APRENDIZAJE AUTOMÁTICO

CLASIFICACIÓN DE LETRAS ESCRITAS A MANO

Jorge Figueroa Triana
Diego Martín Alonso Crego
Manuel Patiño Veiga
Alberto Gil Rivas González

Horario de prácticas: Jueves a las 10:00
Aprendizaje Automático
Universidad de A Coruña
Curso 2019 - 2020

Índice

1. Introducción.

El problema que buscamos resolver es la identificación de caracteres del alfabeto inglés escritos a mano. La dificultad de este problema se encuentra en que la caligrafía varía bastante entre las personas, además de que hay letras parecidas entre ellas(como la i mayúscula y la ele). Esta dificultad justifica el uso de un modelo de machine learning frente al matching de plantillas.

Resolver este problema permitiría reconocer documentos escritos a mano automáticamente para transformarlos en formato digital.

Ya existen numerosos métodos automáticos para resolver este problema, pero la mayoría se basan en Aprendizaje Profundo. Nuestra aproximación usará modelos tradicionales de machine learning, usando las técnicas de Redes de Neuronas Artificiales, Support Vector Machine y K-NN.

El objetivo de esta práctica es probar y comparar los diferentes modelos y métodos de aprendizaje automático, analizando sus características y funcionamiento.

Esta memoria constará de los siguientes apartados:

1. Descripción del problema: En este apartado hablaremos del problema en profundidad.
2. Análisis bibliográfico / Estado del arte: Trataremos los trabajos más recientes en clasificación de caracteres escritos, comparando las técnicas que usan.
3. Desarrollo: Expondremos nuestras aproximaciones así como los resultados obtenidos.

Siglas:

- SVM : Support Vector Machine.
- RNA : Red de Neuronas Artificial.
- K-NN : K - Nearest Neighbour.
- EMNIST: Extended Modified National Institute of Standards and Technology database.
- MNIST : Modified National Institute of Standards and Technology database.
- CNN : Convolutional Neural Network.

2. Descripción del problema

Las imágenes usadas para el entrenamiento y testeo de los modelos planteados provienen de la base de datos EMNIST. Esta base de datos es una versión ampliada del MNIST, la cual solo contaba con dígitos.

Las imágenes provienen de una serie de formularios escritos a mano y fueron reescaladas a 28x28 píxeles antes de ser incluidas en la base de datos. Son imágenes en escala de grises, con el fondo en negro y las letras con diferentes intensidades de blanco. La orientación es igual para todas las imágenes.

El conjunto de datos consta de 103600 muestras que representan 26 clases de forma equilibrada, las cuales ya vienen divididas en 88800 para entrenamiento y 14800 para test.

Están clasificados pero sin distinguir entre mayúsculas y minúsculas, por lo que vamos a restringir el problema, por lo menos inicialmente, a letras que tengan mayúsculas y minúsculas parecidas. También escogeremos letras muy diferentes entre sí.

3. Análisis Bibliográfico / Estado del arte

El estado del arte en la clasificación de objetos/imágenes hoy en día gira principalmente alrededor de las Redes Neuronales Convolucionales, un tipo de redes de neuronas profundas que además reduce la dimensionalidad del problema realizando 'convoluciones' por toda la imagen para extraer características. Esta aproximación resulta lo suficientemente precisa como para que normalmente sea la mejor de las alternativas, además de que es la más sencilla ya que no requiere un pre-procesamiento de la imagen o una extracción de descriptores o características 'manual' previa a la fase de entrenamiento.

En cuanto a nuestro problema concreto el caso es el mismo, se puede ver un ejemplo en el trabajo del Perwej y Yusuf [1], pero hay numerosos usando esta técnica.

En cuanto algoritmos de machine learning convencionales u otras aproximaciones que podamos usar para nuestro trabajo encontramos otros artículos. En el de Kothuri S y Annapurna P [2], extraen el "código de freeman" o código de cadena de las imágenes de caracteres binarizadas y usan estadísticas sobre ésta para entrenar su modelo.

En la aproximación de McDonnell MD y Tissera MD [3], usan el método de entrenamiento del 'Extreme Learning Machine', donde los pesos de las capas ocultas no se

modifican desde su iniciación, y demuestran que es igual de efectiva (con una sola capa oculta) que una red Convolucional.

Por último, Lecun Y y Bottou L [4] comparan numerosos métodos de clasificación para este problema, entre las que destacan por su menor tasa de error en el conjunto de test varios tipos de Redes de Neuronas Convolucionales y de Máquinas de Soporte Vectorial. De estas últimas, la Máquina de Soporte Vectorial Virtual alcanza la misma precisión que las CNN, pero los autores destacan que tiene un coste elevado.

Es posible utilizar la aproximación del código en cadena [2]. Utilizaremos RNA, SVM y KNN, como modelo de aprendizaje en esta asignatura.

4. Desarrollo

Para construir el sistema, realizaremos un desarrollo incremental por aproximaciones que consistirá en el aumento gradual del número de letras ha identificar. Para las máquinas de soporte vectorial, en todas las aproximaciones haremos las medias de los resultados de ejecución con cada letra usando la estrategia de "uno contra todos".

4.1. Primera aproximación

Consiste en identificar las letras 'l' y 'm', con una muestra de 2000 imágenes de cada una de las letras. Para la RNA, distribuimos los patrones en 60 % para entrenamiento, 20 % para validación y 20 % para test. Para la SVM, como no se realiza validación, distribuimos los patrones en 70 % para entrenamiento y 30 % para test.

Las características usadas para la RNA y la SVM son la media y la desviación típica de cada uno de los cuatro cuadrantes de la imagen, dando lugar a un total de 8 atributos. Estas características son normalizadas entre sus máximos y sus mínimos. La RNA presenta una arquitectura 8-X-X-1, donde 8 son las neuronas de input, una por cada característica, 1 las de output (solo necesitamos 1 ya que solo necesitamos separar 2 clases), y X-X son las capas ocultas que hemos ido probando.

A continuación mostraremos los resultados obtenidos para la RNA:

Arquitectura		[1]	[8]	[1,1]	[5,4]	[20,20]
Entrenamiento	Precisión	95.12 %	94.5 %	95.08 %	94.85 %	92.625 %
	σ	0.317	0.183	0.229	0.726	0.483
Validación	Precisión	95.12 %	94.5 %	95.08 %	94.85 %	92.625 %
	σ	0.618	0.389	0.714	1.212	1.034
Test	Precisión	95.31 %	94.93 %	94.75 %	94.06 %	91.12 %
	σ	1.060	0.560	0.486	0.717	0.598

Tabla 1: Resultado de RNA

Como observamos en la tabla 1 , en este problema funcionan mejor las RNA con arquitecturas simples. Esto es debido a que de momento se trata de un problema simple, ya que intentamos diferenciar dos letras muy diferentes entre sí, además de que no tenemos demasiados inputs. Cuanto más compleja es la arquitectura, peores resultados da.

Las RNA serían una mejor opción para clasificar un mayor número de clases con un mayor número de características. En este caso clasificamos solo 2 y muy diferentes, por lo que arquitecturas muy complejas no nos sirven. Como veremos a continuación, generalmente preferiremos utilizar SVM ante estos casos. Los diferentes hiper parámetros que variaremos para la SVM son: γ (gamma), C y la función de kernel.

Kernel de función de base radial						
Parámetros		$\gamma = \frac{1}{N_{dim}}$, $C = 1$	$\gamma = 1$, $C = 1000$	$\gamma = 1$, $C = 0{,}001$	$\gamma = 1000$, $C = 1$	$\gamma = 10$, $C = 2$
Entrenamiento	Precisión	95.15 %	98.55 %	50.64 %	100 %	97.98 %
	σ	0.170	0.312	0.208	0.0	0.170
Test	Precisión	94.76 %	97.63 %	48.50 %	69.4 %	97.73 %
	σ	0.607	0.844	0.485	8.315	0.450

Tabla 2: Resultado de SVM con Kernel de base radial aproximación 1

Kernel Polinómico						
Parámetros		$\gamma = \frac{1}{N_{dim}}$, $C = 1$	$\gamma = 1$, $C = 1000$	$\gamma = 1$, $C = 0{,}001$	$\gamma = 1000$, $C = 1$	$\gamma = 10$, $C = 2$
Entrenamiento	Precisión	92.04 %	98.34 %	89.65 %	98.75 %	98.47 %
	σ	0.506	0.184	0.336	0.852	0.129
Test	Precisión	92.26 %	97.2 %	88.99 %	96.16 %	97.5 %
	σ	0.480	0.217	1.073	0.565	0.754

Tabla 3: Resultado de SVM con Kernel polinómico aproximación 1

Kernel Lineal						
Parámetros		$\gamma = \frac{1}{N_{dim}}$, $C = 1$	$\gamma = 1$, $C = 1000$	$\gamma = 1$, $C = 0{,}001$	$\gamma = 1000$, $C = 1$	$\gamma = 10$, $C = 2$
Entrenamiento	Precisión	95.01 %	94.89 %	50.28 %	95.15 %	95.11 %
	σ	0.482	0.464	0.252	0.216	0.274
Test	Precisión	94.9 %	95.83 %	49.33 %	94.9 %	95.33 %
	σ	1.038	0.9204	0.589	0.480	0.957

Tabla 4: Resultado de SVM con Kernel lineal aproximación 1

Kernel Sigmoidal						
Parámetros		$\gamma = \frac{1}{N_{dim}}$, $C = 1$	$\gamma = 1$, $C = 1000$	$\gamma = 1$, $C = 0{,}001$	$\gamma = 1000$, $C = 1$	$\gamma = 10$, $C = 2$
Entrenamiento	Precisión	94.64 %	17.05 %	50.75 %	50.55 %	48.11 %
	σ	0.290	0.635	0.472	0.283	5.355
Test	Precisión	93.96 %	17.0 %	48.23 %	48.7 %	46.53 %
	σ	1.043	0.589	1.103	0.660	5.259

Tabla 5: Resultado de SVM con Kernel sigmoidal aproximación 1

Como observamos en la tabla anterior, escoger un valor demasiado pequeño de C aumenta el margen del SVM, es decir, la tolerancia al ruido. Escoger un valor demasiado pequeño no es beneficioso, ya que podría generarnos demasiados errores. Por la contra, escoger un valor demasiado alto tampoco es bueno ya que podría producirse sobreajuste y se generaría demasiada sensibilidad al ruido, un patrón demasiado alejado del resto de patrones de su clase influiría bastante a la hora de clasificar.

En este caso, hemos probado con varios parámetros desproporcionados para ver el efecto que producen. En el caso de C muy alto, vemos que no se ve plasmado el sobreajuste en los resultados. Esto es debido al trabajar con solo 2 clases fácilmente diferenciables y con poco ruido, pero al aumentar el número de clases debería dar menos precisión en test, por lo que no sería una opción recomendable, pero en este caso concreto podría serlo, asumiendo que no vamos a tener patrones que generen mucho ruido.

Utilizando la opción de gamma = "scale" ($\frac{1}{N_{dim}}$), tampoco se producen resultados destacables, exceptuando el kernel sigmoidal(Tabla 5), en el que es el mejor valor, y observamos que es muy sensible a los valores desproporcionados. En este caso nos quedaríamos con esta opción.

Al final, por ensayo y error, llegamos a una configuración bastante aceptable, en general con todos los kernel, de gamma = 10 y C = 2, con cierta tolerancia a errores y que

no debería producir sobreajuste. Esta opción la utilizaríamos con el resto de kernels, el de base radial, polinómico y lineal.

En cuanto a qué kernel escoger, en este caso escogeríamos el de función de base radial con la configuración gamma = 10 y C = 2, dado que presenta menor desviación típica que la misma configuración en el kernel polinómico, aunque presentan resultados similares. En aproximaciones futuras esperamos que los kernel funcionen de forma diferente y consideraremos nuevas configuraciones.

Parámetros		Mejor SVM	Mejor RNA
		Kernel de base radial, $\gamma = 10$, $C = 2$	[1,1]
Test	Precisión	97.73 %	94.75 %
	σ	0.450	0.486

Tabla 6: Mejor resultado con cada técnica aproximación 1

En el caso planteado con 2 clases, escogeríamos este último caso de SVM sobre el de RNA. Ya explicamos anteriormente que no tiene mucho sentido utilizar una RNA en casos con tan pocas clases y tan fácilmente diferenciables, por lo que el SVM presenta mejor rendimiento tanto a la hora de clasificar como rendimiento computacional, ya que se ejecuta en mucho menor tiempo.

Ya que la precisión alcanzada en esta aproximación es muy alta, para la siguiente aproximación nos limitaremos a añadir una nueva letra al conjunto de patrones de entrada, en este caso la letra 'o'. Esperamos una peor precisión ya que las características que usamos son bastante simples.

4.2. Segunda aproximación

Añadiremos a nuestro sistema la identificación de la letra 'o', y realizaremos las mismas pruebas que antes con las mismas características. Estudiaremos cómo se comportan la RNA y el SVM ante un aumento de complejidad del problema sin variar las características. Además, añadiremos una nueva técnica de aprendizaje por instancias: KNN.

La RNA presenta la misma arquitectura que en la aproximación, pero esta vez con 3 neuronas de output, ya que esta vez necesitamos distinguir entre 3 clases, por lo que utilizaremos una arquitectura 8-X-X-2. A continuación los resultados de las RNA:

Arquitectura		[1]	[8]	[1,1]	[5,4]	[20,20]
Entrenamiento	Precisión	70.04 %	85.82 %	68.82 %	82.44 %	83.68 %
	σ	0.377	0.660	0.829	5.914	0.980
Validación	Precisión	69.51 %	85.71 %	68.10 %	83.04 %	84.25 %
	σ	1.152	1.047	1.590	6.133	1.346
Test	Precisión	69.22 %	85.89 %	69.24 %	81.60 %	83.51 %
	σ	1.543	1.431	1.164	6.715	0.984

Tabla 7: Resultado de RNA aproximación 2

Como se puede ver (Tabla 7) inmediatamente los resultados son más bajos que en la primera aproximación al añadir una nueva clase a separar. Además, en las RNA ya no es suficiente con una arquitectura tan simple como las de la primera aproximación, con una neurona en una o dos capas. En esta aproximación parecen destacar las arquitecturas con una sola capa y un mayor número de neuronas, como la [8].

Kernel de función de base radial						
Parámetros		$\gamma = \frac{1}{N_{dim}}$, $C = 1$	$\gamma = 1$, $C = 1000$	$\gamma = 1$, $C = 0{,}001$	$\gamma = 1000$, $C = 1$	$\gamma = 10$, $C = 2$
Entrenamiento	Precisión	85.98 %	95.15 %	66.55 %	100.0 %	94.46 %
	σ	0.320	0.200	0.274	0.0	0.300
Test	Precisión	85.51 %	93.44 %	66.93 %	67.02 %	93.0 %
	σ	1.076	0.773	0.641	0.927	0.538

Tabla 8: Resultado de SVM con Kernel de base radial aproximación 2

En la tabla 8, en comparación con la tabla 2 de la anterior aproximación, se ve una reducción de precisión generalizada. Tan solo se mantiene con una precisión y desviación típica aceptable la SVM con parámetros $\gamma = 10$, $C = 2$

Kernel Polinómico						
Parámetros		$\gamma = \frac{1}{N_{dim}}$, $C = 1$	$\gamma = 1$, $C = 1000$	$\gamma = 1$, $C = 0{,}001$	$\gamma = 100$, $C = 1$	$\gamma = 10$, $C = 2$
Entrenamiento	Precisión	83.26 %	93.59 %	75.84 %	93.53 %	93.36 %
	σ	0.409	0.241	1.369	0.185	0.206
Test	Precisión	84.35 %	92.55 %	74.75 %	93.15 %	92.71 %
	σ	1.128	0.561	3.089	0.596	0.468

Tabla 9: Resultado de SVM con Kernel Polinómico aproximación 2

En la tabla 9, en comparación con la tabla 3 de la anterior aproximación, vemos de nuevo una reducción de precisión generalizada. Tan solo se mantiene con una precisión

y desviación típica aceptable la SVM con parámetros $\gamma = 10$, $C = 2$. Es destacable que para este kernel los valores de gamma altos incrementan el tiempo de computación en gran medida, de manera que el valor de gamma que solemos probar(1000) hacía que el entrenamiento no acabase o tardase demasiado.

Kernel Lineal						
Parámetros		$\gamma = \frac{1}{N_{dim}}$, $C = 1$	$\gamma = 1$, $C = 1000$	$\gamma = 1$, $C = 0{,}001$	$\gamma = 1000$, $C = 1$	$\gamma = 10$, $C = 2$
Entrenamiento	Precisión	85.14 %	86.61 %	66.77 %	85.48 %	85.96 %
	σ	0.762	0.398	0.300	0.554	0.658
Test	Precisión	85.37 %	86.22 %	66.42 %	84.97 %	85.57 %
	σ	1.193	0.971	0.700	1.294	1.412

Tabla 10: Resultado de SVM con Kernel Lineal aproximación 2

En la tabla 10, en comparación con la tabla 4 de la anterior aproximación, vemos de nuevo una reducción de precisión generalizada. Donde previamente la precisión era prácticamente aleatoria para los parámetros $\gamma = 1$, $C = 0{,}001$, ahora aumenta ligeramente debido a que al haber una nueva clase, habrá más fallos/ruido entre ellas que serán permitidos por el valor bajo de C.

Kernel Sigmoidal						
Parámetros		$\gamma = \frac{1}{N_{dim}}$, $C = 1$	$\gamma = 1$, $C = 1000$	$\gamma = 1$, $C = 0{,}001$	$\gamma = 1000$, $C = 1$	$\gamma = 10$, $C = 2$
Entrenamiento	Precisión	83.10 %	33.79 %	66.48 %	66.87 %	66.88 %
	σ	0.274	0.358	0.294	0.294	0.394
Test	Precisión	84.13 %	33.17 %	67.08 %	66.17 %	66.15 %
	σ	0.395	1.870	0.686	0.687	0.921

Tabla 11: Resultado de SVM con Kernel sigmoidal aproximación 2

En la tabla 11, en comparación con la tabla 5 de la anterior aproximación, vemos un ligero aumento de precisión, pero este kernel sigue dando malos resultados para este problema.

A continuación veremos los resultados que da el KNN para esta aproximación con distintos valores de K. Los valores son impares para evitar empates.

Parámetros		K=1	K=3	K=5	K=7	K=51
Entrenamiento	Precisión	100 %	99.15 %	98.89 %	98.76 %	96.87 %
	σ	0.0	0.126	0.121	0.147	0.764
Test	Precisión	98.57 %	98.55 %	98.47 %	98.24 %	96.855 %
	σ	0.539	0.484	0.554	0.616	0.764

Tabla 12: Resultado de KNN en la aproximación 2

Esta técnica tiene una precisión muy elevada para la clasificación de tres letars. Como se puede observar, para un problema de este tamaño una K mucho menor conlleva más precisión. Es destacable que para este número de inputs la técnica de KNN tiene un tiempo de ejecución bastante corto.

Parámetros		Mejor SVM	Mejor RNA	Mejor KNN
		Kernel de base radial, $\gamma = 1$, $C = 1000$	[8]	K = 3
Test	Precisión	93.44 %	85.89 %	98.55
	σ	0.773	1.431	0.484

Tabla 13: Mejor resultado con cada técnica aproximación 2

El método elegido en esta aproximación sería KNN con el número de vecinos o K = 3, ya que tiene una precisión muy alta y una desviación típica aceptable. Dado que esta técnica tiene una precisión suficientemente alta, para la siguiente aproximación incrementaremos el número de clases a separar en más de uno.

4.3. Tercera aproximación

En esta aproximación realizaremos las mismas pruebas que antes, pero con algunos cambios. Esta vez tomaremos nuevas características, dividiremos cada cuadrante en 4 subcuadrantes, quedando un grid de 4x4 en vez de 2x2 como anteriormente. Para cada región, seguiremos tomando las mismas medidas, la media y desviación típica de los valores de los píxeles pertenecientes a dicha región. Con esto, esperamos que aporte una mayor precisión, ayudándonos a obtener un resultado aceptable al añadir más clases. Además, añadiremos una nueva técnica, KNN.

Las siguientes pruebas se han realizado con una muestra de 2000 patrones. En el caso de la RNA, utilizamos un 20 % de estos patrones para validación y otro 20 % para test.

En el SVM utilizamos un 30 % para test. Tanto en RNA como en SVM realizaremos un total de 10 ejecuciones. En KNN utilizaremos un 20 % de validación y 50 ejecuciones.

En la aproximación anterior escogimos parámetros algo desproporcionados para comprobar cómo se comportaban la RNA y el SVM. En esta aproximación, sin embargo, probaremos un mayor número de arquitecturas con parámetros dentro de un rango aceptable y mostraremos las arquitecturas más representativas.

En esta aproximación, la RNA presenta una arquitectura 32-X-X-3. A continuación ejecutamos la RNA con las nuevas características y mostramos las arquitecturas más representativas:

Arquitectura		[30,30]	[15]	[14,14]	[5,4]	[40,10]
Entrenamiento	Precisión	97.36 %	98.01 %	98.04 %	98.47 %	97.89 %
	σ	0.315	0.402	0.384	0.258	0.297
Validación	Precisión	97.63 %	97.38 %	97.88 %	97.98 %	97.28 %
	σ	0.489	0.738	0.517	0.492	0.908
Test	Precisión	97.2 %	97.9 %	97.7 %	97.3 %	96.85 %
	σ	0.695	0.502	0.654	0.864	1.001

Tabla 14: Resultado de RNA en la aproximación 3

En los resultados de esta aproximación (Tabla 14) hemos descartado las arquitecturas mas simples, como la [1], [1,1], [2,2]... ya que daban unos resultados inferiores a las demás, ya que en nuestro problema ya va aumentando la complejidad. Ahora, las arquitecturas más complejas presentan un rendimiento bastante mayor.

Generalmente, las arquitecturas que mejor nos han funcionado son las que presentan un número de neuronas entre 10 y 18 aproximadamente. Presentan un rendimiento bastante similar tanto las arquitecturas con una sola capa oculta como las de 2, como podemos apreciar al comparar la [15] con la [14,14]. También hemos probado arquitecturas como [12], [10], [10,10], [18], todas con un rendimiento prácticamente similar a [15] y [14,14], alrededor de un 0.1 % inferior. Otras arquitecturas probadas, como por ejemplo [8] y [8,8], también producen resultados muy similares a los expuestos anteriormente y, dependiendo de la ejecución, su rendimiento es más cercano o no al de las demás arquitecturas, por lo que el rango de 10 a 18 neuronas es orientativo y no siempre se comporta de la misma manera, pero lo que sí que podemos concluir es que las arquitecturas con un número de neuronas en la capa oculta cercano a la decena tienen un buen rendimiento.

También apreciamos que las arquitecturas extremadamente complejas suponen una pérdida de rendimiento, como vemos en las arquitecturas [30,30] y [40,10]. Lo mismo sucede con otras arquitecturas como [40,40], [45], [10,30], que muestran un porcentaje de precisión entre 96.9 % y 97.2 %. Otras arquitecturas con doble capa oculta, como [15,6], presentan

un rendimiento casi idéntico a su versión de capa única [15], por lo que la elección de una segunda capa oculta no parece tener demasiado efecto mientras no se escojan capas desproporcionadas que puedan generar cuello de botella, como [15,30], en cuyo caso empeora el rendimiento llegando nuevamente a 97.2 %.

Por lo general, en el problema planteado en esta aproximación, escogeremos una arquitectura con capas con un número de neuronas alrededor de 12 - 15. En este caso, las que mejor resultado dan son [15,6] y [15], pero entre estas dos escogeríamos la 15 ya que presenta una desviación típica en test ligeramente menor, alrededor de un 0.15 menor.

Pasamos al SVM. En este caso, al tener más de 2 clases, separaremos cada una de ellas del resto y haremos la media de los resultados obtenidos. En comparación a la aproximación 1, esta vez utilizaremos parámetros dentro de un rango más razonable intentando optimizar los resultados. Los resultados obtenidos fueron los siguientes:

Kernel Lineal						
Parámetros		$\gamma = \frac{1}{N_{dim}}$, $C = 1$	$\gamma = \frac{1}{N_{dim}}$, $C = 3$	$\gamma = \frac{1}{N_{dim}}$, $C = 0{,}5$	$\gamma = 5$, $C = 1$	$\gamma = 4$, $C = 2$
Entrenamiento	Precisión	98.86 %	99.01 %	98.73 %	98.88 %	98.89 %
	σ	0.023	0.057	0.161	0.054	0.086
Test	Precisión	98.35 %	98.46 %	98.05 %	98.28 %	98.44 %
	σ	0.497	0.587	0.685	0.388	0.592

Tabla 15: Resultado de SVM con Kernel Lineal aproximación 3

Kernel Polinómico						
Parámetros		$\gamma = \frac{1}{N_{dim}}$, $C = 1$	$\gamma = \frac{1}{N_{dim}}$, $C = 3$	$\gamma = \frac{1}{N_{dim}}$, $C = 0{,}5$	$\gamma = 10$, $C = 2$	$\gamma = 4$, $C = 2$
Entrenamiento	Precisión	96.82 %	97.70 %	95.87 %	100 %	100 %
	σ	0.019	0.054	0.389	0	0
Test	Precisión	96.56 %	97.54 %	95.73 %	98.22 %	98.19 %
	σ	0.542	0.737	1.138	0.629	0.677

Tabla 16: Resultado de SVM con Kernel Polinómico aproximación 3

Kernel Base Radial						
Parámetros		$\gamma = \frac{1}{N_{dim}}$, $C = 1$	$\gamma = \frac{1}{N_{dim}}$, $C = 3$	$\gamma = \frac{1}{N_{dim}}$, $C = 0{,}5$	$\gamma = 5$, $C = 1$	$\gamma = 4$, $C = 2$
Entrenamiento	Precisión	98.2 %	98.52 %	97.85 %	99.94 %	99.99 %
	σ	0.06	0.067	0.041	0.036	0.029
Test	Precisión	98.78 %	98.47 %	97.47 %	91.05 %	94.38 %
	σ	0.569	0.556	0.729	1.401	1.329

Tabla 17: Resultado de SVM con Kernel Base Radial aproximación 3

Kernel Sigmoidal						
Parámetros		$\gamma = \frac{1}{N_{dim}}$, $C = 1$	$\gamma = \frac{1}{N_{dim}}$, $C = 3$	$\gamma = \frac{1}{N_{dim}}$, $C = 0{,}5$	$\gamma = 1$, $C = 5$	$\gamma = 10$, $C = 2$
Entrenamiento	Precisión	97.52 %	98.11 %	97.02 %	51.79 %	66.88 %
	σ	0.049	0.037	0.072	2.923	0.148
Test	Precisión	97.55 %	97.93 %	96.99 %	51.66 %	66.18 %
	σ	0.785	0.661	0.699	4.476	1.312

Tabla 18: Resultado de SVM con Kernel Sigmoidal aproximación 3

Vamos a comenzar analizando los resultados del kernel con resultados más simples, el Sigmoidal. Como observamos (Tabla 15), el kernel Sigmoidal aparentemente solo responde a $\gamma = \frac{1}{N_{dim}}$, de forma que solo podemos modificar el parámetro C para intentar obtener unos resultados buenos, escogiendo entre aumentar el margen de error (C pequeño) o minimizar los errores (C grande). Aún así, la variación de C no parece tener un impacto muy grande y los resultados obtenidos parecen ser mejorables utilizando otros kernels.

Por otra parte, el kernel Lineal (Tabla 16) parece no ser muy sensible a la variación de ambos parámetros, γ y C. En todos los pares de parámetros seleccionados muestra un porcentaje de precisión en test bastante bueno y no muestra un comportamiento muy exagerado en cuanto a permisión de errores, por lo que a priori el kernel lineal parece que nos garantiza un porcentaje de acierto aceptable.

En cuanto al kernel Polinómico (Tabla 17), podemos observar que muestra mucha sensibilidad al parámetro γ, si lo aumentamos un mínimo parece que minimiza el error, aunque produce unos resultados aceptables del 98.2 %. Si utilizamos el valor auto de gamma, $\gamma = \frac{1}{N_{dim}}$, y seleccionamos valores de C relativamente bajos, como 0.5 o 1, obtenemos peores resultados que en otros kernels, por lo que parece que este kernel tambien es más sensible a variaciones del parámetro C. Si queremos obtener resultados con mayor precisión, tenemos que aumentar C. Como observamos, si aumentamos C a 3, obtenemos un porcentaje de acierto en test del 97.5 %, también mejorable por otros kernels como el lineal.

El kernel Base Radial (Tabla 18) parece producir los mejores resultados en test con $\gamma = \frac{1}{N_{dim}}$ y C = 1. Produce unos resultados muy similares al aumentar C a 3, y la tolerancia a errores es ligeramente notable al disminuir C a 0.5, pero no muy pronunciada. Al utilizar otros valores de γ, como 5 y 4, se comporta de forma similar al kernel Polinómico, parece que minimiza los errores en test de forma muy pronunciada, pero en este caso produce resultados bastante peores. Parece que estos resultados podrían arreglarse aumentando C para tolerar menos errores pero resultaría en un modelo demasiado sobreajustado y no es lo que buscamos.

En conclusión con respecto al SVM, el kernel que produce mejores resultados en

este problema es el kernel Polinómico con $\gamma = \frac{1}{N_{dim}}$ y C = 1. A pesar de que este kernel sea el que mejores resultados produce, el kernel Lineal parece mucho más estable y produce siempre resultados aceptables, por lo que también es un kernel que se debería de tener en cuenta en este problema. El resto de kernels de momento parecen demasiado sensible a los parámetros por lo que decidimos descartarlos.

Parámetros		K=1	K=3	K=5	K=7	K=200
Entrenamiento	Precisión	100 %	99.12 %	98.89 %	98.72 %	93.48 %
	σ	0.0	0.131	0.122	0.159	0.301
Test	Precisión	98.62 %	98.595 %	98.56 %	98.395 %	93.48 %
	σ	0.487	0.522	0.585	0.616	1.347

Tabla 19: Resultado de KNN en la aproximación 3

Como observamos en la tabla 19, el KNN presenta sus mejores resultados en valores de K pequeños. Esto es debido a que estamos ante un problema todavía simple y estamos utilizando un dataset que prácticamente no tiene ruido. Cuando introduzcamos nuevas clases que presenten mayores similitudes, veremos que necesitaremos establecer valores mayores de K para que el KNN produzca buenos resultados y, en el caso de utilizar un dataset que tuviera ruido, no obtendríamos estos resultados con unos valores tan bajos de K y tendríamos que subirlo o plantear otras formas de intentar no incluir estos valores en la selección. A pesar de esto, en este caso concreto presenta unos buenos resultados.

También es destacable que los resultados son extremadamente similares a los de la aproximación 2 en la tabla 12. Esto podría ser debido a que las nuevas variables, al ser una división de las anteriores, no separen lo suficiente los patrones en el espacio, con lo que mantendrían las relaciones de distancia que se usan en la clasificación por KNN.

Parámetros		Mejor SVM	Mejor RNA	Mejor KNN
		Kernel de base radial, $\gamma = \frac{1}{N_{dim}}$, $C = 1$	[15]	K=1
Test	Precisión	98.78 %	97.9 %	98.62 %
	σ	0.569	0.502	0.487

Tabla 20: Mejor resultado con cada técnica aproximación 3

Como vemos, el mejor resultado del SVM presenta mejor resultado que el de RNA con el coste de que, a priori, la elección de los parámetros tiene mucho más peso que la elección de la arquitectura en una RNA. Por la contra, la elección de características parece ser lo que más impacto tiene en el rendimiento de la RNA, por lo menos en problemas simples como el tratado en esta aproximación.

En conclusión, el estudio de ambas técnicas en la aproximación 3 nos ha permitido comprobar que una buena extracción de características puede tener bastante impacto. Simplemente extrayendo nuevas características no solo hemos podido seguir proporcionando un porcentaje de acierto en test aceptable con ambas técnicas al añadir una nueva clase, sino que además hemos mejorado ese porcentaje de test.

En lo que respecta al KNN, vemos que nos ha ofrecido buenos resultados, pero esto es debido a que nuestra base de datos no posee ruido y estamos comparando clases muy distintas entre sí, por lo que estos resultados no son muy fiables, son buenos para esta aproximación pero no esperamos que sean así para la siguiente, en la que añadiremos nuevas clases que generen más confusión. En conclusión, el SVM nos proporciona unos resultados muy similares, incluso mejores, y nos proporciona mayor fiabilidad en estos resultados también a largo plazo, por lo que preferiremos esta técnica.

Para la próxima aproximación añadiremos nuevas clases, esta vez más parecidas entre sí, con el objetivo de estudiar qué ocurre al desestabilizar un poco el problema. Además, extraeremos nuevas características para seguir manteniendo un porcentaje de acierto aceptable e intentar solucionar el problema que introducimos. Comprobaremos si las técnicas anteriores siguen comportándose de la misma forma al añadirle algo más de complejidad al problema.

4.4. Cuarta aproximación

Ya que en la aproximación previa obtuvimos unos resultados de precisión bastante elevados, en esta aproximación añadiremos 6 nuevas clases: ‘p’, ‘q’, ‘u’, ‘v’, ‘w’ y ‘z’. También añadiremos nuevos parámetros de entrada, el número de píxeles con valor distinto de cero por cuadrante y las veces que ‘corta’ la letra al iterar tanto vertical como horizontalmente por las líneas de la imagen. A continuación un ejemplo visual (Figura 1) de esta nueva característica, en el que se corta la letra O dos veces:

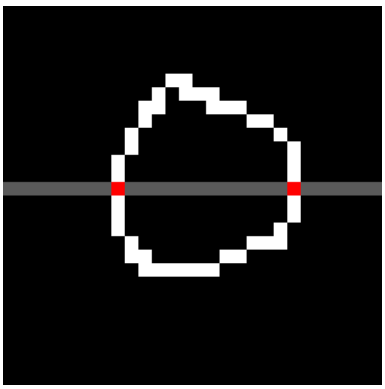

Figura 1: La letra O se corta 2 veces horizontalmente.

La RNA presenta una arquitectura 52-X-X-9. A continuación mostramos los nuevos resultados.

Arquitectura		[15]	[15,6]	[5,4]	[30]	[30,15]
Entrenamiento	Precisión	91.12 %	86.07 %	75.74 %	91.56 %	88.88 %
	σ	1.279	1.955	4.027	0.876	2.383
Validación	Precisión	87.75 %	83.0 %	72.25 %	88.92 %	85.75 %
	σ	1.495	3.535	3.297	0.858	2.776
Test	Precisión	88.27 %	82.9 %	73.0 %	88.42 %	85.42 %
	σ	1.672	1.410	4.593	1.867	3.071

Tabla 21: Resultado de RNA en la aproximación 4

Como podemos apreciar de los resultados de la tabla 21, las arquitecturas más complejas tienen un mayor problema a la hora de clasificar las letras. También se puede contemplar como la RNA obtiene unos valores menos favorables conforme aumentamos el número de letras a clasificar, lo que nos permite comprobar que las RNA clásicas con pocas capas ocultas no son el mejor método para este problema. También observamos que las arquitecturas con pocas neuronas funcionan muy mal en comparación con las demás, y las arquitecturas con capas ocultas con número de neuronas muy dispar entre ellas también ocasionan una pérdida de rendimiento.

A continuación mostraremos los resultados del KNN:

Parámetros		K=1	K=3	K=5	K=7	K=20
Entrenamiento	Precisión	100.0 %	92.71 %	91.96 %	91.04 %	87.98 %
	σ	0.0	0.367	0.354	0.387	0.429
Test	Precisión	87.52 %	88.27 %	87.96 %	88.52 %	86.27 %
	σ	1.365	1.346	1.409	1.629	1.437

Tabla 22: Resultado de KNN en la aproximación 4

Como observamos en la tabla 21, el podemos apreciar que da mejores resultados para k = 7, debido a añadir diferentes clases y aumentar por consecuente la complejidad. Los resultados en esta aproximación ya no son tan buenos como en la anterior, esto es debido a la introducción de nuevas clases muy parecidas entre sí, como por ejemplo la 'u' y la 'v', o la 'p' y 'q'. La 'Q' mayúscula también puede ocasionar conflictos con la 'O'. Esto nos hace pensar que tendríamos que utilizar otros métodos para aumentar la precisión del KNN en este problema, por lo que generalmente preferiríamos la RNA antes que este método al introducir clases similares entre sí.

Kernel Lineal						
Parámetros		$\gamma = \frac{1}{N_{dim}}$, $C = 1$	$\gamma = \frac{1}{N_{dim}}$, $C = 3$	$\gamma = 1$, $C = 0{,}1$	$\gamma = 5$, $C = 1$	$\gamma = 4$, $C = 2$
Entrenamiento	Precisión	97.86 %	98.27 %	96.33 %	97.85 %	98.08 %
	σ	0.095	0.097	0.081	0.102	0.085
Test	Precisión	97.44 %	97.43 %	95.96 %	97.46 %	97.58 %
	σ	0.609	0.483	0.611	0.502	0.546

Tabla 23: Resultado de SVM con Kernel Lineal aproximación 4

Kernel Polinómico						
Parámetros		$\gamma = \frac{1}{N_{dim}}$, $C = 1$	$\gamma = \frac{1}{N_{dim}}$, $C = 3$	$\gamma = 1$, $C = 0{,}1$	$\gamma = 5$, $C = 1$	$\gamma = 4$, $C = 2$
Entrenamiento	Precisión	89.42 %	94.33 %	99.99 %	100.0 %	100.0 %
	σ	0.073	0.107	0.011	0.0	0.0
Test	Precisión	89.22 %	94.24 %	97.92 %	97.82 %	97.95 %
	σ	1.223	1.004	0.501	0.517	0.507

Tabla 24: Resultado de SVM con Kernel Polinómico aproximación 4

Kernel Base Radial						
Parámetros		$\gamma = \frac{1}{N_{dim}}$, $C = 1$	$\gamma = \frac{1}{N_{dim}}$, $C = 3$	$\gamma = 1$, $C = 0{,}1$	$\gamma = 5$, $C = 1$	$\gamma = 4$, $C = 2$
Entrenamiento	Precisión	95.43 %	96.76 %	99.99 %	99.99 %	100.0 %
	σ	0.137	0.117	0.010	0.012	0.0
Test	Precisión	95.24 %	96.31 %	97.58 %	90.04 %	91.52 %
	σ	0.745	0.627	0.526	0.978	1.023

Tabla 25: Resultado de SVM con Kernel Base Radial aproximación 4

Kernel Sigmoidal						
Parámetros		$\gamma = \frac{1}{N_{dim}}$, $C = 1$	$\gamma = \frac{1}{N_{dim}}$, $C = 3$	$\gamma = 1$, $C = 0{,}1$	$\gamma = 5$, $C = 1$	$\gamma = 4$, $C = 2$
Entrenamiento	Precisión	92.67 %	95.77 %	89.27 %	88.88 %	88.75 %
	σ	0.154	0.111	0.206	0.115	0.150
Test	Precisión	92.45 %	95.73 %	89.32 %	88.89 %	89.19 %
	σ	1.044	0.641	1.108	1.029	1.042

Tabla 26: Resultado de SVM con Kernel Sigmoidal aproximación 4

Como podemos apreciar de la aproximación del SVM, el kernel lineal (Tabla 23) nos da los mejores resultados y sigue siendo el kernel más estable, aún incrementando

la complejidad del problema considerablemente con respecto a otras aproximaciones. Los resultados siguen la misma dirección que en la aproximación anterior en lo que respecta a sobreajuste, permisión de errores, etc. por lo que no vamos a analizar tan detalladamente cada kernel de forma individual.

Como observamos, de forma general, esta técnica presenta unos resultados mucho mejores que el resto de técnicas al añadir complejidad al problema, por lo que no cabe duda que será la técnica que escojamos. Aún así, a continuación mostramos la tabla de comparación entre los mejores resultados con cada técnica:

Parámetros		Mejor SVM	Mejor RNA	Mejor KNN
		Kernel lineal, $\gamma = 4$, $C = 2$	[30]	K=7
Test	Precisión	97.58 %	88.42 %	88.52 %
	σ	0.546	1.867	1.629

Tabla 27: Mejor resultado con cada técnica aproximación 4

De nuevo, la máquina de soporte vectorial supera con ventaja a la mejor RNA y al mejor KNN, tanto en precisión de test como en desviación típica. Sería la técnica a elegir para esta aproximación.

Para la aproximación final compararemos una técnica de Deep Learning con CNN y una red de neuronas artificial en la que cada píxel es una característica.

4.5. Aproximación final

Para la aproximación final, comenzaremos ejecutando una RNA convencional usando 26 clases para compararla posteriormente con Deep Learning. Esta RNA recibirá cada píxel en nuestra imagen de 28x28 como una entrada, teniendo un total de 784 entradas. Como la ejecución con este modelo es extremadamente lenta, reducimos considerablemente el número de patrones a 200 por letra. Usando las 26 letras del alfabeto tenemos un total de 5200 patrones. Los resultados con distintas arquitecturas fueron los siguientes:

Arquitectura		[15]	[15,6]	[5,4]	[30]	[30,15]
Entrenamiento	Precisión	57.10 %	36.95 %	22.17 %	67.04 %	56.63 %
	σ	1.616	4.486	4.067	1.271	2.112
Validación	Precisión	48.90 %	31.92 %	19.41 %	55.88 %	47.23 %
	σ	1.960	3.512	3.715	1.481	1.548
Test	Precisión	49.09 %	30.83 %	19.46 %	55.39 %	46.59 %
	σ	2.833	4.753	3.423	1.248	1.376

Tabla 28: Resultado de RNA con entrada por píxel en la aproximación final.

Como se pueda apreciar la precisión en test es bastante baja para todas las arquitecturas lo que nos verifica que una RNA tradicional no es capaz de gestionar tantos inputs.

A continuación realizaremos una aproximación con Deep Learning aplicado a nuestra base de datos de letras. Las entradas serán de nuevo los píxeles de la imagen y han sido previamente normalizadas entre máximo y mínimo. Las capas usadas en el modelo serán 3 capas de convolución intercaladas con 3 capas MaxPool, que después pasarán a una capa totalmente conectada y finalmente a una capa softmax. Esta arquitectura puede observarse en la siguiente figura:

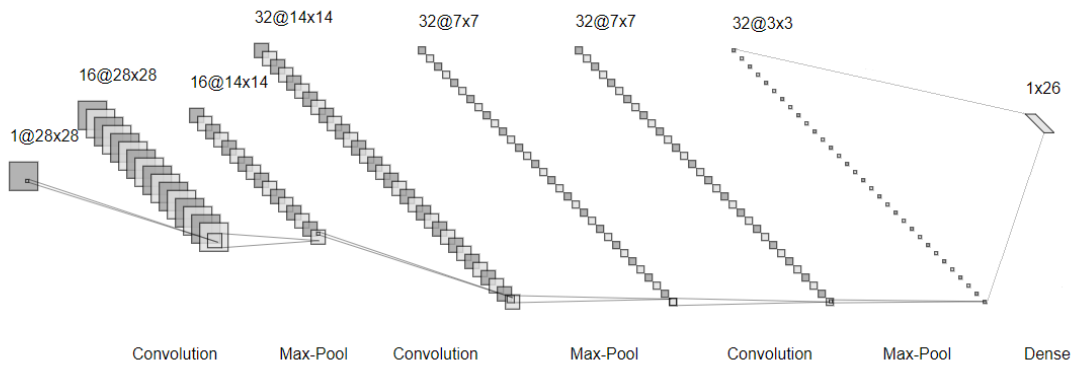

Figura 2: Arquitectura del modelo de Deep Learning.

Comenzaremos usando sólo 13 letras (ABCLMNOPQUVWZ) en lugar de las 26 posibles, y 2000 patrones por letra. Los resultados del mejor modelo obtenido son los siguientes:

Ciclo = 82	Precisión
Entrenamiento	99.9041 %
Test	96.6737 %

Tabla 29: Resultado de CNN con imágenes de 13 letras distintas.

Se puede observar en la tabla 29 que la precisión de la CNN es extremadamente elevada para 13 letras. Además, el tiempo computacional es mucho menor en relación a la cantidad de patrones que puede recibir esta red. Por comparación, la RNA usada anteriormente usó 200 patrones por letra y tuvo un tiempo de ejecución similar a la CNN que pudo usar 10 veces más patrones por letra (2000).

A continuación usaremos las 26 letras del alfabeto para entrenar el modelo, manteniendo los 2000 patrones por letra.

Ciclo = 274	Precisión
Entrenamiento	99.9038 %
Test	90.2797 %

Tabla 30: Resultado de CNN con imágenes de 26 letras distintas.

Como vemos, con este modelo la CNN necesitó más tiempo para terminar el entrenamiento: 274 ciclos frente a los 82 ciclos para las 13 letras. A pesar de aumentar notablemente el número de clases y la complejidad del problema, se llega a un porcentaje en test bastante aceptable, y más teniendo en cuenta la presencia de mayúsculas y minúsculas y de muchas letras similares. Como podemos observar, es evidente que esta técnica es muy superior a las demás en problemas de clasificación de imágenes y nos da unos resultados bastante buenos incluso al llevar el problema a su máxima complejidad.

5. Conclusiones

Los resultados fueron mayoritariamente buenos para nuestro problema, con lo que podríamos concluir que el uso de sistemas de aprendizaje automático es viable para resolverlo. Para unas pocas clases, el mejor de los sistemas fue siempre la máquina de soporte vectorial (excepto en la aproximación 2, donde todavía usábamos pocas características y KNN resultó ganadora). Para el número máximo de clases, las aproximaciones con Deep Learning demostraron ser muy precisas.

Entre las mayores dificultades de este trabajo se encuentra el encontrar nuevas

características para usar en las aproximaciones convencionales del aprendizaje automático. Además, sin una librería de análisis de imagen amplia se reducen las opciones.

6. Trabajo futuro

Este problema es fácilmente aplicable en el mundo real. Se podrían convertir imágenes de documentos escritos a mano a formato digital. Este sistema también podría aplicarse a problemas similares en los que se usan otro tipo de alfabetos o sistemas de escritura. En un futuro, podríamos añadirle a la base de datos nuevas clases, como números, signos como paréntesis, comas, puntos, para que la conversión de texto escrito a mano a digital sea completa.

Otras técnicas de Aprendizaje Automático que podríamos haber comparado para su posible aplicación al problema son aquellas basadas en árboles de decisión como el algoritmo ID3, alguna técnica de aprendizaje no supervisado como el clústering por K-medias.

Un aspecto que se podría mejorar en el futuro son las medidas de evaluación usadas. Al estar comparando más de dos modelos se podría incluir en el proceso una evaluación usando el test de Kruskal-Wallis y algún método de comparación múltiple como el método de Tukey, el de Scheffé o el de Bonferroni, usando además un conjunto distinto de test para evaluar el mejor de los modelos escogidos y evitar que el error esté sesgado.

Bibliografía

[1] Perwej DY, Chaturvedi A. Neural Networks for Handwritten English Alphabet Recognition [Journal Article]. International Journal of Computer Applications (0975 – 8887). 2011;Volume 20– No.7:1–5. Available from: https://www.researchgate.net/publication/224963012_Neural_Networks_for_Handwritten_English_Alphabet_Recognition.

[2] Kothuri S, Annapurna P, Lukka S. Digit Recognition Using Freeman Chain Code [Journal Article]. International Journal of Application or Innovation in Engineering & Management 2319 - 4847. 2013;2:362–365. Available from: https://www.researchgate.net/publication/283496289_Digit_Recognition_Using_Freeman_Chain_Code.

[3] McDonnell MD, Tissera MD, Vladusich T, van Schaik A, Tapson J. Fast, Simple and Accurate Handwritten Digit Classification by Training Shallow Neural Network Classifiers with the 'Extreme Learning Machine' Algorithm [Journal Article]. PLOS ONE. 2015;10(8):e0134254. Available from: https://doi.org/10.1371/journal.pone.0134254.

[4] Lecun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition [Journal Article]. Proceedings of the IEEE. 1998;86(11):2278–2324. Available from: https://ieeexplore.ieee.org/document/726791.