Aprendizaje Automático
Trabajo 3: Programación
Ajuste de Modelos Lineales

Alfonso García Martínez
alfonsogmw@correo.ugr.es

Mayo de 2019

Índice

1. Introducción a los problemas

Nuestro objetivo en este trabajo consiste en ajustar modelos lineales en dos problemas de aprendizaje supervisado: uno de regresión y otro de clasificación.

En el problema de regresión se utiliza el conjunto de datos *Airfoil Self-Noise Data Set*, que consta de un total de 1503 instancias obtenidas a partir de una serie de tests realizados por la NASA en tubos de viento para medir la presión acústica (ruido) generada por turbinas con paletas con diferentes perfiles.

En el problema de clasificación se usa el conjunto de datos *Optical Recognition of Handwritten Digits*, de un total de 5620 instancias, que consisten en bitmaps de dígitos escritos a mano. Como es de imaginar, el problema consiste en predecir el dígito correspondiente $(0, 1, \ldots, 9)$.

Aunque solo emplearemos modelos lineales para ambos problemas, utilizaremos distintas técnicas de preprocesado de datos y de regularización para sacarles el máximo partido y ver hasta dónde podemos llegar con modelos tan simples.

2. Problema de regresión: *Airfoil Self-Noise*

2.1. Exploración del conjunto de datos

Lo primero que debe hacerse cuando nos enfrentamos a un problema es estudiarlo. En nuestro caso, trataremos de visualizar y explorar el conjunto de datos para intentar responder a una serie de cuestiones:

- ¿Qué características (variables) conforman los datos ($\mathcal{X}$)? ¿Qué etiquetas queremos intentar predecir ($\mathcal{Y}$)?
- ¿Existen valores perdidos de los que debamos ocuparnos?
- ¿Cómo se distribuyen los valores de las etiquetas?
- ¿Cómo se distribuyen los valores de las distintas variables de los datos? ¿Existen interdependencias entre características?

En este problema debemos predecir el ruido (medido como presión acústica) que produce una turbina según las siguientes características:

- La frecuencia de giro (Hz)
- El ángulo de ataque de las paletas (º)
- La longitud de acorde de las paletas (m)
- La velocidad de flujo libre (m/s)
- El grosor de desplazamiento lateral de succión (m)

Con la ayuda de funciones que aplican máscaras booleanas para detectar valores perdidos *NaN* (*Not a Number*), vemos que afortunadamente no hay ningún valor perdido en este conjunto de datos.

Imprimiendo un histograma de las etiquetas, podemos ver cómo se distribuyen los rangos de valores de $\mathcal{Y}$ (recordemos que esto es regresión).

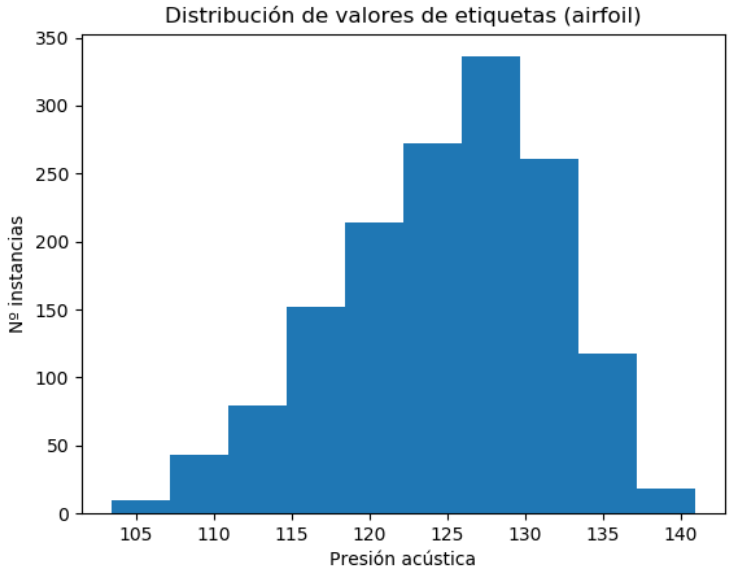

Figura 1: Histograma de *airfoil*

La media de las etiquetas es 124.836, y sus valores más repetidos (3 instancias) son 126.54, 127.315 y 129.395.

En base a lo obtenido, podemos decir que hay cierta variabilidad en las etiquetas, pero aún así parecen concentrarse entre los 120dB y los 130dB.

Veamos cómo se distribuyen las distintas variables o características para detectar dependencias entre ellas:

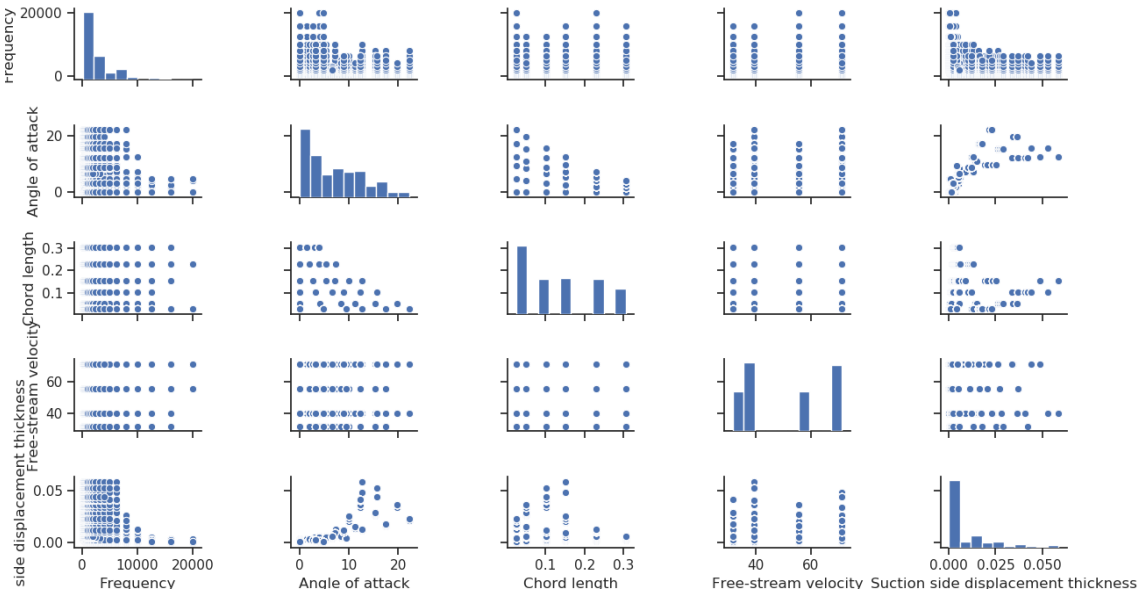

Figura 2: Explorando las variables de *airfoil*

En principio, no parece que las variables tengan poca varianza (salvo quizá el grosor de desplazamiento lateral de succión, cuyos valores son en su inmensa mayoría muy cercanos a 0), por lo que, en ese sentido, todas aportan información útil para determinar el valor de y.

Dos de las cinco variables, en concreto **la frecuencia y el grosor de desplazamiento lateral de succión**, aparentemente **se influyen algo mutuamente**, pero no debemos sacar conclusiones precipitadas. Quizá la matriz de correlación nos de más información sobre posibles interdependencias.

El coeficiente de correlación entre dos variables viene dada por la siguiente expresión:

$$\rho(X, Y) = \frac{\sigma_{XY}}{\sigma_X \sigma_Y}$$

Donde σ_{XY} es la covarianza, y σ_X y σ_Y son las desviaciones típicas de las variables.

Al igual que la covarianza, el coeficiente de correlación de dos variables nos da una idea de cómo varían una respecto a la otra para detectar posibles dependencias, pero el coeficiente de correlación está definido en el intervalo $[-1, +1]$, lo que hace que el rango de valores de las variables no influya en el valor obtenido.

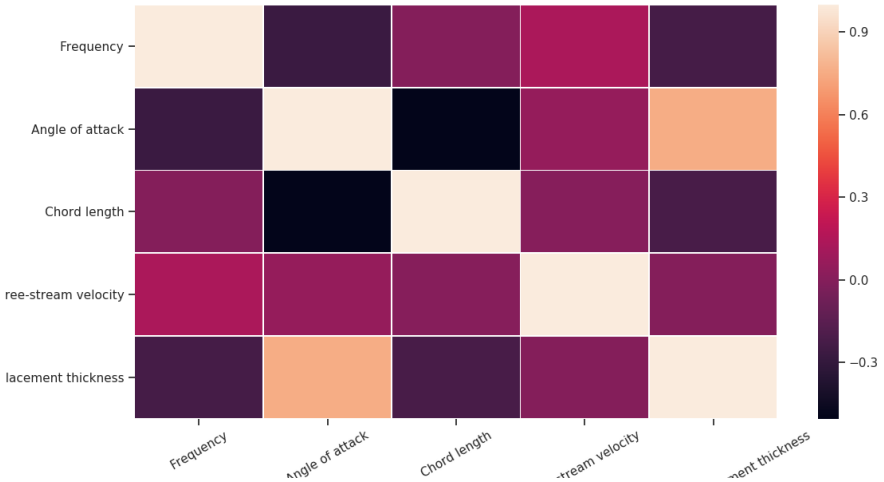

Figura 3: Matriz de correlación de *airfoil*

Tal y como sospechamos al ver la figura 2, sí que existe correlación entre el ángulo de ataque y el grosor de desplazamiento lateral de succión. A juzgar por el mapa de color, la correlación entre esas dos características se sitúa entre 0.6 y 0.9, siendo 1 la máxima correlación posible.

Por otra parte, la longitud de acorde y el ángulo de ataque también parecen estar (inversamente) correlacionadas, pero de forma mucho menos acusada.

2.2. Primer intento: métricas utilizadas y regresión lineal simple

Usaremos en primer lugar la regresión lineal tal cual, sin más añadidos, y a partir de ahí intentaremos mejorar los resultados. Pero para ello, hay que determinar qué métrica/s se usarán para saber cómo rinde nuestro regresor.

En regresión, una medida muy típica es el error cuadrático medio o MSE (*Mean Squared Error*):

$$MSE(h) = \frac{1}{N}\sum_{n=1}^{N}(h(x_n) - y_n)^2$$

Pero existen otras métricas de rendimiento además de esta, como el error absoluto medio o MAE (*Mean Absolute Error*):

$$MAE(h) = \frac{1}{N}\sum_{n=1}^{N}|h(x_n) - y_n|$$

Es cierto que el MSE es usado en regresión porque es fácil obtener su derivada para minimizarlo, tal y como se hace en SGD. Pero además, esta medida es también de utilidad porque es muy sensible a los errores: como los eleva al cuadrado, penaliza mucho las diferencias grandes y, a su vez, mitiga o aminora las pequeñas (las diferencias menores que 1).

Por su parte, el error absoluto no aplica ninguna modificación al error aparte de cambiar su signo cuando es negativo (de otra forma, los datos que se pasan de largo *cancelarían* a los que se quedan cortos, pues al sumar errores positivos con errores negativos el error promedio estaría más cerca de 0 de lo que debería).

Por ese mismo motivo, MAE nos da una idea más *realista* e inmediata de cuánto hemos errado al predecir un valor, aunque eso no significa que el MSE sea menos válido o menos útil. Son dos métricas hasta cierto punto similares, pero que nos informan de distinta forma de cuánto hemos fallado.

Ahora que por fin tenemos dos métricas para medir el rendimiento de nuestros modelos, probemos la regresión lineal sobre todo este conjunto de datos, sin aplicar ninguna transformación, ningún procesamiento ni ninguna regularización sobre el modelo.

Regresión lineal para *airfoil*		
Modelo	MSE	MAE
Regresión lineal simple	23.033	3.729

No está mal para una primera aproximación y para ser un modelo tan sencillo como lo es el lineal, pero recordemos que estamos probando nuestro modelo con el mismo conjunto de datos con el que lo hemos entrenado.

2.3. Mecanismo de validación y conjunto de test

Ya sabemos lo poco útil que es medir nuestro modelo con los mismos datos con los que lo hemos entrenado, pues de esta forma nunca conoceremos la capacidad de generalización del modelo. Este es un error básico que siempre hay que evitar y para el que existen soluciones: los mecanismos de validación.

Un método de validación muy conocido y muy ampliamente usado es la **validación cruzada** o CV (*Cross Validation*). Consiste, a grandes rasgos, en hacer una partición de k trozos de nuestro conjunto de datos, de forma que en cada una de las k iteraciones, uno de los trozos se utiliza para medir o validar nuestro modelo y el resto se usa para entrenarlo. El resultado final que obtiene este método es el promedio de los resultados de todas las iteraciones. Típicamente suele usarse un $k = 5$ para tener en cada iteración un 20 % de los datos para validar y el 80 % para entrenar. Para que cada trozo sea lo más representativo posible, se desordenan los datos usando una.

Lógicamente, los métodos de validación se aplican solo sobre en conjunto de datos que disponemos para entrenar el modelo. En los problemas reales, el modelo que construyamos será aplicado a datos con los que jamás habremos trabajado, y eso es justo lo que intentaremos emular en este trabajo reservando un 20 % de los datos para *test*, de forma que el error de *training* y el de validación se medirán únicamente con el 80 % restante.

Con todo esto, ahora sí podemos medir más fielmente la capacidad de nuestro modelo:

Regresión lineal para *airfoil* - *Training*		
Modelo	MSE_{Train}	MAE_{Train}
Regresión lineal simple	22.509	3.674

Regresión lineal para *airfoil* - Validación		
Modelo	MSE_{CV}	MAE_{CV}
Regresión lineal simple	22.988	3.708

Regresión lineal para *airfoil* - *Test*		
Modelo	MSE_{Test}	MAE_{Test}
Regresión lineal simple	25.282	3.886

Como es de esperar, el error más bajo de todos es el de *training*, pero aún así los errores de *test* y validación no son mucho más altos. Esto es buena señal, pues significa que nuestro modelo no se sobreajusta, sino que tiene suficiente capacidad de generalización.

2.4. Primeros ajustes: normalización de datos

A partir de ahora, nuestra tarea se enfocará en la obtención del mejor preprocesado y el mejor modelo posible, ajustando los parámetros y probando diferentes técnicas. Por tanto, **usaremos el conjunto de test solo para evaluar el modelo final que obtengamos**. Hasta entonces, trabajaremos únicamente con el conjunto de training para probar y validar los sucesivos modelos que vayamos construyendo y ajustando.

Lo primero que probaremos para mejorar los resultados será el preprocesamiento de los datos. En muchas ocasiones, un correcto tratamiento de los datos sirve para facilitarle el trabajo a los modelos predictivos para que ofrezcan mejores resultados.

Muchas de las técnicas de preprocesamiento consisten en reducir y simplificar el conjunto de datos (selección de instancias y selección de características), otras consisten en eliminar ruido , otras en asignar valores a los datos perdidos (cosa de la que afortunadamente no hay que preocuparse en este caso), y otras consisten en normalizar los datos para que los distintos rangos de valores de las variables no influyan.

Comenzaremos usando una **normalización de reescalado**, que consiste en calcular los valores máximos y mínimos de cada variable, de forma que se utilicen para transformar todos los valores originales en nuevos valores comprendidos en el intervalo $[0, 1]$. Al aplicar este reescalado (únicamente sobre los datos $\mathcal{X}$), los errores no varían al volver a ajustar una regresión lineal, pero ahora tenemos garantía de que los rangos de valores no alterarán los resultados al hacer otro tipo de preprocesados y ajustes.

Regresión lineal para *airfoil* - *Training*		
Modelo	MSE_{Train}	MAE_{Train}
Regresión lineal simple	22.509	3.674
Regresión lineal + Normalización	22.509	3.674

Regresión lineal para *airfoil* - Validación		
Modelo	MSE_{CV}	MAE_{CV}
Regresión lineal simple	22.988	3.708
Regresión lineal + Normalización	22.988	3.708

2.5. Probando selección de características

En un problema con datos de alta dimensionalidad, donde se trabaja con un gran número de características, el modelo empleado debe ser muy complejo para pueda *dar la talla* y que el error pueda minimizarse lo suficiente. Por otra parte, lo que siempre se pretende es lograr siempre la mayor capacidad de generalización posible, y para ello suelen usarse técnicas para reducir los datos y que se puedan usar modelos más simples. Por ejemplo, una forma de reducir o simplificar los datos es la selección de características, con la cual eliminamos algunas variables y mantenemos el resto para entrenar al modelo.

Sin embargo, no todo es de color rosa: si no tenemos cuidado al eliminar características, probablemente estaremos eliminando variables que nos aportan información útil para la predicción. Es por ello que las características que no se conservan son aquellas con poca varianza (se mantienen siempre en valores muy similares, sin que varíen demasiado para distintos valores de y) o que varían de una forma similar a como lo hace otra variable (tiene una covarianza o coeficiente de correlación altos).

Probaremos la primera opción: eliminaremos la característica con menor varianza.

Varianzas de las variables de *airfoil*	
Característica	Varianza
Frequency	0.0255
Angle of attack	0.0728
Chord length	0.1103
Free-stream velocity	0.1533
Suction side displacement thickness	0.0532

La frecuencia es la característica con menos varianza. Aplicaremos entonces un filtro de varianza sobre los datos con un umbral ligeramente superior a la varianza mínima para eliminar momentaneamente esa variable.

Regresión lineal para *airfoil - Training*		
Modelo	MSE_{Train}	MAE_{Train}
Regresión lineal simple	22.509	3.674
Regresión lineal + Normalización	22.509	3.674
Regresión lineal + Normalización + Elimin. característica menor varianza	37.514	4.975

Regresión lineal para *airfoil* - Validación		
Modelo	MSE_{CV}	MAE_{CV}
Regresión lineal simple	22.988	3.708
Regresión lineal + Normalización	22.988	3.708
Regresión lineal + Normalización + Elimin. característica menor varianza	37.752	4.992

Los errores han aumentado notablemente. No parece haber sido buena idea eliminar esa variable, y no tiene sentido que sigamos eliminando variables por ese criterio

Si recordamos lo que vimos en las figuras 2 y 3, existia cierta correlación entre el ángulo de ataque y el grosor de desplazamiento lateral de succión. Probaremos a eliminar del conjunto de datos original (normalizado) primero el ángulo y después el grosor.

Regresión lineal para *airfoil* - *Training*		
Modelo	MSE_{Train}	MAE_{Train}
Regresión lineal simple	22.509	3.674
Regresión lineal + Normalización	22.509	3.674
Regresión lineal + Normalización + Elimin. característica menor varianza	37.514	4.975
Regresión lineal + Normalización + Elimin. ángulo ataque	24.539	3.841
Regresión lineal + Normalización + Elimin. grosor	24.036	3.847

Regresión lineal para *airfoil* - Validación		
Modelo	MSE_{CV}	MAE_{CV}
Regresión lineal simple	22.988	3.708
Regresión lineal + Normalización	22.988	3.708
Regresión lineal + Normalización + Elimin. característica menor varianza	37.752	4.992
Regresión lineal + Normalización + Elimin. ángulo ataque	24.947	3.868
Regresión lineal + Normalización + Elimin. grosor	24.531	3.881

En definitiva, la selección de características no ha funcionado, y no puede decirse que sea de extrañar: nuestro problema no tiene una dimensionalidad alta, y al quitar variables en este caso estamos eliminando información útil para determinar el valor de la etiqueta.

2.6. Análisis de Componentes Principales

El análisis de componentes principales o PCA(*Principal Component Analisys*) también reduce la dimensionalidad de los datos, pero lo hace de una forma mucho más elaborada que la selección de características.

A grandes rasgos, consiste en reducir el conjunto de datos sustituyendo variables por otras no correlacionadas entre sí, llamadas **componentes principales**, que contienen gran parte de la información que tenían los datos originales. Para obtener las componentes principales, se debe aplicar una descomposición en valores singulares (SVD) a la matriz de covarianza de los datos para obtener, tras una serie de pasos, los valores y vectores propios de la matriz de covarianza y con ellos definir las componentes principales.

Como nuestro problema no posee una alta dimensionalidad, no podemos reducir mucho con PCA. Lo usaremos solo para generar 4 y 3 componentes.

Regresión lineal para *airfoil - Training*		
Modelo	MSE_{Train}	MAE_{Train}
Regresión lineal simple	22.509	3.674
Regresión lineal + Normalización	22.509	3.674
Regresión lineal + Normalización + Elimin. característica menor varianza	37.514	4.975
Regresión lineal + Normalización + Elimin. ángulo ataque	24.539	3.841
Regresión lineal + Normalización + Elimin. grosor	24.036	3.847
Regresión lineal + Normalización + PCA 4 comp.	23.455	3.754
Regresión lineal + Normalización + PCA 3 comp.	40.735	5.196

Regresión lineal para *airfoil* - Validación		
Modelo	MSE_{CV}	MAE_{CV}
Regresión lineal simple	22.988	3.708
Regresión lineal + Normalización	22.988	3.708
Regresión lineal + Normalización + Elimin. característica menor varianza	37.752	4.992
Regresión lineal + Normalización + Elimin. ángulo ataque	24.947	3.868
Regresión lineal + Normalización + Elimin. grosor	24.531	3.881
Regresión lineal + Normalización + PCA 4 comp.	23.872	3.784
Regresión lineal + Normalización + PCA 3 comp.	40.977	5.214

Seguimos sin suerte: el PCA no consigue mejorar los resultados.

2.7. Transformación no lineal: polinómica de orden desde 2 hasta 5

Hasta ahora hemos probado a reducir el número de características, pero no hemos tenido éxito. Probemos ahora justo lo contrario: ampliaremos la dimensionalidad del problema añadiendo monomios de hasta cierto grado para ver si en ese nuevo espacio los datos pueden predecirse mejor con un modelo lineal.

La transformación consiste en añadir todas las posibles combinaciones de variables que generen monomios de hasta el orden dado (aunque existen opciones para que solo se añadan interacciones, es decir, que solo haya monomios cuyas variables tengan exponente igual a 1). En particular, si el orden de los monomios es igual a 2, y tenemos datos de dos variables, la transformación sería la siguiente:

$$(a, b) \mapsto (1, a, b, a^2, ab, b^2)$$

Veamos qué resultados obtenemos si utilizamos transformaciones polinómicas de órdenes desde el 2 y hasta el 5.

Regresión lineal para *airfoil - Training*		
Modelo	MSE_{Train}	MAE_{Train}
Regresión lineal simple	22.509	3.674
Regresión lineal + Normalización	22.509	3.674
Regresión lineal + Normalización + Elimin. característica menor varianza	37.514	4.975
Regresión lineal + Normalización + Elimin. ángulo ataque	24.539	3.841
Regresión lineal + Normalización + Elimin. grosor	24.036	3.847
Regresión lineal + Normalización + PCA 4 comp.	23.455	3.754
Regresión lineal + Normalización + PCA 3 comp.	40.735	5.196
Regresión lineal + Normalización + Transf. polinómica orden 2	16.725	3.145
Regresión lineal + Normalización + Transf. polinómica orden 3	11.168	2.572
Regresión lineal + Normalización + Transf. polinómica orden 4	6.483	1.898
Regresión lineal + Normalización + Transf. polinómica orden 5	4.316	1.517

Regresión lineal para *airfoil* - Validación		
Modelo	MSE_{CV}	MAE_{CV}
Regresión lineal simple	22.988	3.708
Regresión lineal + Normalización	22.988	3.708
Regresión lineal + Normalización + Elimin. característica menor varianza	37.752	4.992
Regresión lineal + Normalización + Elimin. ángulo ataque	24.947	3.868
Regresión lineal + Normalización + Elimin. grosor	24.531	3.881
Regresión lineal + Normalización + PCA 4 comp.	23.872	3.784
Regresión lineal + Normalización + PCA 3 comp.	40.977	5.214
Regresión lineal + Normalización + Transf. polinómica orden 2	17.473	3.199
Regresión lineal + Normalización + Transf. polinómica orden 3	12.493	2.715
Regresión lineal + Normalización + Transf. polinómica orden 4	8.875	2.180
Regresión lineal + Normalización + Transf. polinómica orden 5	14.716	2.141

La transformación polinómica ha dado resultados sorprendentemente buenos. Mejorar la complejidad de nuestro modelo aumentando su dimensionalidad (21 dimensiones para orden 2, 56 para orden 3, 126 para orden 4 y 252 para orden 5) ha disminuido considerablemente el error, pero **al llegar al orden 5 la complejidad del modelo empieza a ser excesiva, lo que provoca un ligero sobreajuste**. Este sobreajuste puede apreciarse si nos fijamos en los errores de las transformaciones polinómicas de órdenes 4 y 5: al pasar a orden 5, el error MSE en *training* disminuye, pero no así el de validación, que en su lugar aumenta. Veamos qué puede hacerse para paliar este efecto.

2.8. Controlando el exceso de complejidad: Transformación no lineal + PCA

Si combinamos el aumento de complejidad de la transformación polinómica de orden 5 con la reducción del PCA, quizá obtengamos lo mejor de ambas partes: los errores bajos de la transformación polinómica pero paliando el sobreajuste con la reducción.

El PCA que aplicaremos concretamente hará que la dimensionalidad de los datos transformados se reduzca a 100 (parecido al de la transformación polinómica de orden 4, con 120 dimensiones), para ver si conseguimos mejores resultados en validación.

Regresión lineal para *airfoil - Training*		
Modelo	MSE_{Train}	MAE_{Train}
Regresión lineal simple	22.509	3.674
Regresión lineal + Normalización	22.509	3.674
Regresión lineal + Normalización + Elimin. característica menor varianza	37.514	4.975
Regresión lineal + Normalización + Elimin. ángulo ataque	24.539	3.841
Regresión lineal + Normalización + Elimin. grosor	24.036	3.847
Regresión lineal + Normalización + PCA 4 comp.	23.455	3.754
Regresión lineal + Normalización + PCA 3 comp.	40.735	5.196
Regresión lineal + Normalización + Transf. polinómica orden 2	16.725	3.145
Regresión lineal + Normalización + Transf. polinómica orden 3	11.168	2.572
Regresión lineal + Normalización + Transf. polinómica orden 4	6.483	1.898
Regresión lineal + Normalización + Transf. polinómica orden 5	4.316	1.517
Regresión lineal + Normalización + Transf. polinómica orden 5 + PLA	8.326	2.193

Regresión lineal para *airfoil* - Validación		
Modelo	MSE_{CV}	MAE_{CV}
Regresión lineal simple	22.988	3.708
Regresión lineal + Normalización	22.988	3.708
Regresión lineal + Normalización + Elimin. característica menor varianza	37.752	4.992
Regresión lineal + Normalización + Elimin. ángulo ataque	24.947	3.868
Regresión lineal + Normalización + Elimin. grosor	24.531	3.881
Regresión lineal + Normalización + PCA 4 comp.	23.872	3.784
Regresión lineal + Normalización + PCA 3 comp.	40.977	5.214
Regresión lineal + Normalización + Transf. polinómica orden 2	17.473	3.199
Regresión lineal + Normalización + Transf. polinómica orden 3	12.493	2.715
Regresión lineal + Normalización + Transf. polinómica orden 4	8.875	2.180
Regresión lineal + Normalización + Transf. polinómica orden 5	14.716	2.141
Regresión lineal + Normalización + Transf. polinómica orden 5 + PLA	12.020	2.474

Tal y como esperábamos, el PCA ha conseguido reducir algo el MSE de validación, pero no lo suficiente. Probaremos una última técnica más para mejorar en validación.

2.9. Controlando el exceso de complejidad: Transformación no lineal + Regularización

Nuestro intento previo no ha tenido éxito. Es el momento de hacer uso de la **regularización**.

La regularización consiste, a grandes rasgos, en *limitar* nuestro modelo para controlar su complejidad, en un intento por evitar el sobreajuste. Las técnicas de regularización que vamos a usar, llamadas *Ridge* (o L2) y *Lasso* (L1) se basan en el *weight decay*, que consiste en penalizar los modelos con pesos grandes, en función de sus valores al cuadrado en el caso del L2 o de sus valores absolutos en el caso de L1.

Ambas variantes reciben un parámetro α, que controla la intensidad de la regularización. Ajustaremos los valores de este parámetro para tratar de encontrar el nivel adecuado de regularización para los datos originales normalizados y los datos con transformación polinómica de órdenes 4 y 5, que son los que mejores resultados han ofrecido hasta ahora.

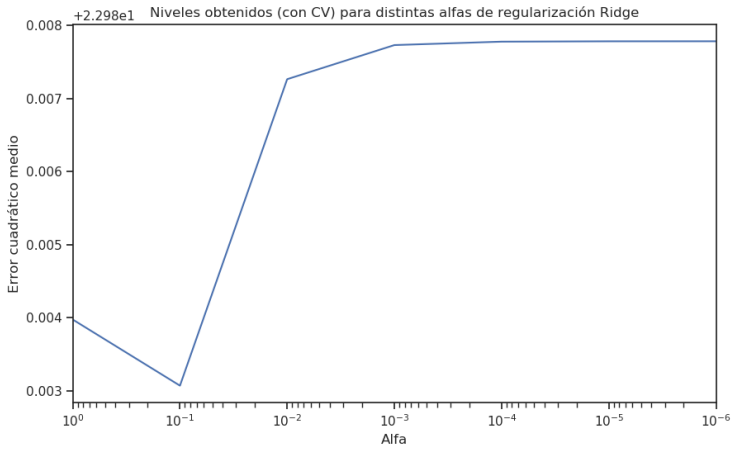

Figura 4: Regularización L2 con datos originales

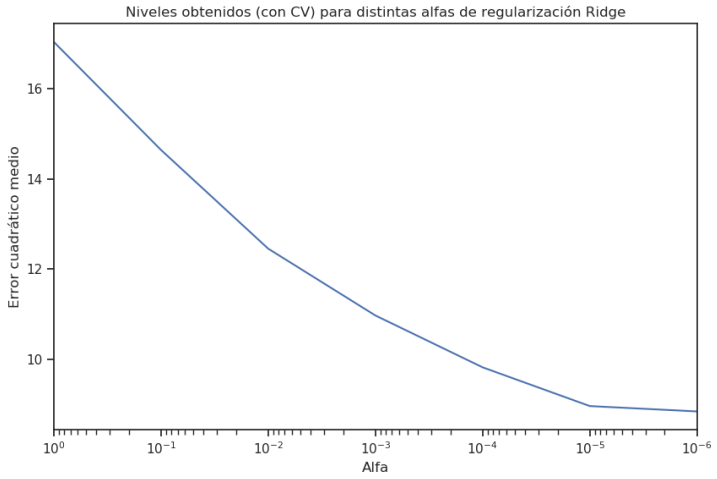

Figura 5: Regularización L2 con transf. polinómica orden 4

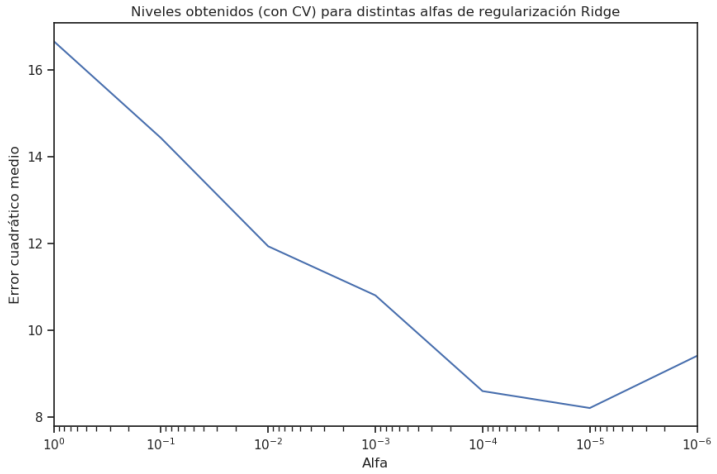

Figura 6: Regularización L2 con transf. polinómica orden 5

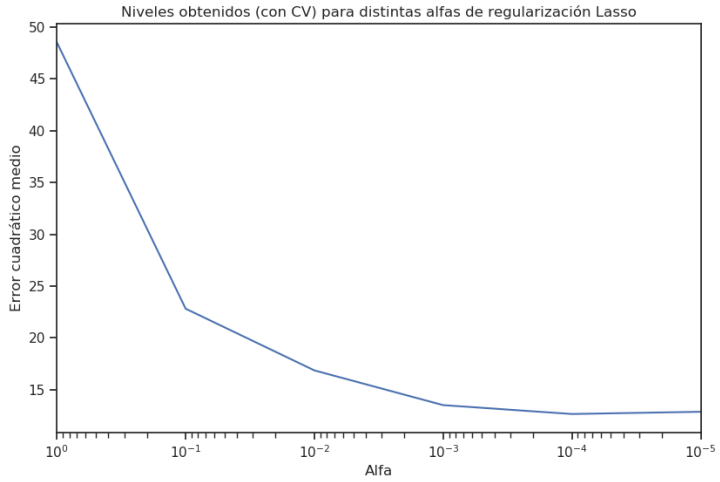

Figura 7: Regularización L1 con transf. polinómica orden 4

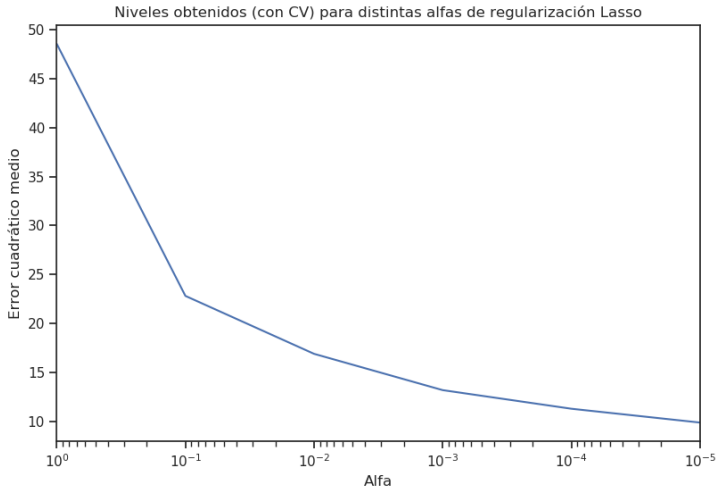

Figura 8: Regularización L1 con transf. polinómica orden 5

Por lo visto, la única regularización que ha funcionado ha sido la L2, con un $\alpha = 0{,}1$ para los datos originales (la mejora es ínfima), y con un $\alpha = 10^{-5}$ para la transformación polinómica de órden 5. Midamos más a fondo el desempeño de esta última combinación.

Regresión lineal para *airfoil - Training*		
Modelo	MSE_{Train}	MAE_{Train}
Regresión lineal simple	22.509	3.674
Regresión lineal + Normalización	22.509	3.674
Regresión lineal + Normalización + Elimin. característica menor varianza	37.514	4.975
Regresión lineal + Normalización + Elimin. ángulo ataque	24.539	3.841
Regresión lineal + Normalización + Elimin. grosor	24.036	3.847
Regresión lineal + Normalización + PCA 4 comp.	23.455	3.754
Regresión lineal + Normalización + PCA 3 comp.	40.735	5.196
Regresión lineal + Normalización + Transf. polinómica orden 2	16.725	3.145
Regresión lineal + Normalización + Transf. polinómica orden 3	11.168	2.572
Regresión lineal + Normalización + Transf. polinómica orden 4	6.483	1.898
Regresión lineal + Normalización + Transf. polinómica orden 5	4.316	1.517
Regresión lineal + Normalización + Transf. polinómica orden 5 + PLA	8.326	2.193
Regresión lineal con regul. L2 + Normalización + Transf. polinómica orden 5	5.203	1.670

Regresión lineal para *airfoil* - Validación		
Modelo	MSE_{CV}	MAE_{CV}
Regresión lineal simple	22.988	3.708
Regresión lineal + Normalización	22.988	3.708
Regresión lineal + Normalización + Elimin. característica menor varianza	37.752	4.992
Regresión lineal + Normalización + Elimin. ángulo ataque	24.947	3.868
Regresión lineal + Normalización + Elimin. grosor	24.531	3.881
Regresión lineal + Normalización + PCA 4 comp.	23.872	3.784
Regresión lineal + Normalización + PCA 3 comp.	40.977	5.214
Regresión lineal + Normalización + Transf. polinómica orden 2	17.473	3.199
Regresión lineal + Normalización + Transf. polinómica orden 3	12.493	2.715
Regresión lineal + Normalización + Transf. polinómica orden 4	8.875	2.180
Regresión lineal + Normalización + Transf. polinómica orden 5	14.716	2.141
Regresión lineal + Normalización + Transf. polinómica orden 5 + PLA	12.020	2.474
Regresión lineal con regul. L2 + Normalización + Transf. polinómica orden 5	8.217	1.996

2.10. Modelo final: cálculo de error en *test* y estimación del error fuera de la muestra

Llegó el momento de decidir cuál es el mejor modelo para que sea medido ante el conjunto de *test* (a parte del primer intento efectuado como prueba).

La regresión lineal con regularización L2 para un $\alpha = 10^{-5}$, datos normalizados y con una transformación polinómica de orden 5 es el modelo que menor error presenta en la fase de validación, y por ello es el mejor modelo entre todos los que hemos generado. Con él, obtenemos:

- Un MSE en *test* de 7.991
- Un MAE en *test* de 2.076

A partir del error en *test*, podemos tratar de estimar sencillamente el error fuera de este conjunto de datos:

$$E_{\text{out}} \leq E_{\text{test}} + \sqrt{\frac{1}{2N} \ln \frac{2|\mathcal{H}|}{\delta}}$$

Cogiendo el error MSE, y teniendo en cuenta que $N = 301$, que $|\mathcal{H}| = 1$ por estar en test, y eligiendo un $\delta = 0{,}05$, obtenemos que:

$$E_{\text{out}} \leq 8{,}0698$$

En conclusión: el problema, que en un principio no respondía muy bien a modelos lineales, ha ofrecido buenos resultados al aplicársele la transformación polinómica. A esto hay que sumarle la regularización L2 para controlar la alta complejidad que se ha generado con esa misma transformación, con lo que, en conjunto, se ha obtenido un buen equilibrio que permite una buena capacidad de generalización.

3. Problema de clasificación: *Optical Recognition of Handwritten Digits*

3.1. Exploración del conjunto de datos

Tal y como hemos comentado, este problema consiste en predecir a qué dígito decimal (del 0 al 9) corresponde un dato escrito a mano, representado por un bitmap. Las etiquetas $\mathcal{Y}$ son dígitos binarios y los datos $\mathcal{X}$ corresponden a vectores de 64 enteros entre 0 y 16, que corresponden a valores obtenidos tras aplicar una convolución sobre los píxeles de la imagen.

Comencemos a explorar e intentar visualizar el conjunto de *training*:

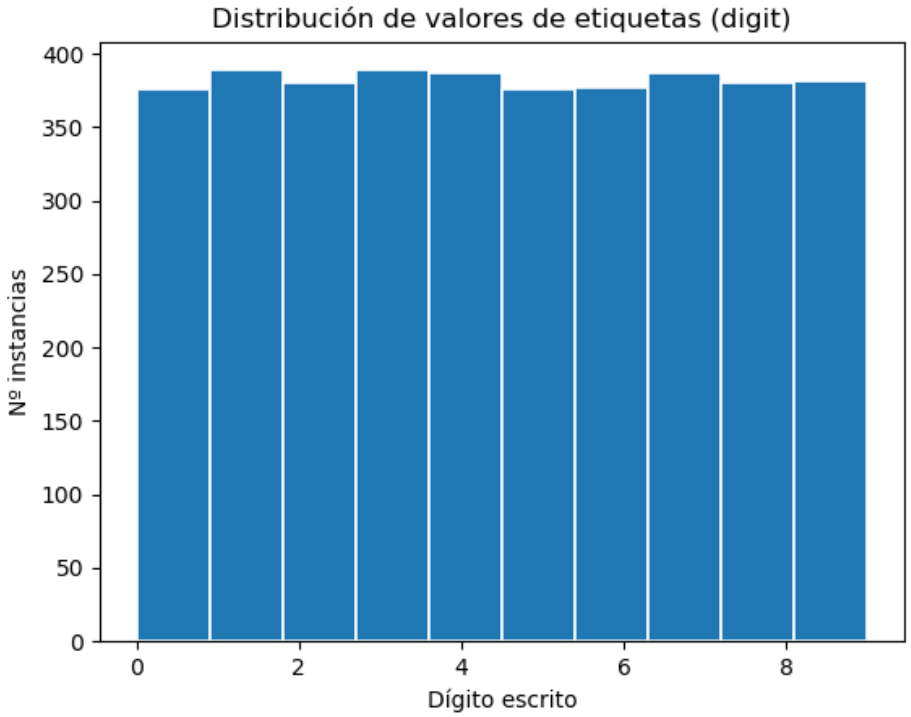

Figura 9: Distribución de valores de las etiquetas de *digits*

Tal y como puede verse en el histograma, los valores de las etiquetas $\mathcal{Y}$ están perfectamente balanceadas. Lo tendremos en cuenta más adelante cuando decidamos el método de validación.

Como este problema es de una dimensionalidad mucho mayor que el de regresión, va a ser bastante más complicado explorar los datos. Por ejemplo, es impensable ver las distribuciones de todas las variables dos a dos como hicimos en el problema de regresión.

Es difícil detectar posibles interdepenencias, pero sí que se aprecia una varianza casi nula en algunas variables que casi siempre valen 0.

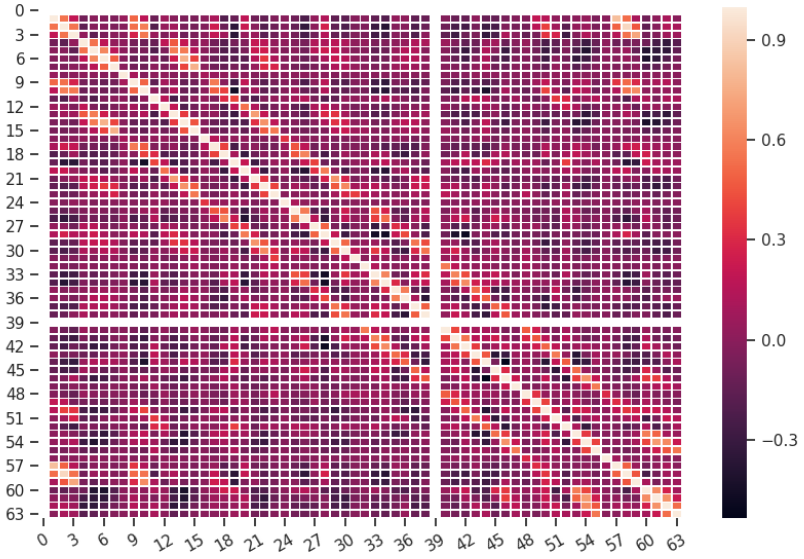

Figura 10: Matriz de correlación de *digits*

Obviando la *cruz blanca* que hay presente en la variable 39 por tener todos los valores a 0, en general la mayoría de variables no están relacionadas, aunque sí que se detectan algunos focos de correlación, como los hay entre las variables 1 y 58, o entre las variables 6 y 14, por poner algunos ejemplos. Además, se aprecian dos líneas más clareadas de correlación, ambas paralelas a la diagonal principal (la correlación de una variable consigo misma siempre es 1). Quizá esto sea indicador de posibles simetrías en muchos de los dígitos escritos...

3.2. Validación y métricas de rendimiento

De forma similar a como hicimos en el problema anterior, usaremos validación cruzada para medir la capacidad de generalización de nuestro modelo. Como estamos en un problema de clasificación en el que, además, las etiquetas están perfectamente balanceadas, estamos en condiciones de usar **validación cruzada estratificada**, de forma que cada partición (estrato) sea lo más representativo posible de todo el conjunto de *training*.

Respecto a las medidas de rendimiento, debemos tener en cuenta que no es un problema de **clasificación binaria**, sino que tenemos 10 posibles etiquetas distintas en $\mathcal{Y}$. Por tanto, podremos usar métricas como la exactitud, y a partir de ella calcular el número de instancias mal clasificadas (error de la clasificación). También podemos usar un promedio ponderado de la medida F1 para cada caso binario de las etiquetas (es o no es un 0, es o no es un 1, es o no es un 3...).

La medida F1 es útil, porque sintetiza muy bien en un solo valor el desempeño de un clasificador. En el caso binario, viene dado por la media harmónica de la precisión (*precision*) y la exhaustividad (*recall*):

$$F1 = 2\frac{precision \cdot recall}{precision + recall}$$

Donde

$$precision = \frac{True\ Positives}{True\ Positives + False\ positives}$$

$$recall = \frac{True\ Positives}{True\ Positives + False\ negatives}$$

3.3. Primer intento: perceptrón simple

Hagamos un primer intento con el perceptrón, cuyo funcionamiento se basa en el SGD. Afortunadamente, la implementación que vamos a utilizar soporta problemas de clasificación con más de dos posibles etiquetas, como es el caso.

El perceptrón acepta dos parámetros: el número máximo de iteraciones, que hemos fijado en 50000, y la tolerancia (precisión), fijada en 10^{-14}. Lo que se persique con esta configuración es obtener el mejor ajuste posible en un tiempo razonable.

Clasificación lineal para *digits* - *Training*			
Modelo	Exactitud	Ejemplos mal clasificados	F1
Perceptrón	0.9649	134	0.9647

Clasificación lineal para *digits* - Validación			
Modelo	Exactitud	Ejemplos mal clasificados	F1
Perceptrón	0.9349	248	0.9349

Los resultados son prometedores, pues en validación ya superamos el 90 % de exactitud con un perceptrón sin ningún tipo de preprocesamiento.

3.4. Normalización y perceptrón simple

Siguiendo los pasos del problema anterior, probamos un reescalado entre 0 y 1 de los valores de las características.

Clasificación lineal para *digits* - *Training*			
Modelo	Exactitud	Ejemplos mal clasificados	F1
Perceptrón	0.9649	134	0.9647
Perceptrón + Normalización	0.9566	165	0.9564

Clasificación lineal para *digits* - Validación			
Modelo	Exactitud	Ejemplos mal clasificados	F1
Perceptrón	0.9349	248	0.9349
Perceptrón + Normalización	0.9529	179	0.9529

Con normalización tenemos peor desempeño en *training*, pero hay mejoría en validación.

3.5. Regresión logística

Aunque la regresión logística es un modelo lineal, no es exactamente un clasificador, sino un clasificador estadístico. La clasificación *per se* se hace una vez obtenidas las probabilidades.

Aunque la implemetación de regresión logística permite usar directamente regularización, en un principio no vamos a utilizarla. La tolerancia o precisión será ajustada a 10^{-8} para, de nuevo, obtener un buen ajuste en un tiempo razonable.

Otra cosa a tener en cuenta: debemos seleccionar un *solver* que soporte problemas con más de dos etiquetas. Por ejemplo, probaremos con el *newton-cg*. También se debe ajustar el parámetro *multi_class*, por ejemplo a *multinomial*.

Clasificación lineal para *digits* - *Training*			
Modelo	Exactitud	Ejemplos mal clasificados	F1
Perceptrón	0.9649	134	0.9647
Perceptrón + Normalización	0.9566	165	0.9564
Regresión logística + Normalización	0.9804	75	0.9804

Clasificación lineal para *digits* - Validación			
Modelo	Exactitud	Ejemplos mal clasificados	F1
Perceptrón	0.9349	248	0.9349
Perceptrón + Normalización	0.9529	179	0.9529
Regresión logística + Normalización	0.9691	117	0.9691

3.6. Regresión logística + reducción de dimensionalidad

Partiendo de la regresión logística, con la que hemos obtenido muy buenos resultados, vamos a probarla ahora eliminando características con poca varianza (menor que 0.05).

Clasificación lineal para *digits* - *Training*			
Modelo	Exactitud	Ejemplos mal clasificados	F1
Perceptrón	0.9649	134	0.9647
Perceptrón + Normalización	0.9566	165	0.9564
Regresión logística + Normalización	0.9804	75	0.9804
Regresión logística + Normalización + Eliminación caracteríticas varianza pequeña	0.9791	80	0.9791

Clasificación lineal para *digits* - Validación			
Modelo	Exactitud	Ejemplos mal clasificados	F1
Perceptrón	0.9349	248	0.9349
Perceptrón + Normalización	0.9529	179	0.9529
Regresión logística + Normalización	0.9691	117	0.9691
Regresión logística + Normalización + Eliminación caracteríticas varianza pequeña	0.9665	127	0.9665

3.7. Modelo final: cálculo de error en *test* y estimación del error fuera de la muestra

La regresión logística con normalización ha dado los mejores resultados en validación, luego ese será el modelo y procesado que utilizaremos para *test*:

- Exactitud en *test*: 0.9477
- Ejemplos mal clasificados en *test*: 93
- F1 en *test*: 0.9479

Ahora, si queremos saber cómo se comportará el algoritmo con datos de fuera de la muestra:

$$E_{\text{out}} \leq E_{\text{test}} + \sqrt{\frac{1}{2N} \ln \frac{2|\mathcal{H}|}{\delta}}$$

Cogiendo $E_{\text{test}} = (1 - exactitud)$, y teniendo en cuenta que $N = 1797$, que $|\mathcal{H}| = 1$ por estar en test, y eligiendo un $\delta = 0{,}05$, obtenemos que:

$$E_{\text{out}} \leq 0{,}0843$$

Si intentamos acotar con el error de *training* del mejor modelo en vez del de *test*, la cota es más difícil de calcular puesto que hay que tener en cuenta la dimensión VC de la clase de funciones (modelos lineales de 64 dimensiones, $d_{VC} = 65$):

$$E_{\text{out}} \leq E_{\text{train}} + \sqrt{\frac{8}{N} \ln \frac{4((2N)^{d_{VC}} + 1)}{\delta}}$$

Con $\delta = 0{,}05$, $d_{VC} = 65$, $N = 3823$ y $E_{\text{train}} = 1 - exactitud_{train}$, tenemos:

En conclusión: una normalización y una regresión logística con tolerancia ajustada han bastado para obtener muy buenos resultados, tanto en *training* y validación como en *test*. El problema tiene suficiente separabilidad como para lograr un buen desempeño con modelos lineales.

4. Correspondencia aproximada de apartados con guión de prácticas

1. Comprender el problema a resolver. → 1, 2.1, 3.1
2. Preprocesado los datos: datos categoricos, normalización, proyección, etc → 2.4, 2.5, 2.6, 2.7, 3.4, 3.6
3. Selección de clases de funciones a usar. → Se han usado únicamente modelos lineales, de distintas dimensionalidades
4. Definición de los conjuntos de training, validación y test usados en su caso. → 2.3, 3.1, 3.2
5. Discutir la necesidad de regularización y en su caso la función usada para ello. → 2.9
6. Definir los modelos a usar y estimar sus parámetros e hyperparámetros. → 2.2, 3.3, 3.5 (si no contamos ajuste de parámetros de los métodos de preprocesado)
7. Selección y ajuste modelo final. → 2.10, 3.7
8. Discutir la idoneidad de la métrica usada en el ajuste → 2.2, 3.2
9. Estimacion del error E_{out} del modelo lo más ajustada posible. → 2.10, 3.7
10. Discutir y justificar la calidad del modelo encontrado y las razones por las que considera que dicho modelo es un buen ajuste que representa adecuadamente los datos muestra → 2.10, 3.7

5. Bibliografía

Material citado y/o consultado:

- *Singular Value Decomposition and Principal Component Analysis*, Rasmus Elsborg Madsen, Lars Kai Hansen & Ole Winther, Febrero 2004
- Apuntes de Regularización del Prof. Fernando Berzal (https://elvex.ugr.es/decsai/deep-learning/slides/NN5%20Regularization.pdf)