Laboratorio de Datos

Trabajo Práctico 02

Dato Stereo

Integrantes del grupo	L.U.
Manuel Andrés Beren	320/22
Sofía Roitman	563/21
Dafne Sol Yudcovsky	1888/21

Resumen

En este trabajo práctico desarrollamos distintos modelos para identificar letras representadas con lenguaje de señas a partir de imágenes. Para esto, lo primero que hicimos fue analizar los datos obtenidos, teniendo en cuenta las similitudes y diferencias entre las imágenes. Luego, armamos un modelo KNN para diferenciar dos letras (A y L). A continuación, armamos un árbol de decisión para distinguir las vocales. Por último, evaluamos nuestros modelos con distintas métricas de clasificación. Concluimos finalmente que el modelo que toma 3 vecinos y los 8 píxeles con mayor diferencia entre las imágenes promedio es el más adecuado para la clasificación de las letras A y L y el modelo más eficiente para clasificar las vocales es un árbol de profundad máxima 15 con criterio entropía.

Introducción

El objetivo de este trabajo práctico es, en base a una imágen de una letra del lenguaje de señas, poder identificar qué letra es. Para esto, primero hicimos un análisis exploratorio de los datos para entender qué pixeles eran más representativos a la hora de distinguir imágenes y poder ver si las imágenes eran todas similares para una letra. Luego, determinamos qué letra se representaba entre dos opciones, la A y la L. Para esto, hicimos un modelo de KNN analizando distintas cantidades de atributos y distinta cantidad de vecinos. Después, armamos un modelo para poder determinar imágenes que contenían vocales, para esto, entrenamos un árbol de decisión, evaluando las métricas según distintas profundidades máximas y diferentes criterios, para luego quedarnos con el modelo que mejor precisión daba. Esto último se calculó usando *cross validation* y *K-folding* con K=5. Finalmente, generamos métricas de clasificación y las utilizamos para poder evaluar los modelos.

Análisis exploratorio de los datos

En este trabajo práctico desarrollamos distintos modelos para identificar letras representadas con lenguaje de señas a partir de imágenes. Para esto, lo primero que hicimos fue analizar los datos obtenidos, teniendo en cuenta las similitudes y diferencias entre las imágenes. Nos enfocamos en buscar letras que sean o bien visualmente parecidas, o bien muy diferentes, y decidimos quedarnos con una selección de estas para poder analizarlas. En particular, vimos que las señas para las letras D y U son muy similares, así como las señas que representan la M y la N. Sin embargo, las señas que representan la D y la M difieren bastante entre sí.

Teniendo lo anterior en cuenta, decidimos usar comparaciones entre imágenes promedio de estas letras, para luego indagar si hay píxeles que sean menos importantes que otros con el objetivo de determinar qué letra representa la imágen. Para esto, tomamos en consideración que las imágenes están representadas por 784 píxeles, mientras que cada uno de dichos pixeles está representado por un número entre 0 y 255 que representa la intensidad del color. Mientras más bajo sea dicho número, más oscuro es el píxel correspondiente.

Sabiendo esto, el método para comparar las imágenes promedio fue generar una nueva imágen con la diferencia entre los valores de ambas fotos para identificar los píxeles más representativos de cada imagen. Cuanto más claro el píxel, más difieren las imágenes en el mismo. En la siguiente imagen, los puntos rojos marcan los 10 píxeles más claros de cada imagen, donde se puede apreciar la diferencia en el gráfico de barras de al lado.

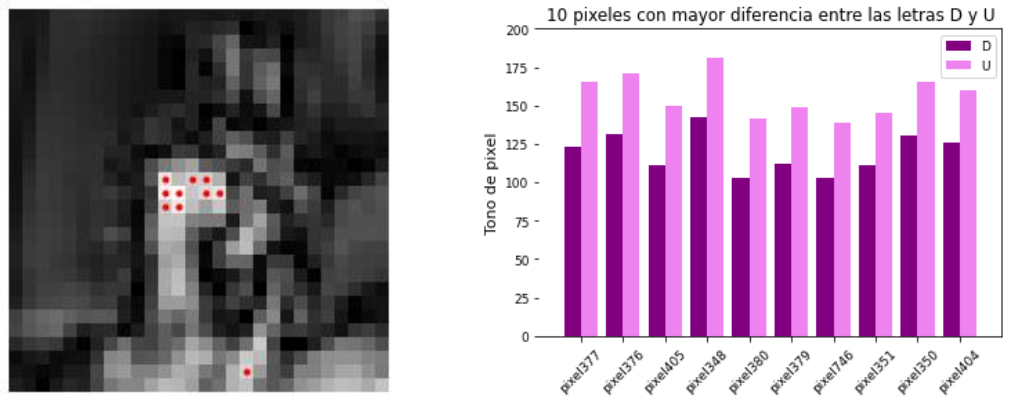

Imagen 1: Diferencia entre imágenes promedio de las letras D y U

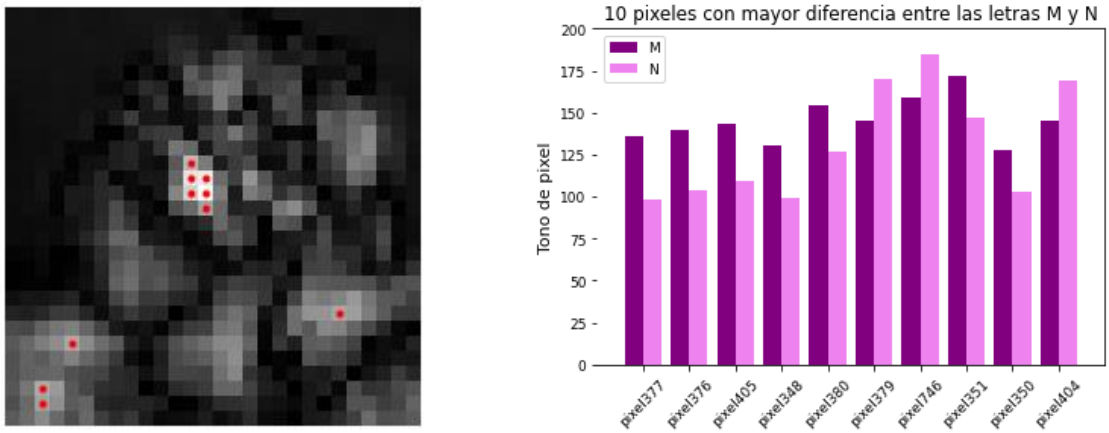

Imagen 2: Diferencia entre imágenes promedio de las letras M y N

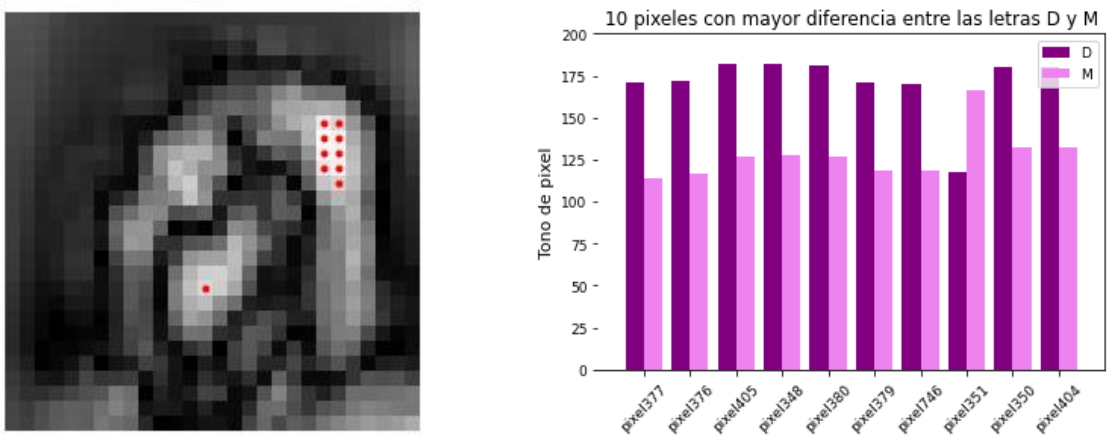

Imagen 3: Diferencia entre imágenes promedio de las letras D y M

Como podemos ver en las imágenes, los píxeles más oscuros suelen estar en los bordes, por lo cual es seguro decir que las imágenes son más distinguibles por los píxeles del centro, que es lógico, pues es donde está la mano que hace la seña.

Por otra parte, podemos afirmar que hay señas que son más fáciles de distinguir entre sí que otras. Por ejemplo, en la imagen 2, los bordes y el contorno de la mano son más oscuros y la mayor diferencia se encuentra alrededor del centro, similar a lo que ocurre con la imagen 1. Sin embargo, podemos ver que la imágen 3 tiene más partes claras que las otras, sobre todo en el centro, con lo cual, la D y la M serán más fáciles de reconocer que la M y la N, al tomar este caso particular.

Luego, para ver si todas las imágenes de una misma letra son similares entre sí, decidimos calcular una imágen en base a las letras E y K usando la desviación estándar de cada pixel. Recordemos que la desviación estándar es una medida de variabilidad entre distintas instancias de una variable aleatoria. Nuevamente, los píxeles en donde la imágen es más oscura muestran donde las imágenes son más similares entre sí y los píxeles más claros muestran lo opuesto.

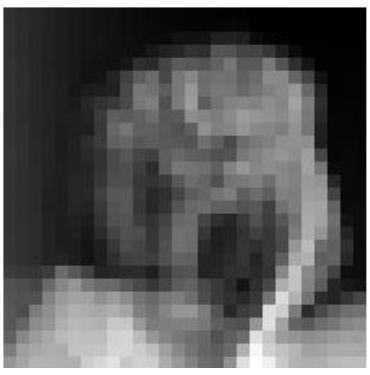

Imagen 4: Diferencia entre todas las imágenes de la letra E

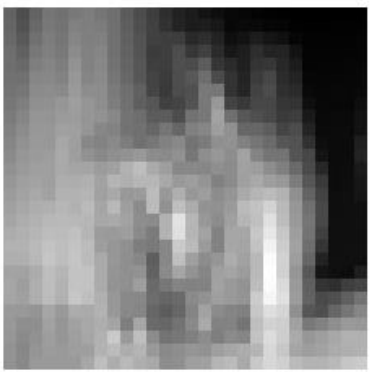

Imagen 5: Diferencia entre todas las imágenes de la letra K

Observando estas imágenes, se puede observar que no necesariamente todas las imágenes de una misma letra son iguales. En particular, las imágenes de la letra K (imagen 5) parecen ser mucho más distintas entre sí que las de la letra E (imagen 4) por la cantidad de píxeles claros de la K.

El *dataset* utilizado en este trabajo es diferente a los *datasets* que normalmente usamos para trabajar en clase. En primer lugar, tienen 784 atributos, que son muchos más que los que tienen los archivos que solemos utilizar, por ejemplo, el *dataset* del Titanic. En segundo lugar, no se puede mirar el *dataset* del lenguaje de señas directamente de las tablas porque es una fila de píxeles (números entre 0 y 255) pero una vez ploteadas, resulta mucho más sencillo de visualizar que representa cada dato y realizar comparaciones. A diferencia del *dataset* del Titanic, que si bien son menos atributos, requieren un análisis más profundo y comprender qué significa cada uno de ellos. Por otro lado, entrenar un modelo para distinguir imágenes puede ser computacionalmente más costoso por la cantidad de atributos que puede comparar, por ejemplo, la profundidad de un árbol de decisión puede ser muchísimo mayor para poder determinar la letra representada en la imágen.

Experimentos realizados

En primer lugar, queríamos determinar a qué letra pertenecía una imágen en un dominio acotado, en este caso, las letras A y L. Lo primero que hicimos fue calcular la cantidad de cada una de las letras y obtuvimos que había 1126 A y 1241 L, es decir, que las cantidades estaban balanceadas y además eran suficientes casos para poder entrenar correctamente un modelo. Después, dividimos los datos para tener un conjunto de entrenamiento y otro de test.

A continuación, decidimos armar una imágen promedio para cada una de estas letras (imágenes 6 y 7), para luego armar una matriz que represente la diferencia entre ambas, teniendo en cuenta únicamente los casos de entrenamiento. Los elementos de esta matriz de valor más grande serán los más representativos para poder distinguir a qué letra pertenece la imágen.

Imagen 6: A promedio

Imagen 7: L promedio

El modelo utilizado para distinguir las letras en este caso fue KNN (K-Nearest Neighbors) con K=3, variando la cantidad de píxeles que se usaban para comparar las imágenes. Intuitivamente, a mayor cantidad de píxeles usados, mejor debería funcionar el modelo. Sin embargo, esta relación no necesariamente se cumple a medida que aumenta el valor de K. Las cantidades de píxeles usadas fueron 3, 5 y 8, y, para cada uno de estos valores, se usaron, en primer lugar, aquellos píxeles con mayor diferencia en la matriz ya mencionada y luego los que menos diferencia tenían. Cada uno de estos modelos se evaluó con los datos de test. Las exactitudes obtenidas para los modelos que tomaban los píxeles con mayor diferencia fueron 0,97, 0,98 y 0,99 respectivamente. Y para los modelos con los píxeles de menor diferencia, fueron 0,84, 0,84 y 0,91, respectivamente.

Finalmente, decidimos hacer un *grid search* con distintos valores de K y distinta cantidad de píxeles, donde estos últimos se seleccionaron de forma aleatoria. La evaluación de cada combinación de parámetros se realizó sobre los datos de test. Esto se llevó a cabo de forma manual para poder registrar los resultados obtenidos en una tabla para poder analizarlos.

Cantidad de píxeles	Cantidad de vecinos									
	1	2	3	4	5	6	7	8	9	10
1	0.508446	0.495777	0.523649	0.519426	0.539696	0.519426	0.552365	0.546453	0.563345	0.57348
2	0.713682	0.662162	0.716216	0.712838	0.734797	0.728885	0.72973	0.716216	0.717061	0.722973
3	0.878378	0.836149	0.865709	0.847128	0.855574	0.849662	0.847128	0.848818	0.847128	0.851351
4	0.940878	0.929054	0.930743	0.921453	0.921453	0.909628	0.915541	0.903716	0.912162	0.905405
5	0.952703	0.950169	0.943412	0.935811	0.930743	0.929054	0.923986	0.919764	0.919764	0.912162
6	0.973818	0.961993	0.956926	0.949324	0.950169	0.938345	0.936655	0.936655	0.935811	0.924831
7	0.980574	0.960304	0.961993	0.954392	0.954392	0.943412	0.943412	0.938345	0.943412	0.930743
8	0.985642	0.978041	0.972973	0.964527	0.967905	0.956926	0.956926	0.946791	0.951014	0.944257
9	0.987331	0.984797	0.983108	0.977196	0.974662	0.961149	0.959459	0.951014	0.955236	0.946791
10	0.991554	0.990709	0.985642	0.983953	0.982264	0.972128	0.96875	0.966216	0.967061	0.959459

Tabla 1: Exactitud según cantidad de vecinos y cantidad de píxeles

Como se puede observar en la tabla 1 y con lo dicho anteriormente, a medida que aumentan los píxeles, mejora la precisión, pero esto no necesariamente ocurre cuando aumenta la cantidad de vecinos. De acuerdo con esta tabla, los mejores valores se dan cuando se toman muchos píxeles y pocos vecinos, aunque esto depende también de los píxeles tomados, que, en este caso, fueron seleccionados aleatoriamente (con el comando 'sample'). Es importante notar que, igualmente, cada combinación de parámetros usó los mismos píxeles a la hora de entrenarse (misma semilla).

Luego, nos dispusimos a determinar a qué letra pertenecía una imágen en un dominio mayor, que fue el de las vocales. Para esto, realizamos un árbol de decisión. En primer lugar, separamos los datos en dos, entrenamiento y test. Para determinar los hiper parámetros del árbol, primero realizamos uno genérico y tomamos su máxima profundidad, que fue 16. Luego, hicimos una *randomized search* donde los hiper parámetros propuestos fueron los criterios entropía y gini y la profundidad máxima todos los valores entre 1 y 16 (la profundidad obtenida anteriormente). Para *la randomized search*, usamos *stratified K fold* con K = 5 para dividir los datos y poder hacer validación cruzada y evaluamos la mitad de las combinaciones de hiper parámetros posibles. Finalmente, luego de armar el modelo con el mejor resultado obtenido de la *randomized search,* que fue entropía como criterio y 15 como profundidad máxima del árbol (imagen 8), lo evaluamos sobre los datos de test y usamos métricas de clasificación para este. El primer valor obtenido fue una exactitud de 0,97, que es la proporción de positivos reales sobre el total de positivos del experimento. Calculamos también el *recall* de cada vocal, que es el cociente entre las que fueron predichas correctamente sobre el total de la muestra de esa letra (le sumo las que predijo mal). Esto dio como resultado 0,98 para la A, 0,97 para la E, 0,95 para la I, 0,98 para la O y 0,98 para la U. Además, hicimos la matriz de confusión (tabla 2).

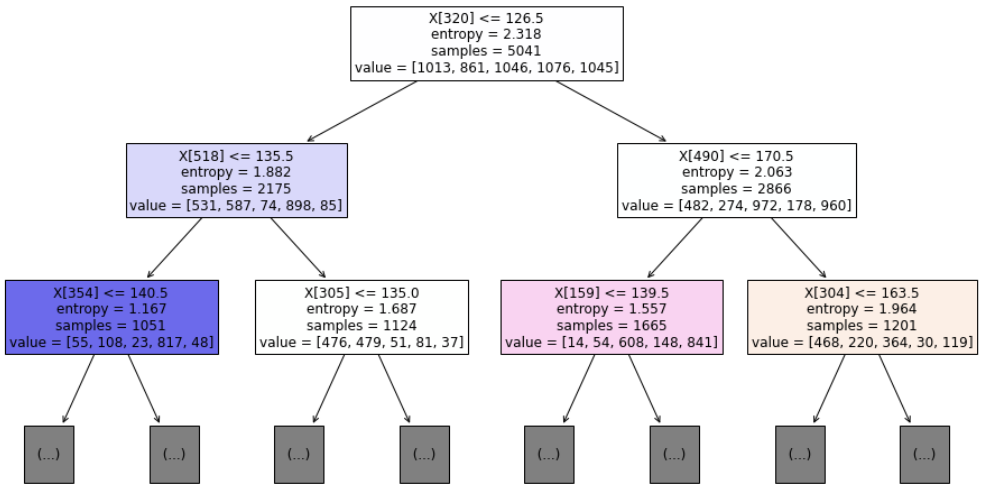

Imagen 8: Primeros 2 niveles del árbol de decisión

		Letra predicha				
		A	E	I	O	U
Letra real	A	111	0	1	1	0
	E	1	93	2	0	0
	I	2	3	110	0	1
	O	1	0	0	118	1
	U	0	0	2	0	114

Tabla 2: Matriz de confusión del árbol de decisión.

Conclusiones

Para la primera parte del trabajo, concluimos que basta con tomar pocos píxeles para distinguir las letras, aunque estos deberían ser los más representativos de cada clase. En cuanto a la cantidad de vecinos, como vimos antes, no necesariamente tiene que ser un valor alto, porque esto puede generar peores resultados que si se tomasen pocos vecinos. Creemos que, por ejemplo, el modelo que toma 3 vecinos y los 8 píxeles con mayor diferencia entre las imágenes promedio es bastante adecuado dado que tiene una precisión de 0.99.

Para la segunda parte del trabajo, las métricas de clasificación ya exhibidas muestran que el modelo armado es bastante eficiente. Podemos ver que la I es la que menor *recall* tiene, es decir que, fue la letra a la que más veces clasificó de forma incorrecta. Esto es consistente con la matriz de confusión que muestra dada una I, el modelo la clasificó como una A dos veces y como una E tres veces. Podemos ver en esta matriz que las dos letras que son más difíciles de distinguir para el modelo son la E y la I.