# Trabajo 3: Programación

Patricia Córdoba Hidalgo

# Índice



# 1. Problema de clasificación

Este problema consiste en clasificar imágenes de dígitos escritos a mano, asignándole a cada una el dígito que representan. Nuestro espacio de características, $\chi$, está formado por datos de 64 características, cada una de ellas representando la intensidad de trazado de una de las casillas de una matriz $8 \times 8$, donde se ha trazado el dígito. El conjunto de etiquetas, $Y$, son los dígitos del 0 al 9. La función desconocida $f$ es aquella que a cada imagen (elemento de $\chi$) le asigna el dígito que representa (su etiqueta correspondiente).

Primero leemos los ficheros de datos con la función `readData`, que separa los datos de sus etiquetas. Los datos del fichero test serán los datos que usaremos para calcular el $E_{out}$ y los del fichero training los dividimos en datos de entrenamiento y datos de validación. Un 25 % de los datos del fichero de datos training serán usados para validar. En un problema de aprendizaje corriente lo ideal sería mezclar los datos de ambos ficheros, ya que normalmente nos dan una base de datos y no tenemos un conjunto test, y dividir de ahí los conjuntos de entrenamiento, validación y test. En este caso, esto no es plausible, ya que los datos son números dibujados por personas y si mezclamos los ficheros existe la posibilidad de que haya dígitos trazados por la misma persona tanto en el conjunto training como en el test. Si esto ocurriese, como el algoritmo se entrenó para adaptarse a la caligrafía de ese individuo, la estimación del error fuera de la muestra que nos porporciona el conjunto test no sería realista. Por esto, en este caso, no mezclamos los ficheros, dejamos que el fichero test se use como conjunto test.

Las clases de funciones que usaré para resolver el problema son los polinomios de grado uno y dos. Usaré estas clases de funciones porque son relativamente sencillas y no tienen tanta variabilidad como otras clases de funciones, así no habrá tanta tendencia al sobreajuste y el error $E_{out}$ no será tan alto.

Antes de empezar a trabajar con los datos, se preprocesan. Primero se usa PCA, para deshechar aquellas características que no aporten mucha información al modelo, y nos quedamos con aquellas capaces de explicar el 99 % de la distribución. Tras esto, escalamos los datos, ya que, si una característica tiene una varianza varios órdenes mayor que otra, puede tener repercusiones en el cálculo de la función objetivo y puede hacer que no se aprenda correctamente del resto de características.
Estas transformaciones se ajustan a los datos de entrenamiento (con la función `fit`) y luego se usan las mismas sobre el resto de datos (tanto al conjunto de validación como al de test).

Para clasificar los datos, usaremos regresión logística multietiqueta. Por el teorema de "No-Free-Lunch" todos los algoritmos son iguales en media, no hay un algoritmo mejor que otro, hay problemas donde unos tienen mejor desempeño que otros y elegí usar este algoritmo inicialmente porque ya lo hemos trabajado en otras prácticas, es sencillo y hemos visto como usarlo en caso de clasificación multietiqueta. Los resultados obtenidos fueron buenos, por lo que concluí que era un algoritmo con buen desempeño en éste problema.

La implementación del algoritmo de *Regresión Logistica* está en la función `rl_sgd`. La función `rl_sgd` devuelve una lista de 10 vectores, cada uno de ellos calculado en una de las iteraciones del bucle principal (usando gradiente descendente estocástico y el error visto en teoría), que corresponde a los pesos de la función que separa esa clase del resto.
Antes de ejecutar esta función, es necesario ajustar el formato de las etiquetas, pasando de tener dígitos a tener

vectores de 10 coordenadas. A la etiqueta que representa el dígito $i$ se le asocia el vector $e_i$, aquel que tiene ceros en todas sus componentes menos en la posición $i$, que tiene un 1. La etiqueta 0 pasa a ser [1 0 0 0 0 0 0 0 0 0], la etiqueta 1 pasa a ser [0 1 0 0 0 0 0 0 0 0], y así sucesivamente, hasta la etiqueta 9 que pasa a ser [0 0 0 0 0 0 0 0 0 1].

El error que minimizamos usando el algoritmo de *Regresión Logistica* es la pérdida logarítmica, que nos permite penalizar mucho cuando la estimación es errónea y que el error sea muy cercano a cero cuando la estimación es buena. Sin embargo, la métrica que usaremos para elegir el mejor modelo será la precisión (accuracy), que es el porcentaje de etiquetas que acierta nuestro modelo. Usaré esta métrica para ver realmente cuál es el desempeño del modelo y cómo de bien clasifica las etiquetas.

Los resultados obtenidos son:

| | **CARACTERÍSTICAS LINEALES** | **CARACTERÍSTICAS CUADRÁTICAS** |
|---|---|---|
| Error del conjunto training | 0.06822748834535933 | 0.005245633985463137 |
| Error conjunto de validación | 0.08416293421735029 | 0.03036210074385761 |
| Error del conjunto test | 0.07509996756431692 | 0.06367515319684557 |
| Precisión del conjunto training | 0.7900976290097629 | 0.939679218967922 |
| Precisión conjunto de validación | 0.7979057591623037 | 0.8680628272251308 |

El modelo con características cuadráticas consigue una mayor precisión que el modelo con características lineales, es decir, consigue asignar la etiqueta correcta al mayor número de datos. El error dentro de la muestra es considerablemente menor que con características lineales. Esto se debe a que la clase de funciones es más rica y puede ajustarse mejor a los datos, aunque en el proceso puede producirse sobreajuste. Es por esto que la regularización es importante, para evitar que el modelo se ajuste demasiado bien a los datos de entrenamiento y no tan bien a los datos de la distribución.

**********
Nosotros, que poseemos un conjunto de test, podemos ver que realmente el modelo con características cuadráticas se ajusta mejor a los datos de fuera de la muestra, pero esta información no debería impulsarnos a tomar nunguna decisión ya que, en teoría, esta información no la poseemos.
**********

Para comprobar que no hemos sobreajustado el modelo a los datos y para limitar la varianza de la clase, que influye en el error $E_{out}$, es muy importante hacer siempre regularización. En mi caso, para regularizar usé la función `rl_reg` que usa el algoritmo de *Regresión Logistica* minimizando ahora el error $E_{aug} = E_{in} + \lambda||w||^2$, que es equivalente a imponer la restricción de que $||w||^2$ no supere cierta constante, y así acotamos la variabilidad de la clase. En el algoritmo, como calculamos cada uno de los 10 vectores de peso por separado, $w$ sí es un vector, pero en el cálculo del error de regularización, con `Err_reg`, $w$ es una matriz (contiene a los 10 vectores), luego lo que usamos es la norma de Frobenius, que es equivalente a la norma 2 por filas.
Los resultados obtenidos con la regularización son:

| | **CARACTERÍSTICAS LINEALES** | **CARACTERÍSTICAS CUADRÁTICAS** |
|---|---|---|
| Error del conjunto training | 0.11641129755974719 | 0.17732784788169068 |
| Error conjunto de validación | 0.2385966274134429 | 0.5587784527220588 |
| Error del conjunto test | 0.16734674140002598 | 0.3341999985046917 |
| Precisión del conjunto training | 0.8009065550906556 | 0.944560669456067 |
| Precisión conjunto de validación | 0.7853403141361257 | 0.8869109947643979 |

MODELO ELEGIDO
El error del conjunto de test es: 0.10848095502383011 La precisión del conjunto test es: 0.8336115748469671
EJEMPLO:
La etiqueta es: 0.0 La etiqueta que predice es: 0 Con probabilidad: 0.999999996657045 ******* La etiqueta es: 4.0 La etiqueta que predice es: 4 Con probabilidad: 0.9999261321009139 ******* La etiqueta es: 2.0 La etiqueta que predice es: 2 Con probabilidad: 0.9999999852647863 *******