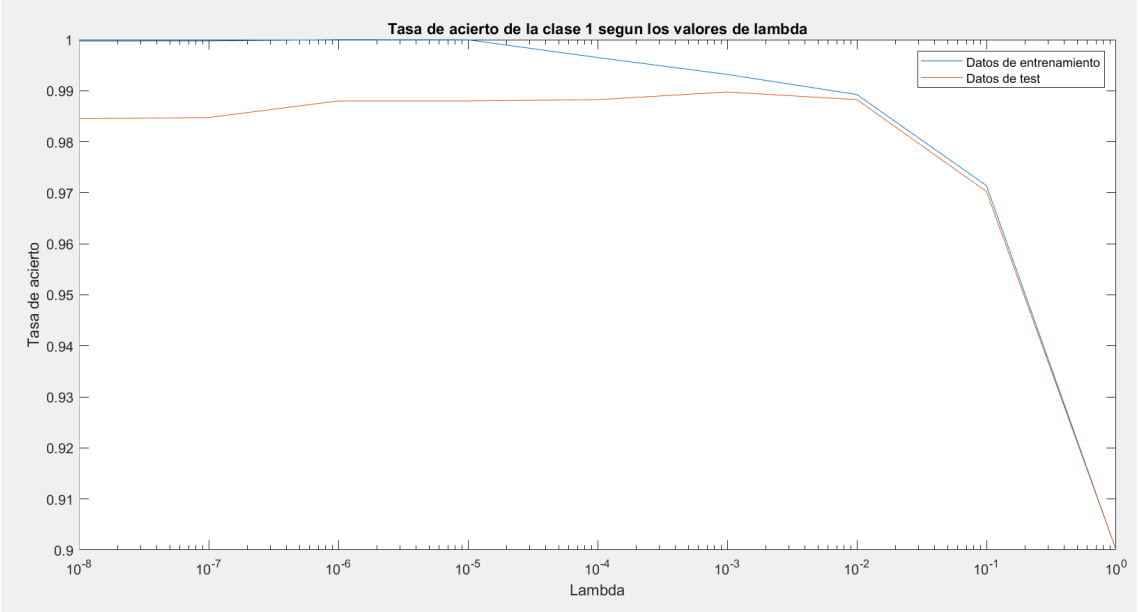

Figura 1: Evolución de la tasa de acierto para el aprendizaje de la clase 1

Dado que el error con los datos de validación es el más bajo cuando lambda =

10^{-3} elegimos lambda = 10^{-3} y ahora estamos aprendiendo todos los datos de entrenamiento con este parámetro de regularización. Obtenemos un vector de columna theta_clase de 400 valores, que representa los pesos de los 400 atributos (20 * 20 píxeles), que se tienen en cuenta para el aprendizaje.

Podemos repetir este proceso para las 10 clases a considerar. En cada etapa de aprendizaje k, añadimos al vector general theta una nueva columna que contiene los valores del vector theta_clase, que representa el aprendizaje de la clase k.

Obtenemos para cada uno de los modelos los gráficos siguientes, las conclusiones se deducen de la misma manera que anteriormente para elegir el valor de lambda.

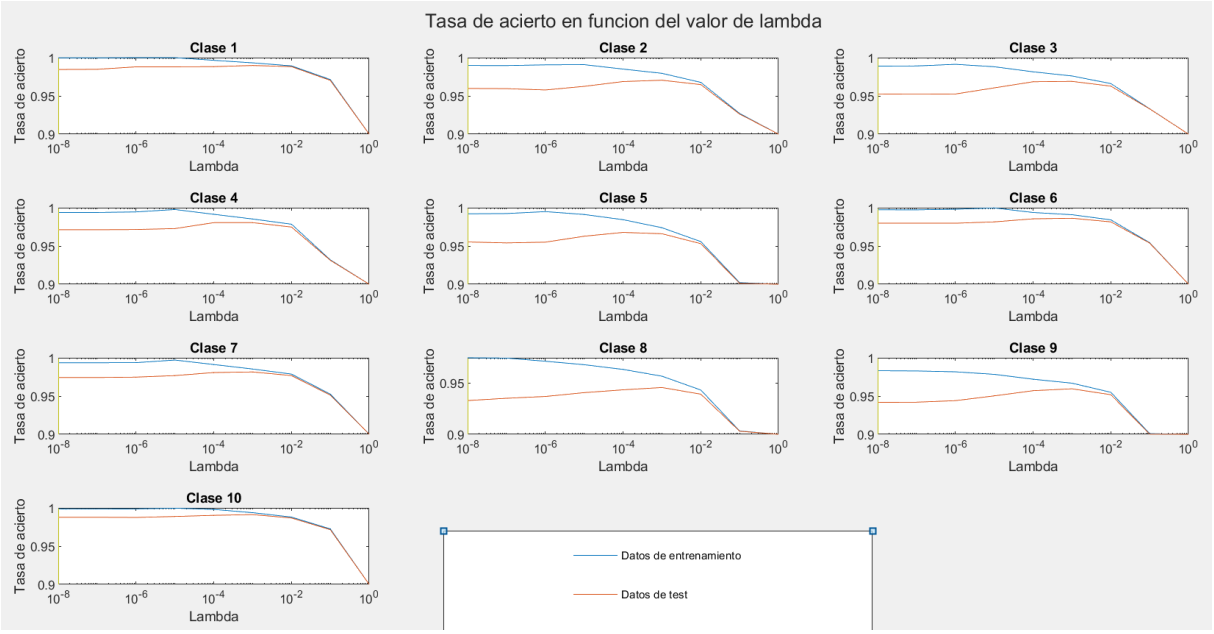

Figura 2: Evolución de la tasa de acierto para los diferentes aprendizajes

Así, se seleccionan los siguientes valores de lambda para las diferentes clases:

lase	Valor de lambda
1	10^{-3}
2	10^{-3}
3	10^{-3}
4	10^{-3}
5	10^{-4}
6	10^{-3}
7	10^{-3}
8	10^{-3}
9	10^{-3}
10	10^{-3}

Cuadro 1: Valores de lambda para las diferentes clases

Al final, obtenemos un vector theta de dimensión 400 * 10.

2.2. Cuestión 2 : Evaluación del modelo final

2.2.1. Evaluación del modelo con la métrica adecuada

Ahora tenemos los valores de peso en el vector theta. Así que vamos a poder evaluar el modelo con una métrica adecuada.

Podemos observar que el problema está equilibrado, en el sentido de que cada clase está representada uniformemente. De hecho, por cada número entre 0 y 9, tenemos exactamente 200 imágenes en nuestro juego de entrenamiento. La métrica más fácil de calcular, analizar y adaptar a nuestro caso concreto es la tasa de acierto.

Sin embargo, para poder calcularla, primero habría que encontrar la manera de calcular las clases previstas. En efecto, aplicando el producto matricial como en las prácticas anteriores, se obtienen para cada imagen 10 probabilidades, que corresponde respectivamente a las probabilidades de ser uno de los dígitos entre 0 y 9. Por lo tanto, la clase predicha para una imagen es en realidad el índice de la probabilidad más alta calculada. Por lo tanto, para lograr esto, podemos utilizar el siguiente truco en Matlab. Dado que las 10 probabilidades de una imagen están en línea, debemos extraer el índice del máximo de cada línea. Para ello, tenemos el siguiente código:

```
[~,y_pred_tr] = max(pred(X, theta),[],2);
[~,y_pred_test] = max(pred(Xtest, theta),[],2);
```

Así que obtuvimos un vector con números enteros entre 1 y 10, modelando la clase predicha para una imagen de entrada.

La tasa de acierto se calcula de la siguiente manera: tasa_acierto = num_buenas_pred / num_total_pred. Para ello, tenemos el siguiente código en Matlab:

```
accuracy_tr = mean(y_pred_tr == y)
accuracy_test = mean(y_pred_test == ytest)
```

Al calcular la métrica adecuada propuesta anteriormente, obtenemos los siguientes resultados:

	Tasa de acierto
Datos de entrenamiento	0.9745
Datos de test	0.8730

Cuadro 2: Tasa de acierto con los datos MNIST

El modelo propuesto anteriormente parece adecuado. De hecho, tenemos una alta tasa de acierto, tanto con datos de entrenamiento como de test. Aunque la tasa de acierto es un poco más baja con los datos de test, me parece difícil hablar de sobreajusto aquí.

2.2.2. Comprensión de las confusiones del modelo

Utilizando la función adjunta, podemos observar varios ejemplos de confusiones realizadas por nuestro modelo, ya sea con datos de entrenamiento o de test.

Obtenemos esto con los datos de entrenamiento y test:

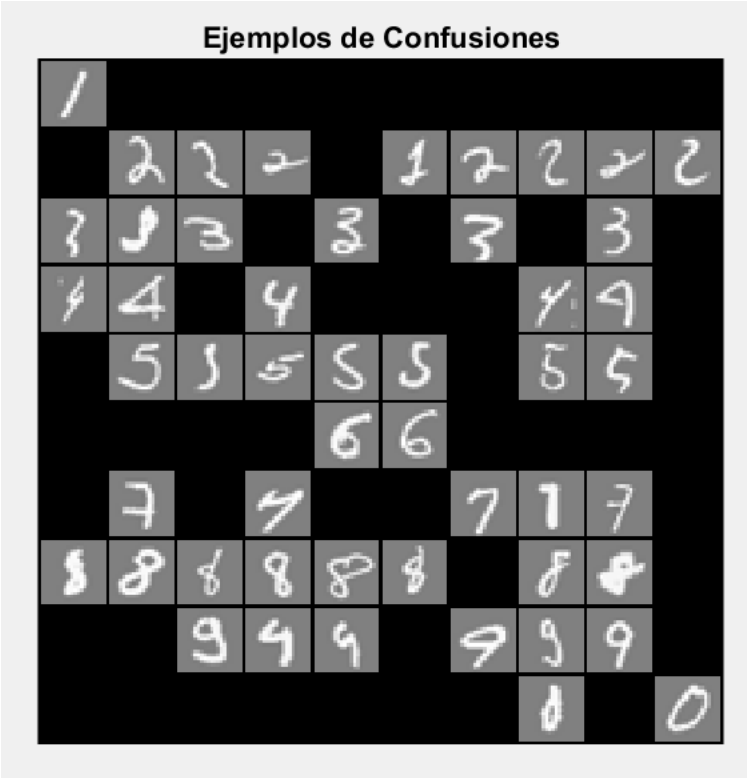

Figura 3: Ejemplos de confusiones con los datos de entrenamiento

En el mejor de los casos, lo ideal sería tener solo imágenes en la diagonal. Esto significaría que el modelo no cometió ningún error y que solo predijo las mismas clases de salida que las clases reales de las imágenes.

En particular, podemos notar que con los datos de entrenamiento, nuestro modelo tiene dificultades con el número 2. De hecho, en muchos casos, el modelo predijo el número 2 cuando en realidad no era así. Podemos sacar las mismas conclusiones para el número 8.

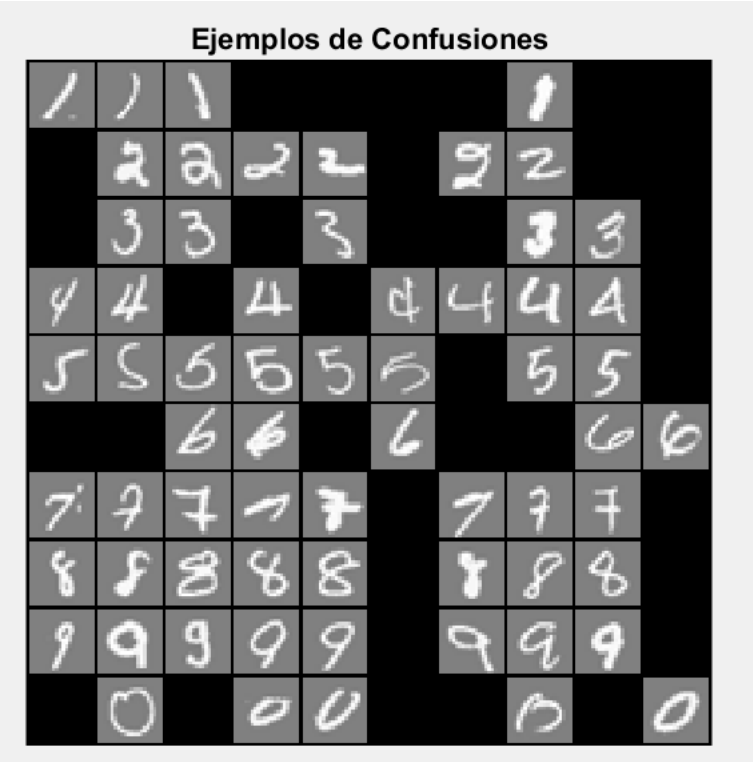

Figura 4: Ejemplos de confusiones con los datos de test

Con los datos de test, el modelo cometió un poco más de error. Esto se vio en el estudio de la métrica, y se confirma de nuevo. En efecto, se observa que casi toda la matriz contiene imágenes, lo que significa que se han cometido errores en las predicciones de las diferentes cifras.

Ahora que tenemos conocimiento de algunos errores de predicción cometidos, sería interesante poder estudiar su frecuencia para ver cuáles son los errores más frecuentes.

Para ello, vamos a construir una matriz de confusión. Esto nos permitirá ver el número de predicciones buenas y malas para los diferentes números, con datos de entrenamiento y datos de test.

En un primer momento, me encargué de construirla yo misma, para comprender cómo podía ser creada. Luego, con el fin de mejorar la representación visual, utilicé las funciones confusionmat y confusionchart que permiten respectivamente crear una matriz de confusión y representarla gráficamente.

He obtenido las siguientes matrices de confusión con los datos de entrenamiento y validación:

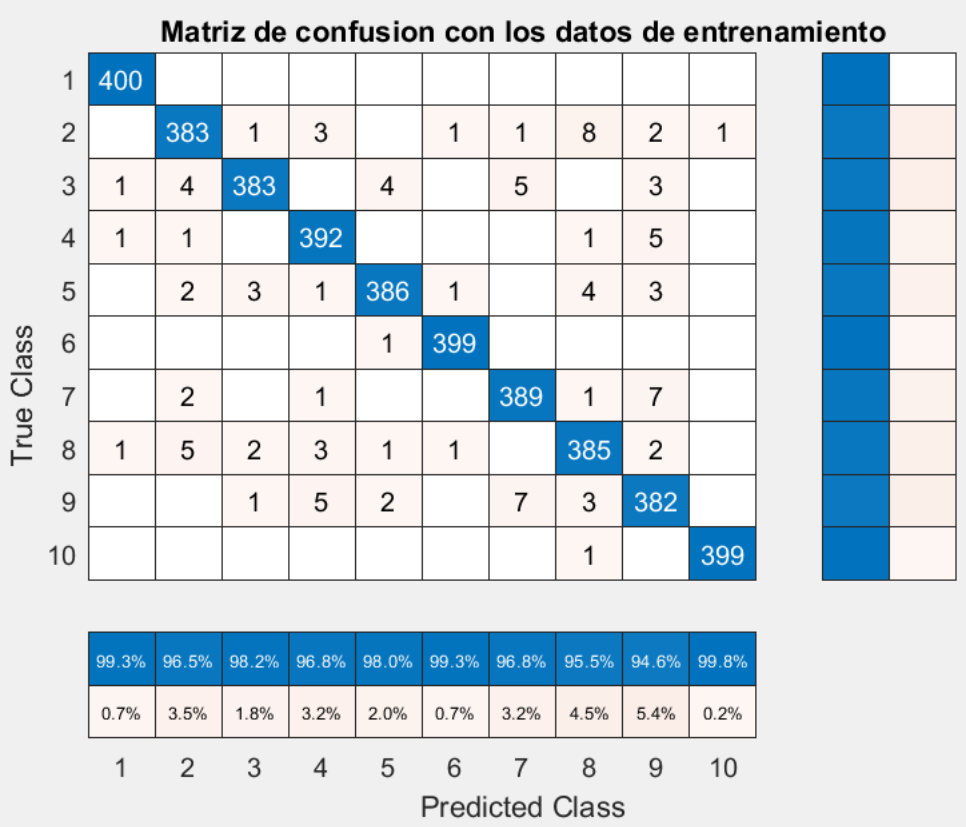

Figura 5: Matriz de confusión con los datos de entrenamiento

Los valores más altos están presentes en la diagonal, lo que es tranquilizador. Esto significa que nuestro modelo predijo correctamente la mayoría de los números en las imágenes. Sin embargo, los datos interesantes para el estudio de la confusión están fuera de la diagonal. En particular, se observa que el modelo predijo 8 veces la clase 8 en lugar de la clase 2. También se observa que el modelo invirtió 7 veces los números 7 y 9, en un sentido como en el otro. Se deduce que con los datos de entrenamiento, nuestro modelo tiene dificultades para detectar el número 2 y confunde a menudo los números 7 y 9.

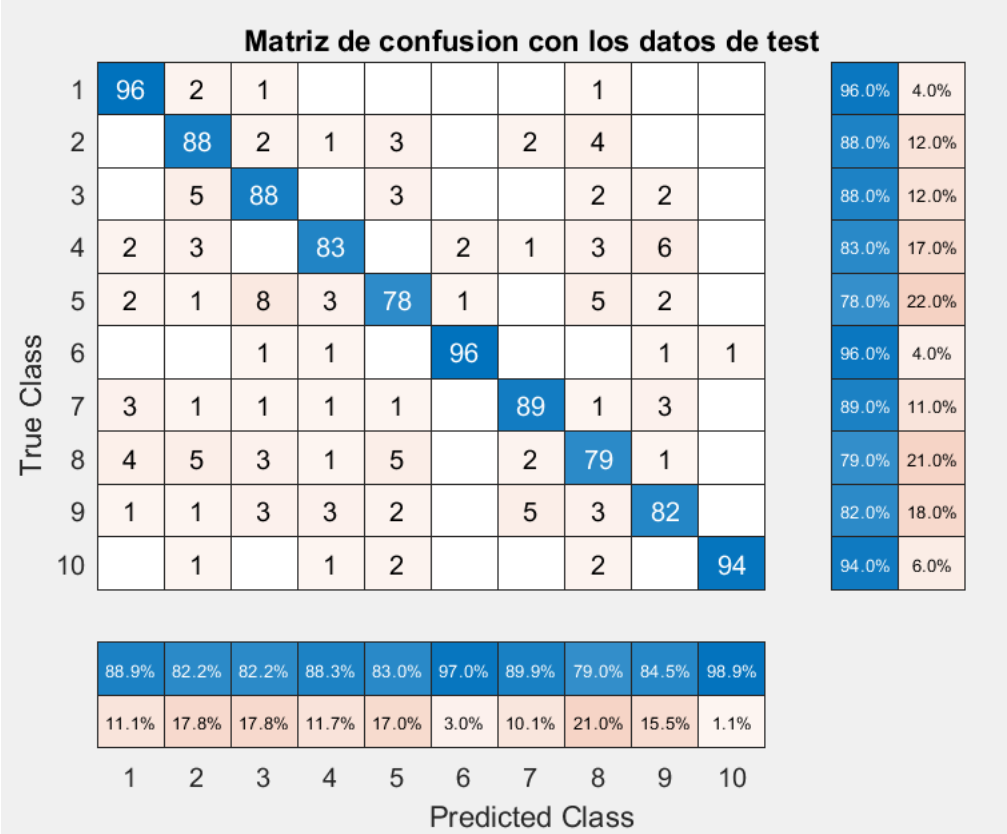

Figura 6: Matriz de confusión con los datos de test

Con los datos de test, el resultado es similar: una mayoría de buenas predicciones. Sin embargo, con este conjunto de datos, nuestro modelo tuvo más dificultades para adivinar la clase 5 y se equivocó 8 veces al predecir la clase 3.

En cuanto al estudio de la precisión (valores que resumen las columnas), destaca las dificultades del modelo para predecir la clase correcta y se equivoca proponiendo el número 8 (79 % de precisión). En cuanto al estudio del recall, destaca las dificultades del modelo para predecir la clase 5 (78 % de buenas predicciones del número 5) y la clase 8 (79 % de buenas predicciones del número 8).

Conclusión

Para concluir, esta práctica nos ha permitido ampliar los conocimientos adquiridos en la práctica anterior. De hecho, después de haber sido capaz de predecir si pertenecemos o no a una clase (problema binario), ahora somos capaces de predecir la pertenencia a una clase entre tantas otras. Pudimos entender las diferencias en los métodos de minimización de funciones (LBFGS y Newton). Por último, hemos aprendido a diseñar una matriz de confusión que permite comprender mejor los errores cometidos y su frecuencia respectiva. Así, pudimos analizar los puntos fuertes y débiles de nuestro modelo.

Estos conocimientos nos serán útiles en el resto del curso y en nuestra vida de ingeniero, ya que los problemas de clasificación están omnipresentes en la vida cotidiana, sobre todo con la aportación de nuevos datos que analizar.

Índice de figuras

Índice de cuadros