UNIVERSIDAD DE GRANADA

Aprendizaje Automático

Práctica 3

Ajuste de Modelos Lineales

Autor: Francisco Javier Bolívar Expósito

Curso: 2019-2020

Índice

1. Problema de clasificación

1.1. Comprender el problema a resolver. Identificar los elementos X, Y and f del problema.

El problema a resolver es el ‘reconocimiento óptico de dígitos escritos a mano’.

Disponemos de un data set con 5620 muestras, que se encuentra separado previamente en un ‘training set’ con 3823 muestras y un ‘test set’ con 1797 muestras. El test set se ha obtenido a partir de los dígitos escritos a mano por personas diferentes a las que contribuyeron para el training set.

Las características de cada muestra se han obtenido a partir de bitmap de 32x32 de los dígitos, que es dividido en bloques de 4x4 que no se solapan. Por cada bloque de 4x4 tenemos una característica que cuenta el número de píxeles ‘activos’ (pintados) del bloque. Esto permite reducir la dimensionalidad y que pequeñas distorsiones no den lugar a varianza en las características. Por lo tanto disponemos de 64 características de números enteros en el rango [0, 16].

Las clases a la que puede pertenecer una muestra son aquellas con el código en el rango de enteros [0, 9] correspondiendo a los dígitos, del 0 al 9.

Por lo tanto nuestro modelo tiene que ‘predecir’ a partir de un input de 64 enteros en el rango [0, 16] la clase ‘Y’ que indica a partir de que dígito dibujado a mano se ha obtenido la muestra.

Otra información que es importante tener en cuenta sobre nuestro data set es la distribución de clases. No es necesario calcularla ya que se nos indica en el fichero optdigits.names, vemos que no hay desbalanceo de clases ya que tanto en el training set como en el test set cada clase representa aproximadamente un 10% del conjunto.

En principio no se nos indica que ninguna característica del data set sea ‘no predictiva’, es decir que no aporte información relevante para predecir la clase.

Voy a analizar la varianza de cada característica, si presenta con frecuencia los mismos valores puede que no esté aportando información relevante. Como idea intuitiva puede que algunas zonas de la imagen no sean pintadas para ningún dígito y por lo tanto no aporten información, si esto es verdad estas características tendrán una varianza y una media cercana a 0.

```
Buscamos características con baja varianza
Se muestran aquellas con var < 0.4:

La característica nº0 tiene var = 0.0 y media = 0.0
La característica nº8 tiene var = 0.01 y media = 0.0
La característica nº16 tiene var = 0.01 y media = 0.0
La característica nº23 tiene var = 0.19 y media = 0.05
La característica nº24 tiene var = 0.0 y media = 0.0
La característica nº31 tiene var = 0.0 y media = 0.0
La característica nº32 tiene var = 0.0 y media = 0.0
La característica nº39 tiene var = 0.0 y media = 0.0
La característica nº40 tiene var = 0.1 y media = 0.03
La característica nº47 tiene var = 0.05 y media = 0.02
La característica nº48 tiene var = 0.07 y media = 0.02
La característica nº56 tiene var = 0.0 y media = 0.0
```

Varianzas relevantes de las características. Valores redondeados

Efectivamente se observa como muchas de las características correspondientes a los laterales tienen una varianza y media cercanas a 0, ya que no son pintadas la mayoría de las veces.

Por lo tanto estas características parecen no aportar información. Sin embargo podría darse el caso de que aunque en general estas característica no aporten mucha información tengan un valor mayor a 0 con frecuencia para una clase en concreto y sean relevantes para predecir esta, por lo tanto voy a contar los valores diferentes a 0 de las características con media mayor a 0.005 para cada clase.

Valores distintos de 0 de características con var < 0.4 para cada clase

	Index	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
0	Total samples of each Class	376	389	380	389	387	376	377	387	380	382
1	Caract. 23 not zero	0	0	0	0	36	0	0	21	0	10
2	Caract. 40 not zero	0	0	0	0	39	0	0	0	0	0
3	Caract. 47 not zero	0	0	0	2	1	2	40	0	0	3
4	Caract. 48 not zero	0	0	1	0	31	0	0	0	0	0

No parece que exista el problema mencionado, siendo estas características buenas candidatas a ser eliminadas, evitando así que el modelo haga overfitting por variables no relevantes.

Además vamos a analizar la correlación entre características, usando tanto el coeficiente de correlación de Pearson como el de Spearman. Mostramos solo aquellas características que tengan una correlación muy alta (mayor a 0.9 o menor a -0.9).

```
Las siguientes características tienen un coeficiente de correlación de Pearson mayor a 0.9 o menor a -0.9
nº2 con nº58 tiene corr: 0.9211786949205991

Las siguientes características tienen un coeficiente de correlación de Spearman mayor a 0.9 o menor a -0.9
nº2 con nº58 tiene corr: 0.9338115316378144
```

He decido quitar también una de estas dos características ya que aportan valores muy similares. Arbitrariamente he escogido la 58.

1.2. Selección de las clase/s de funciones a usar. Identificar cuáles y porqué.

Con el objetivo de no perder tiempo en optimización prematura es buena idea obtener un modelo funcional simple lo más rápido posible, y una vez que tenemos este modelo buscar los mejores cambios que podemos hacer para evitar problemas. Por esta razón en principio usaré combinaciones lineales de los valores observados.

Tras analizar el error de validación cruzada y el error de entrenamiento del modelo dependiendo de hay un problema de ‘bias’ (para el que aumentar la complejidad de la función podría ser una solución) o de alta varianza / overfitting tomaré decisiones acordes al problema para mitigarlo.

Por lo tanto nuestra función de predicción lineal será:

$$f(x) = w^T x + b$$

Siendo x el vector de características de nuestra muestra de entrenamiento y w y b los parámetros que el modelo deberá ajustar. (También podría eliminarse la b de la fórmula y entender la característica X_0 como una que siempre tenga valor 1)

1.3. Fijar conjuntos de training y test que sean coherentes.

En este problema se nos proporciona directamente un conjunto de training que tiene el 68% de la muestras y un conjunto de test con el 32% de las muestras que parecen bastante coherentes, conservando la proporción de clases y con una cantidad suficiente de muestras en cada conjunto, por lo que utilizaré los conjuntos proporcionados.

Usaré validación cruzada ya que proporciona una medida independiente de las muestras usadas para el entrenamiento y mitiga el problema del overfitting a las muestras elegidas para el entrenamiento. No es necesario por lo tanto definir un conjunto de validación ya que usaré funciones de librerías para la validación cruzada sobre el conjunto de entrenamiento y una vez tenga el modelo seleccionado usaré todo el training set para el entrenamiento.

1.4. Preprocesado los datos: codificación, normalización, proyección, etc.

Como hemos observado no tenemos datos perdidos, por lo que no es necesario inputación o tomar otra decisión sobre estos casos.

Tampoco es necesario modificar la codificación, la representación de cuantos píxeles hay activos en un bloque como una cantidad numérica se considera correcta.

Todas nuestras variables están en un rango definido [0, 16], por lo que la escala de las variables no va a hacer que algunas tengan más relevancia que otras, ni tampoco habrá problema por algún ‘outlier’ una muestra que tenga un valor muy separado del resto, ya que están distribuidas en este rango. Por lo tanto tampoco encuentro una justificación para que la normalización de los datos sea necesario.

En conclusión, el único procesamiento de los datos será eliminar las características 0, 8, 16, 23, 24, 31, 32, 39, 40, 47, 48, 56, ya que hemos detectado en el análisis inicial que no aporta información predictiva al estar con valor 0 en la gran mayoría de las muestras.

También he eliminado la característica 58 por tener una fuerte correlación con la característica 2.

De esta forma mitigamos que el modelo haga overfitting al intentar explicar variables no relevantes y lo simplificamos, si el modelo tuviera un alto ‘bias’ por no tener suficiente información podríamos probar a volver a añadirlas.

1.5. Fijar la métrica de error a usar. Discutir su idoneidad para el problema.

Para evaluar el rendimiento del modelo voy a usar la métrica ‘Accuracy’ que simplemente mide el porcentaje de muestras correctamente clasificadas por el modelo. Es una métrica adecuada para

problemas de clasificación de múltiples clases y tiene la ventaja de que nos permite evaluar el modelo de forma simple con un solo número.

También se escoge esta métrica debido a que las clases están balanceadas, si no lo estuvieran o predecir alguna clase tuviera mayor importancia que predecir otras podría ser más adecuado utilizar otras métricas como 'Precision' o 'Recall' que nos permitieran fijarnos en el rendimiento para una clase en concreto.

1.6 Discutir la técnica de ajuste elegida.

Se han utilizado dos técnicas de ajuste. SGD (Stochastic Gradient Descent) ya que en comparación con otros algoritmos vistos en prácticas tiene un rápido ratio de convergencia y SAGA, un algoritmo que al igual que el SGD es un método iterativo basado en gradientes, muy reciente, con mayor efectividad y ratio de convergencia.

SGD ha sido escogido por su eficiencia, aunque tiene la desventaja de tener un mayor número de parámetros a escoger que influyen en el algoritmo (punto de inicio, learning rate) y la necesidad de que la función sea derivable, que en este caso no supone un problema.

En pseudocódigo el SGD utilizado sigue el siguiente proceso:

- Escoger un vector inicial de parametros w y un learning rate η.
- Repetir hasta que se llegue a las iteraciones máximas o la mejora en una época sea menor a tol:
  - Permutar aleatoriamente el orden de la muestras de entrenamiento.
  - Para cada i = 1, 2, …, n:
    - $w := w - \eta \nabla Q_i(w)$

Siendo $\nabla Q_i(w)$ una estimación del gradiente de la función a minimizar para todo el data set pero calculada con una solo muestra, reduciendo así la carga computacional.

Respecto a SAGA se presenta la siguiente imagen del pseudocódigo tal y como se presenta en el paper original sobre esta técnica de ajuste.

(Fuente: https://papers.nips.cc/paper/5258-saga-a-fast-incremental-gradient-method-with-support-for-non-strongly-convex-composite-objectives.pdf)

We start with some known initial vector $x^0 \in \mathbb{R}^d$ and known derivatives $f_i'(\phi_i^0) \in \mathbb{R}^d$ with $\phi_i^0 = x^0$ for each i. These derivatives are stored in a table data-structure of length n, or alternatively a $n \times d$ matrix. For many problems of interest, such as binary classification and least-squares, only a single floating point value instead of a full gradient vector needs to be stored (see Section 4). SAGA is inspired both from SAG [1] and SVRG [5] (as we will discuss in Section 3). SAGA uses a step size of γ and makes the following updates, starting with $k = 0$:

SAGA Algorithm: Given the value of x^k and of each $f_i'(\phi_i^k)$ at the end of iteration k, the updates for iteration $k+1$ is as follows:

1. Pick a j uniformly at random.
2. Take $\phi_j^{k+1} = x^k$, and store $f_j'(\phi_j^{k+1})$ in the table. All other entries in the table remain unchanged. The quantity ϕ_j^{k+1} is not explicitly stored.
3. Update x using $f_j'(\phi_j^{k+1})$, $f_j'(\phi_j^k)$ and the table average:

$$w^{k+1} = x^k - \gamma \left[f_j'(\phi_j^{k+1}) - f_j'(\phi_j^k) + \frac{1}{n} \sum_{i=1}^{n} f_i'(\phi_i^k) \right], \tag{1}$$

$$x^{k+1} = \text{prox}_\gamma^h \left(w^{k+1} \right). \tag{2}$$

The proximal operator we use above is defined as

$$\text{prox}_\gamma^h (y) := \underset{x \in \mathbb{R}^d}{\text{argmin}} \left\{ h(x) + \frac{1}{2\gamma} \|x - y\|^2 \right\}. \tag{3}$$

1.7. Discutir la necesidad de regularización y en su caso justificar la función usada para ello.

Tenemos un gran número de características, por lo que utilizar una regularización L1 o Lasso puede ser una buena idea para intentar reducir el coeficiente de alguna de estas a 0. Sin embargo, ya hemos eliminado algunas características manualmente por lo que podría llegar a tener demasiado sesgo.

Se puede estimar fácilmente usando validación cruzada cual es la mejor opción, así que aunque en principio usar regularización L1 parece ser buena idea se probará en el apartado de estimación de hiperparámetros a aplicar una regularización L1 y L2, asignando distintos factores de importancia a la regularización.

Como se verá en el apartado 1.9. la opción escogida es L1.

Las funciones usadas para la regularización son las siguientes, donde se usa el parámetro C para controlar la relevancia de la regularización.

Regularización Lasso, consistente en añadir a la función de coste como penalización el valor absoluto de la magnitud de los coeficientes.

$$min_w \sum_{i=1}^{n} |w_i| + C \cdot CostFunction \qquad min_w \|w\|_1 + C \cdot CostFunction$$

Forma Vectorizada

Regularización Ridge, consistente en añadir a la función de coste como penalización la magnitud al cuadrado de los coeficientes.

$$min_w \sum_{i=1}^{n} W_i^2 + C \cdot CostFunction \qquad min_w w^T w + C \cdot CostFunction$$

Forma Vectorizada

1.8. Identificar los modelos a usar

Como primer modelo vamos a considerar un modelo lineal simple para la clasificación, el Perceptron, que obtiene los coeficientes del modelo minimizando la siguiente función de coste. En este modelo no se aplica regularización.

$$min_{w,b} \frac{1}{n} \sum_{i=1}^{n} max(0, -y_i(W_i^T x_i + b))$$

Para obtener una predicción con este modelo obtendríamos el signo de f(x).

Partiendo de un modelo lineal para la clasificación, la Regresión Logística, en el que se modelan las probabilidades de las posibles clasificaciones de una muestra usando una función logística (la función sigmoide), que tendrá ligeras variaciones al usar regularización.

Según la regularización a usar el modelo consiste en minimizar funciones de coste, compuestas del término de regularización y el coste de no acertar en las predicciones (este último se presenta en una forma más compacta a la estudiada en clase, siendo una función equivalente).

La intensidad de la regularización se controlará con el valor de C, cuanto mayor lo establezcamos menor será la intensidad, si establecemos un valor muy alto de C sería equivalente a no tener regularización.

Las funciones de coste serían las siguientes:

Regresión logística con regularización Lasso

$$min_{w,c} \|w\|_1 + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1)$$

Regresión logística con regularización Ridge

$$min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1)$$

Estos modelos sin embargo sirven para resolver un problema de clasificación binaria, cuando nuestro problema es de multiclasificación.

Para adaptar el Perceptron se utiliza la estrategia OvR (One-vs.-rest) que consiste en entrenar K clasificadores, uno por cada clase, con las muestras que corresponden a esa clase como positivas y el resto de muestras como negativas. Después la predicción se obtiene como:

$$\hat{y} = \underset{k \in \{1 \dots K\}}{\operatorname{argmax}} \; f_k(x)$$

Para adaptar la regresión logística se utiliza regresión logística multinomial. Parecido a OvR se modela la probabilidad de que cada muestra pertenezca a una clase, pero además se tiene en cuenta la dependencia entre estas probabilidades. La probabilidad de todas las clases tiene que sumar 100% (20% de que sea clase 1, 60% clase 2, 20% clase 3), por lo que la probabilidad de que una muestra pertenezca a una clase será igual a 1 – la suma de las probabilidades de que pertenezca a otras clases.

1.9. Estimación de hiperparámetros y selección del mejor modelo

Para la estimación de hiperparámetros se ha usado validación cruzada.

Esta técnica se usa para estimar la precisión de un modelo en un hipotético conjunto de datos de prueba (ya que nuestro modelo puede ajustarse muy bien al train set pero no generalizar bien), teniendo independencia esta estimación en comparación a simplemente realizar una partición entre un conjunto de training y otro de prueba.

Se ha utilizado '5-Fold cross-validation', dividiendo los datos de entrenamiento en 5 subconjuntos. Repitiendo el proceso para cada conjunto en 5 iteraciones, se calcula el error del modelo usando un subconjunto como conjunto de prueba y el resto como conjunto de entrenamiento. Tras esto se realiza la media de los resultados de cada iteración y se obtiene el error de validación Ecv. $E_{cv} = \frac{1}{K} \sum_{i=1}^{K} E_i$

Esta forma de estimar la calidad de un modelo está menos sesgada al evaluar a partir de 5 combinaciones, teniendo la desventaja sin embargo de requerir un alto tiempo de computación en comparación con una validación simple.

Teniendo esta forma de estimar la calidad de un modelo, simplemente realizaremos validación cruzada para cada posible conjunto de parámetros y seleccionaremos como mejor modelo el que mejor resultado de Ecv obtenga. Se ha usado Accuracy como la métrica de error para la validación cruzada.

CV para RL

	mean_fit_time	param_C	param_penalty	mean_test_score	std_test_score	rank_test_score
0	1.027821	0.001	l1	0.875236	0.018196	14
1	0.200027	0.001	l2	0.959980	0.003459	12
2	1.090899	0.01	l1	0.947163	0.004489	13
3	0.794276	0.01	l2	0.966260	0.007495	2
4	1.101941	0.1	l1	0.965737	0.006623	4
5	0.784613	0.1	l2	0.966259	0.008154	3
6	1.108380	1	l1	0.967044	0.007545	1
7	0.789786	1	l2	0.964951	0.006680	9
8	1.202024	10	l1	0.965474	0.006536	6
9	0.780999	10	l2	0.965735	0.006313	5
10	1.251472	100	l1	0.965212	0.005698	7
11	0.789469	100	l2	0.964951	0.005921	9
12	1.300856	1000	l1	0.965212	0.006214	7
13	0.789193	1000	l2	0.964428	0.005977	11

Resultados de la CV para los parámetros de la RL

```
Resultados de selección de hiperparámetros por validación cruzada
LR Best hyperparameters:  {'C': 1.0, 'penalty': 'l1'}
LR CV-Accuracy : 0.9670437668959382

P Best hyperparameters:  {'tol': 1e-06}
P CV-Accuracy : 0.9382753310748383
```

Para el modelo de regresión logística se han considerado los parametros ‘Regularización’ (L1 o L2) y el valor en la función de coste de ‘C’ (1.e-03, 1.e-02, 1.e-01, 1.e+00, 1.e+01, 1.e+02, 1.e+03).

Finalmente los hiperparámetros escogidos son regularización L1 y C = 1.0, obteniendo un Ecv en Accuracy de 0.96704

CV para Perceptron

	mean_fit_time	param_tol	mean_test_score	std_test_score	rank_test_score
0	0.036146	1e-06	0.938275	0.011409	1
1	0.036350	1e-05	0.938275	0.011409	1
2	0.036236	0.0001	0.938275	0.011409	1
3	0.036274	0.001	0.938275	0.011409	1
4	0.036135	0.01	0.938275	0.011409	1
5	0.034744	0.1	0.934348	0.011253	7
6	0.028792	1	0.938013	0.009982	6
7	0.020410	10	0.916814	0.014631	10
8	0.018701	100	0.922840	0.016975	8
9	0.018625	1000	0.922840	0.016975	8

Resultados CV para parámetros del Perceptron

Para el modelo del Perceptron se han considerado las siguientes tolerancias ‘tol’ para parar (parando si función de coste > función de coste iter. Anterior – tol): (1.e-06, 1.e-05, 1.e-04, 1.e-03, 1.e-02, 1.e-01, 1.e+00, 1.e+01, 1.e+02, 1.e+03). Se observa que a partir de tol = 0.1, se obtiene los mismos resultados para resultados más pequeños, ya que llega al máximo de iteraciones (1000) antes que a la condición de parada.

Finalmente el hiperparámetro escogido es tol 1.e-06, obteniendo un Ecv en Accuracy de 0.93827

El tiempo de entrenamiento del Perceptron es normalmente menor al de la Regresión Logística, pero esta última consigue para la mayoría de parámetros mejores resultados. Basándonos en los resultados obtenidos el mejor modelo a seleccionar sería la regresión logística con los parámetros indicados. Por completitud se van a tener en cuenta los dos modelos en los siguientes apartados, pero teniendo en cuenta que tras el proceso de validación cruzada me hubiera quedado con la regresión logística.

1.10. Estimación por validación cruzada del error E_{out} del modelo. Comparación con E_{test}

Con la estimación por validación cruzada obtenemos que aproximadamente

$E_{out}(g) \leq E_{cv}(g)$ (Para una medida de error en la que cuanto menor error mejor modelo)

Como hemos evaluado los modelos con una métrica que cuanto mayor es mejor, podemos extraer de Accuracy (Porcentaje de muestras correctamente clasificadas) una medida de 'Miss-classification' (Porcentaje de muestras mal clasificadas) haciendo 1 – Accuracy y utilizar esta medida para las estimaciones de $E_{out.}$ Por lo tanto para la estimación por validación cruzada obtenemos Ecv como la media de los resultados en las 5 iteraciones de la validación cruzada y tendríamos:

$E_{out}(g) \leq 0{,}0329563$ *Cota RL*

$E_{out}(g) \leq 0{,}061724669$ *Cota Perceptrón*

```
Métricas de evaluación para los modelos entrenados para train y test
LR Train-Accuracy: 0.9934606330107245
P Train-Accuracy: 0.9521318336385038

LR Test-Accuracy: 0.9571508069003896
P Test-Accuracy: 0.9187534780189204
```

A partir de la desigualdad de Hoeffding podemos obtener una cota probabilística de Eout.

$$E_{out}(g) \leq E_{test}(g) + \sqrt{\frac{1}{2N}\ln\frac{2}{\delta}}$$

N = 1797 δ = 0.05

Tenemos las siguientes cotas de Eout con un 95% de confianza

$$E_{out}(g) \leq 0{,}0438492 + \sqrt{\frac{1}{2\cdot 1797}\ln\frac{2}{0.05}} = 0.0758877$$

Cota RL

$$E_{out}(g) \leq 0{,}0812466 + \sqrt{\frac{1}{2\cdot 1797}\ln\frac{2}{0.05}} = 0.113284$$

Cota Perceptrón

Las cotas basadas en el test set nos dan un resultado menos optimista y con menor sesgo al estar basada en prueba con datos fuera del conjunto de entrenamiento.

Si el error se hubiera obtenido de una muestra más grande daríamos una cota con un intervalo más reducido, esto muestra la importancia de la cantidad de los datos en el aprendizaje automático, tanto para realizar un buen entrenamiento como para acotar el error.

El 7% de error con un 95% de confianza de la Regresión Logística nos da un resultado bastante bueno y mejor que el del Perceptrón, lo que indica que efectivamente el modelo seleccionado fue el correcto.

1.11. ¿Qué modelo propondría y que error E_{out} diría que tiene?

Propondría el modelo de regresión logística con L1 y C = 1.0 ya que es el que mejor resultados ha presentado en la validación cruzada.

Les diría que tiene un Eout menor o igual a clasificar incorrectamente un 7.5% de las muestras al 95% de confianza.

Gracias a las cotas (que no podríamos usar en un caso real para escoger el modelo) podemos ver que efectivamente el modelo seleccionado es el que proporciona el mejor error.

El modelo representa de manera adecuada los datos, la combinación lineal es suficiente para explicarlos claramente, viendo como el error de entrenamiento es ínfimo pero aun así se conserva la generalización no hay ninguna razón aparente para probar una clase de funciones más compleja o para volver a incorporar las características eliminadas.

Si fuera necesario mejorar el modelo, de hecho, me decantaría antes por intentar reducir la varianza / evitar overfitting que pensar que existe un problema de sesgo. Incluso después de que la mayoría de decisiones tomadas hayan sido con el objetivo de reducir el overfitting (regularización, reducción de características, clase de funciones simple) la muestra de entrenamiento obtiene un error casi nulo a diferencia de en la validación cruzada y el conjunto de test, lo que nos da a entender que es capaz de explicar la varianza de la muestra de entrenamiento pero no generalizar con un resultado tan bueno fuera de esta. Sin embargo, los resultados obtenidos no indican que este problema sea muy acentuado y se considera que el modelo es de calidad.

2. Problema de Regresión

2.1. Comprender el problema a resolver. Identificar los elementos X, Y and f del problema.

El problema a resolver es la predicción de crímenes violentos por cápita para comunidades de EEUU.

Disponemos de un data set con 1994 muestras sin separación en distintos conjuntos.

Cada muestra tiene 127 características, a diferencia del problema de clasificación cada característica puede tener una naturaleza muy diferente, desde los ingresos por cápita a el porcentaje de personas en un rango de edad. Sin embargo, todas las características se encuentran normalizadas en el rango [0, 1] conservando aproximadamente la relación entre los valores de una misma característica.

La variable a predecir (crímenes violentes por cápita) también se encuentra normalizada en el rango [0, 1]

Por lo tanto nuestro modelo tiene que ‘predecir’ a partir de un input X de factores socio-económicos de una comunidad el elemento ‘Y’ que indica los crímenes violentos por cápita de esa comunidad.

Respecto a la distribución de la variable a predecir se proporciona la siguiente información con el dataset:

Range	Frequency
0.000-0.067	484
0.067-0.133	420
0.133-0.200	284
0.200-0.267	177
0.267-0.333	142
0.333-0.400	113
0.400-0.467	59
0.467-0.533	76
0.533-0.600	57
0.600-0.667	38
0.667-0.733	37
0.733-0.800	20
0.800-0.867	23
0.867-0.933	14
0.933-1.000	50

Se han obtenido más datos sobre aquellas comunidades con un menor índice de violencia, y se tienen menos muestras de los casos extremos con un alto índice de violencia.

Parece una distribución natural, no está causada porque se hayan recogido menos muestras de un tipo si no por la proporción real de cada tipo de muestra, por lo que no debería influir muy negativamente en el entrenamiento.

Será relevante tener en cuenta está frecuencia de los rangos al decidir la separación entre el conjunto de entrenamiento y el conjunto de test.

Los 5 primeros atributos no son predictivos ya que son simplemente identificadores (código numérico de la comunidad, del estado, nombre de esta, …) y una variable para usos de depuración, que asigna cada muestra a un ‘fold’ de una validación cruzada no aleatoria.

También se nos indica que el dataset contiene valores perdidos, echando un vistazo rápido a la información proporcionada en .names la mayoría de características que tienen valores perdidos tienen

1675 perdidos de los 1994 (84% de valores perdidos). Vamos a calcular una pequeña tabla para ver solo los valores perdidos.

```
Missing Values Analisis
-----------------------
Característica  Valores perdidos
25              1
96              1675
97              1675
98              1675
99              1675
100             1675
101             1675
102             1675
103             1675
104             1675
105             1675
106             1675
107             1675
108             1675
109             1675
110             1675
111             1675
112             1675
116             1675
117             1675
118             1675
119             1675
121             1675
```

Vemos que de las características predictivas que tienen valores perdidos todas excepto una variable tienen la mayor parte sin valor (84% de valores perdidos). (Al no tener en cuenta las variables no predictivas la numeración está cambiada, siendo la característica n.º 5 la n.º 0)

Al igual que en el problema de clasificación vamos a estudiar si hay variables que aportan poca información, aunque esta vez no he detectado intuitivamente ningún caso tan claro. Al estar esta vez los datos normalizados entre 0 y 1 el criterio para que un atributo tenga poca varianza tiene que ser más duro.

```
Variance analisis
-----------------
Features with low variance (var < 0.02)

Feature nº0: var = 0.014008754178936116  mean = 0.05
Feature nº10: var = 0.014362549102644647  mean = 0.06
Feature nº27: var = 0.012888381981007604  mean = 0.05
Feature nº49: var = 0.007896391123438411  mean = 0.03
Feature nº51: var = 0.00584952203383286  mean = 0.03
Feature nº71: var = 0.01932148532467447  mean = 0.07
Feature nº89: var = 0.007845195866498388  mean = 0.03
Feature nº90: var = 0.006664074270002122  mean = 0.02
Feature nº96: var = 0.011700528582108329  mean = 0.06

Valores distintos de 0
  Feature nº  Non zero values
0          0             1609
1         10             1426
2         27             1680
3         49             1244
4         51             1168
5         71             1912
6         89              734
7         90              397
8         96             1922
```

2.2. Selección de las clase/s de funciones a usar. Identificar cuáles y porqué.

Al igual que en el problema de clasificación, con el objetivo de no perder tiempo en optimización prematura se intentará obtener un modelo funcional simple lo más rápido posible, y una vez que tenemos este modelo buscar los mejores cambios que podemos hacer para evitar problemas. Por esta razón en principio usaremos combinaciones lineales de los valores observados. Tras analizar el error de validación cruzada y el error de entrenamiento del modelo dependiendo de si encontramos un problemas de ‘bias’ (para el que aumentar la complejidad de la función podría ser una solución) o de alta varianza / overfitting se tomarán decisiones acordes al problema para mitigarlo.

Por lo tanto nuestra función de predicción lineal será:

$$f(x) = w^T x + b$$

Siendo x el vector de características de nuestra muestra de entrenamiento y w y b los parámetros que el modelo deberá ajustar.

2.3. Fijar conjuntos de training y test que sean coherentes.

En este problema no hay distinción en conjuntos de las muestras por lo tanto tenemos que establecerla nosotros.

Debido al reducido número de muestras con un valor de la variable a predecir mayor a 0.5 corremos el riesgo de que la mayoría de este tipo no entren en el training set si fijamos los conjuntos de manera

aleatoria, sin tener un conjunto representativo con el que el modelo pueda aprender y generalizar de forma adecuada. Por este motivo se ha decidido respetar la proporción tanto en el training set como en el test set.

En la implementación se han separado los datos en varios conjuntos (16), cada conjunto contiene muestras que tienen un rango concreto de valores de Y, estos rangos se han calculado de forma que estén equitativamente separados en el intervalo [0, 1]. Teniendo los rangos [0, 0.067), [0.067, 0.133), …

El training set contiene un 75% de las muestras de cada uno de estos conjuntos y el test set el 25% restante.

2.4. Preprocesado los datos: codificación, normalización, proyección, etc.

El data set tiene un gran número de valores perdidos, por lo que es necesario tomar alguna decisión respecto a estos, ya sea imputar valores o eliminar muestras / características.

Se ha decidido eliminar las características que como hemos visto en el apartado 2.1. solo tienen valores en el 16% de las muestras, por lo que no sufriremos una perdida de información muy grave y conseguiremos un modelo más exacto que si se imputaran los valores provocando un añadido de varianza o sesgo al modelo. Para la característica en la que solo hay un valor perdido se ha decidido imputarlo con la media de la columna.

También se han eliminado las características indicadas como no predictivas indicadas en .names.

Sobre eliminar características basándose en el estudio de la varianza no se ha considerado que ninguna no aporte suficiente información, en principio no se eliminará ninguna, si existiera un problema de overfitting se podría revisar esta decisión.

Las datos ya se encuentran normalizados previamente, por lo que no se va a realizar ninguna transformación de este tipo.

2.5. Fijar la métrica de error a usar. Discutir su idoneidad para el problema.

Como métrica de error para medir la calidad del modelo se va a usar principalmente MAE (Mean Absolute Error) que calcula el valor medio de los errores absolutos y de forma secundaria RMSE (Root Mean Squared Error) que calcula la raíz cuadrada del valor medio de los cuadrados de los errores. Cuanto más pequeños sean los valores de estas métricas mejor resultado.

$$RMSE=\sqrt{\frac{1}{n}\sum_{i=1}^{n}(predicted_i-y_i)^2} \qquad MAE=\frac{1}{N}\sum_{i=1}^{n}|y_i-predicted_i|$$

Ambas métricas son adecuadas para problemas de regresión y tienen la ventaja de ser fácilmente interpretables ya que nos permiten evaluar el modelo de forma simple con un solo número. Además, a

diferencia de por ejemplo MSE (Mean Squared Error) las métricas devueltas están dentro de la escala del problema [0, 1].

Se observan los resultados de las dos ya que MAE nos da una media del error absoluto que es una métrica fácilmente interpretable y adecuada al problema ya que un error el doble de grave nos basta con penalizarlo como el doble de pero. RMSE sin embargo al penalizar muy fuertemente los 'outliers' puede ser interesante también ya que si es muy alta podemos detectar que hay grandes errores para algunas muestras en concreto (puede que según la distribución del rango de Y) y si este fuera el caso lo podríamos analizar más en profundidad.

2.6 Discutir la técnica de ajuste elegida.

Se han escogido dos técnicas para ajustar los pesos del modelo lineal. Primero SGD (Sthocastic Gradient Descent) y el uso del algoritmo de la pseudo-inversa.

SGD ha sido escogido por su eficiencia, aunque tiene la desventaja de tener un mayor número de parámetros a escoger que influyen en el algoritmo (punto de inicio, learning rate) y la necesidad de que la función sea derivable, que en este caso no supone un problema.

En pseudocódigo el SGD utilizado sigue el siguiente proceso:

- Escoger un vector inicial de parametros w y un learning rate η_0. $\eta = \eta_0$
- Repetir hasta que se llegue a las iteraciones máximas o la mejora en una época sea menor a tol:
  - Permutar aleatoriamente el orden de la muestras de entrenamiento.
  - Para cada i = 1, 2, …, n:
    - $w := w - \eta \nabla Q_i(w)$
- $\eta = \eta_0 / t^{0.25}$

Siendo $\nabla Q_i(w)$ una estimación del gradiente de la función a minimizar para todo el data set pero calculada con una solo muestra, reduciendo así la carga computacional. t es el número de actualizaciones de los pesos realizadas, que como se ve en el pseudocódigo se usa para realizar una actualización del learning rate, disminuyendo su valor conforme aumentamos en las iteraciones para mejorar en la convergencia.

La Pseudo-Inversa puede ser algo más costosa de calcular pero nos da una solución precisa.

El ajuste de pesos con la pseudo-inversa se realiza como se indica a continuación:

$$w = (X^T X)^{-1} X^T y$$

2.7. Discutir la necesidad de regularización y en su caso justificar la función usada para ello.

Tenemos un gran número de características y en el preprocesado de los datos no hemos hecho una preselección muy exhaustiva, por lo que una fuerte regularización puede ayudar a quitar relevancia si

existieran características no predictivas y a que el modelo generalice mejor. Probaremos tanto con regularización Lasso (L1) como con Ridge (L2). No se ha detectado que haya muchas características que influyan mucho menos que otras en el modelo, por lo que si la mayoría de atributos son relevantes podemos obtener mejor resultado con L2.

Las funciones usadas para la regularización son similares a las utilizadas para el ejercicio de clasificación, pero en este caso usamos el parámetro α para controlar la intensidad de la regularización, siendo esta mayor conforme más grande sea α.

Regularización Lasso, consistente en añadir a la función de coste como penalización el valor absoluto de la magnitud de los coeficientes.

$$min_w \alpha \cdot \sum_{i=1}^{n} |w_i| + CostFunction \qquad min_w \alpha \cdot ||w||_1 + CostFunction$$
Forma Vectorizada

Regularización Ridge, consistente en añadir a la función de coste como penalización la magnitud al cuadrado de los coeficientes.

$$min_w \alpha \cdot \sum_{i=1}^{n} W_i^2 + CostFunction \qquad min_w \alpha \cdot w^T w + CostFunction$$
Forma Vectorizada

Tras la comparación por validación cruzada en el apartado 2.9 se ha escogido regularización L2 pero con un parámetro α de 10^{-42}, por lo que el termino de regularización tiene una intensidad tan pequeña que tiene una influencia muy reducida en el entrenamiento del modelo.

2.8. Identificar los modelos a usar

Se usan un modelo de regresión lineal, con dos variantes dependiendo de la técnica de ajuste utilizada.

La regresión lineal consiste en entrenar un modelo eligiendo los coeficientes w necesarios para minimizar la suma de los errores cuadráticos en la predicción. Usando la función del apartado 2.2. obtenemos las predicciones del modelo. La función de coste base sería:

$$min_w \sum_{i=1}^{n} (y_i - predicted_i)^2$$

Añadiendo como se indica en el apartado 2.7. los términos de regularización.

Se usará este modelo con SGD y regularización L1, SGD y regularización L2 y utilizando el algoritmo de la Pseudo-Inversa.

2.9. Estimación de hiperparámetros y selección del mejor modelo

Para la estimación de hiperparámetros se ha usado validación cruzada al igual que en el problema de clasificación. Se repite la explicación de esta.

Esta técnica se usa para estimar la precisión de un modelo en un hipotético conjunto de datos de prueba (ya que nuestro modelo puede ajustarse muy bien al train set pero no generalizar bien), teniendo independencia esta estimación en comparación a simplemente realizar una partición entre un conjunto de training y otro de prueba.

Se ha utilizado '5-Fold cross-validation', dividiendo los datos de entrenamiento en 5 subconjuntos. Repitiendo el proceso para cada conjunto en 5 iteraciones, se calcula el error del modelo usando un subconjunto como conjunto de prueba y el resto como conjunto de entrenamiento. Tras esto se realiza la media de los resultados de cada iteración y se obtiene el error de validación Ecv. $E_{cv} = \frac{1}{K} \sum_{i=1}^{K} E_i$

Esta forma de estimar la calidad de un modelo está menos sesgada al evaluar a partir de 5 combinaciones, teniendo la desventaja sin embargo de requerir un alto tiempo de computación en comparación con una validación simple.

Teniendo esta forma de estimar la calidad de un modelo, simplemente realizaremos validación cruzada para cada posible conjunto de parámetros y seleccionaremos como mejor modelo el que mejor resultado de Ecv obtenga. Se ha usado MAE como la métrica de error para la validación cruzada.

Para el modelo de regresión linear ajustado con SGD se han considerado una gran cantidad de parámetros de regularización, tanto L1 como L2 y desde $\alpha = 10^{-50}$ hasta $\alpha = 10^{0}$ aumentando de 1 en 1 el factor en el que está elevado 10. Se ha decidido de esta manera ya que debido a la rapidez del ajuste no requería un gran tiempo de computación obtener unos resultados más completos. También se han probado varios valores para el learning rate (1.e-04, 1.e-03, 1.e-02, 1.e-01, 1.e+00). No se presenta la tabla de resultados por el gran número de combinaciones de parámetros disponibles.

Los hiperparámetros escogidos son regularización L2, $\alpha = 10^{-42}$ y learning rate = 0.1 obteniendo un Ecv en MAE de 0.08870

El modelo de regresion linear ajustado con la Pseudoinversa no dispone de parámetros, obteniéndose para este un Ecv en MAE de 0.0922.

```
Resultados de selección de hiperparámetros por validación cruzada
SGD Best hyperparameters:  {'alpha': 1e-42, 'eta0': 0.1, 'penalty': 'l2'}
SGD CV-MAE : 0.08870800423302641

Lin Best hyperparameters:  {'normalize': False}
Lin CV-MAE : 0.09223229622593739
```

Escogemos el SGD por presentar un Ecv mejor.

2.10. Estimación por validación cruzada del error E_{out} del modelo. Comparación con E_{test}

Con la estimación por validación cruzada obtenemos que aproximadamente

$E_{out}(g) \leq E_{cv}(g)$ (Para una medida de error en la que cuanto menor error mejor modelo)

Por lo tanto para la estimación por validación cruzada obtenemos Ecv como la media de los resultados en las 5 iteraciones de la validación cruzada y tendríamos las siguientes cotas medidas en MAE:

$E_{out}(g) \leq 0.088708$ *Cota RL-SGD*

$E_{out}(g) \leq 0.0922323$ *Cota RL-Pseudoinversa*

```
Métricas de evaluación para los modelos entrenados para train y test
SGD Train-RMSE:  0.12291831975329809
SGD Train-MAE:  0.08504927975056543
Lin Train-RMSE:  0.11715283217744356
Lin Train-MAE:  0.0841899652848829

SGD Test-RMSE:  0.1287120871146455
SGD Test-MAE:  0.08894752089989538
Lin Test-RMSE:  0.12940114749539608
Lin Test-MAE:  0.0913577371557753
```

A partir de la desigualdad de Hoeffding podemos obtener una cota probabilística de Eout.

$$E_{out}(g) \leq E_{test}(g) + \sqrt{\frac{1}{2N}\ln\frac{2}{\delta}}$$

N = 493 δ = 0.05

Tenemos las siguientes cotas de Eout con un 95% de confianza

$$E_{out}(g) \leq 0.0889475 + \sqrt{\frac{1}{2\cdot 493}\ln\frac{2}{0.05}} = 0.150113$$

Cota RL-SGD MAE

$$E_{out}(g) \leq 0.12871208 + \sqrt{\frac{1}{2\cdot 493}\ln\frac{2}{0.05}} = 0.1898779$$

Cota RL-SGD RSME

$$E_{out}(g) \leq 0.0913577 + \sqrt{\frac{1}{2\cdot 493}\ln\frac{2}{0.05}} = 0.152524$$

Cota RL-Pseudoinversa MAE

$$E_{out}(g) \leq 0.12940114 + \sqrt{\frac{1}{2\cdot 493}\ln\frac{2}{0.05}} = 0.1905670$$

Cota RL-Pseudoinversa RMSE

Las cotas basadas en el test set nos dan un resultado menos optimista y con menor sesgo al estar basada en prueba con datos fuera del conjunto de entrenamiento.

Vemos que al tener una muestra mucho más reducida del dataset que en el problema de clasificación el intervalo proporcionado por la cota es mucho más amplio o malo. Si el error se hubiera obtenido de una muestra más grande daríamos una cota con un intervalo más reducido, esto muestra la importancia de

la cantidad de los datos en el aprendizaje automático, tanto para realizar un buen entrenamiento como para acotar el error.

El 0.15 de error con un 95% de confianza de la Regresión Lineal con MAE nos da un resultado bastante peor al de la validación cruzada, aunque en ambos casos es mejor que el de la pseudoinversa al generalizar mejor, lo que indica que efectivamente el modelo seleccionado fue el correcto.

2.11. ¿Qué modelo propondría y que error E_{out} diría que tiene?

Propondría el modelo de Regresión Lineal con regularización L2, $\alpha = 10^{-42}$ y learning rate = 0.1 ya que es el que mejor resultados ha presentado en la validación cruzada.

Les diría que tiene un Eout MAE menor o igual 0.15 al 95% de confianza y RMSE menor o igual a 0.189 al 95% de confianza.

Gracias a las cotas (que no se podrían usar en un caso real para escoger el modelo) se puede ver que efectivamente el modelo seleccionado es el que proporciona el mejor error. Esto se debe claramente a que la Pseudoinversa presenta un problema de overfitting, presenta un error de entrenamiento menor al modelo seleccionado, pero después tiene un error de cross-validation peor, ya que al sobreajustarse al conjunto de entrenamiento no generaliza correctamente.

En el modelo seleccionado el error de validación cruzada y el de entrenamiento son ambos bajos y similares, por lo que se considera que el modelo representa de manera adecuada los datos, siendo la combinación lineal suficiente para explicarlos, no parece haber ninguna razón para eliminar características.

Basándose en la cota de RMSE se puede ver que no hay una diferencia muy grande respecto a MAE, lo que significa que la distribución de errores en cada muestra es similar. Esto es bueno ya que indica que no hay un problema a pesar del reducido número de muestras que indican un alto índice de criminalidad, puede que gracias a la separación balanceada que se ha hecho del train / test set.

Si fuera necesario mejorar el modelo, ya que todas las características parecen relevantes y la regularización no aporta mejores resultados, podría considerarse una clase de funciones más compleja, aunque está podría causar Overfitting además de que conllevaría un tiempo de validación cruzada y de entrenamiento mucho mayor.