Universidad Nacional de Córdoba

Facultad de Matemática, Astronomía y Física

Reconocimiento de caracteres en imágenes no estructuradas

Autor:
Rodrigo Carranza Astrada

Directores:
Dr. Jorge Sanchez
Dr. Franco Luque

Agradecimientos

A Jorge Sanchez y Franco Luque, mis directores, por su infinita paciencia, predisposición y ayuda.
A todas las autoridades de FaMAF, sin las que mis aspiraciones académicas y de investigación no hubieran podido concretarse.
A Juan Norris y Hernán Cuneo, amigos y compañeros desde el inicio de la carrera, que me ayudaron durante la duración de la misma.
A Luciana Vaggione quien amablemente me ayudo a diseñar y crear algunas de las imágenes del presente trabajo.
A mis padres y hermanos que siempre me apoyaron desde que ingrese a la carrera.

Índice

Índice de figuras

Índice de cuadros

Resumen

El reconocimiento de texto siempre ha sido un desafío a nivel computacional. Partiendo del reconocimiento de textos mecanografiados, la dificultad se acentúa enormemente cuando se intenta lograr lo mismo con textos presentes en condiciones naturales como la letra manuscrita o el texto que se puede encontrar en la etiqueta de algún producto. El que una computadora pueda discernir un carácter de otro en la imagen de un texto no es una tarea sencilla. El objetivo es clasificar caracteres en escenas naturales en donde las técnicas tradicionales de OCR no se pueden aplicar de forma directa (De Campos et al., 2009). En esta tesis se presenta un análisis del impacto producido en la performance de clasificación al entrenar un clasificador de caracteres con imágenes sintéticas (Wang et al., 2011). Se complementa esto realizando una análisis de performance utilizando diferentes conjuntos de entrenamiento sintéticos generados a partir del dataset público conocido como *Chars74k*. El resultado final de este trabajo sirve para corrobar que este tipo de datos produce un impacto positivo en la clasificación y más aún al combinar estas con datos reales.

Keywords: computer vision, Random Ferns, classification, supervised algorithm.

1. Introducción

Desde la aparición de las primeras fotografías, las personas han buscado "inmortalizar" escenas, objetos o personas, con el objetivo de, en el área de las ciencias, extraer información útil de las mismas que pueda ser utilizada para su análisis o estudio. Con el surgimiento de los primeros formatos digitales, la necesidad pasó por encontrar métodos automáticos que permitieran clasificar y reconocer elementos dentro de las imágenes. Desde reconocer texto manuscrito, texto en carteles publicitarios y patentes, hasta identificar personas, animales y zonas con agua en una imagen satelital. La cantidad de potenciales aplicaciones que se pueden obtener es enorme. Es por eso que en campos de investigación como visión por computadora, este es un tema de interés. Sin embargo, la clasificación en imágenes naturales no es una tarea para nada sencilla. Por ejemplo, en el reconocimiento de texto, las imágenes naturales contienen mucha información "extra" que se tiene que tener en cuenta. Ya sea la existencia de otros objetos ajenos a la clasificación, es decir, elementos que no son texto como así también variaciones propias en las características de la misma imagen.

Hoy en día se ha avanzado mucho en el área de la clasificación en escenas naturales. Se han desarrollado muchas aplicaciones como aquellas capaces de reconocer personas [7], patentes de vehículos [12], entre otros. Si bien, dichos avances muestran que es posible realizar lo mismo en algunos ámbitos, en otros como el reconocimiento de texto sigue siendo un desafío.

1.1. El Problema

La gran cantidad de imágenes y documentos existentes en la actualidad ha motivado el desarrollo de modelos y métodos robustos para la búsqueda automatizada de información con el objeto de reducir los problemas asociados a su análisis e interpretación. La extracción y reconocimiento de texto en imágenes naturales, es decir, imágenes de escenas de la vida diaria y/o adquiridas en condiciones no controladas, es un problema de gran interés tanto desde el punto de vista teórico como del de las aplicaciones. Sin embargo, a diferencia del problema de reconocimiento de texto en documentos escaneados, el reconocimiento de texto en imágenes naturales plantea problemas difíciles de abordar mediante el uso de técnicas tradicionales de OCR (*Optical Character Recognition*, por su denominación en inglés). Esto resulta evidente si se consideran los cambios en la iluminación, los distintos puntos de vista, las diferentes tipografías, estilos, etc. que se ven reflejadas en imágenes de la vida cotidiana tal como se puede ver en la Figura 1.

Figura 1: La parte superior muestra la imagen de un texto representada con una fuente de computadora donde se puede observar las palabras "Starbucks Coffee". La parte inferior expone las mismas palabras pero en una escena natural.

En la literatura, uno de los esquemas de procesamiento más utilizado ha sido la detección de regiones que corresponden a texto dentro la imagen, su rectificación y la posterior aplicación de técnicas estándar de OCR [6]. Sin embargo, esta clase de técnicas se encuentra limitada a los escenarios en donde el OCR funciona correctamente, por ejemplo en el reconocimiento de texto impreso [5].

Recientemente, Wang et. al. propusieron un modelo en [1], que emplea un esquema basado en técnicas conocidas de la literatura de *reconocimiento de objetos* [7]. En este caso, cada carácter alfanumérico se considera como un objeto a detectar y, empleando un conjunto de muestras de entrenamiento, se genera un modelo de clasificación mediante técnicas de aprendizaje supervisado [3]. Dada una nueva imagen, cada uno de estos clasificadores genera un conjunto de hipótesis sobre la presencia (y ubicación) de cada símbolo alfanumérico en particular. Estas detecciones se utilizan luego en la detección de palabras específicas a partir de un léxico predefinido. Una particularidad del modelo propuesto por Wang et. al. es la utilización de imágenes *sintéticas* (generadas mediante simulación) en la generación de muestras de entrenamiento. Este enfoque reduce el problema de tener que recolectar una gran cantidad de imágenes naturales, lo cual consume tiempo y esfuerzo.

1.2. Sobre el Trabajo

Esta tesis presenta una reimplementación de una parte del trabajo presentado por Wang et. al. en [1]. En esta parte, los autores buscan establecer un método para poder reconocer caracteres en imágenes naturales. Para esto proponen usar un clasificador llamado *Random Ferns*.

Este trabajo tiene como finalidad principal analizar el impacto producido por el uso de datos sintéticos en la clasificación. Esto se realiza a través de experimentos sobre diferentes conjuntos de imágenes. El primer experimento, es usando imágenes reales, de la misma manera que Wang et. al., con lo cual se busca comparar ambas implementaciones. Posteriormente, se busca analizar cómo influyen el uso de los caracteres sintéticos en el entrenamiento. Estos se alteran con el objetivo de intentar imitar a las imágenes reales y ver si se puede alcanzar o superar los resultados del primer conjunto. Por último y a diferencia de los autores originales, se propone entrenar el clasificador con un conjunto nuevo que surge de mezclar en diferentes proporciones imágenes reales y sintéticas.

1.3. Trabajos Relacionados

Se han propuesto muchos enfoques para afrontar el problema del reconocimiento de texto en imágenes naturales. De Campos et al. en [5] comparan la performance de varios clasificadores, dentro de los cuales hay un sistema de OCR comercial[1], sobre un dataset que ellos mismos crearon llamado *Chars74K*. Las conclusiones del trabajo destacan la dificultad que tienen los sistemas de OCR al momento de clasificar caracteres en imágenes naturales. Además, remarcan los beneficios de usar datos sintéticos para el entrenamiento los cuales logran un porcentaje de reconocimiento muy similar al obtenido con imágenes reales. También, realizan experimentos sobre un conjunto de imágenes de caracteres manuscritos. Sin embargo, no logran obtener buenos resultados en comparación con los otros experimentos.

Otro enfoque lo proponen B. Gatos et al. en [29]. El mismo consiste en una nueva metodología que ayuda a la detección, la segmentación y el reconocimiento automático de texto en imágenes naturales. Básicamente, la metodología consiste en lograr una eficiente binarización de las imágenes naturales. Para esto, dada una imagen natural, generan dos imágenes nuevas a partir de la original. La primera es una representación en escala de grises y la segunda es la versión invertida de la primera. Posteriormente, se les aplican diferentes técnicas para mejorarlas y así obtener la imagen binaria. Luego

[1] http://abbyy.com/finereader

utilizan una función de decisión para elegir qué imagen contiene información de texto y a dicha imagen se le realiza un post-procesamiento para eliminar el ruido existente y mejorar su calidad. Después se realiza la detección de las áreas que contienen texto para poder utilizar finalmente el motor o sistema de OCR. Uno de los problemas que se desprenden de este enfoque es que al depender de un motor de OCR, el procesamiento que se realiza a la imagen tiene que ser muy bueno ya que la mayoría de las imágenes contienen ciertos defectos como una pobre iluminación, falta de foco, entre otros.

L. Neumann y J. Matas [16] se diferencian de los enfoques tradicionales que constan en varias etapas de procesamiento y lo reemplazan con un marco de trabajo que consta de la verificación de hipótesis procesando de manera simultánea múltiples líneas de texto. Además, usan fuente de computadora como conjunto de entrenamiento.

1.4. Estructura de la Tesis

Esta tesis se desarrolla a lo largo de 5 capítulos.

En el capítulo 2 se procede a explicar los conceptos teóricos que involucran el presente trabajo. Principalmente se abordan los principios básicos del aprendizaje supervisado. Posteriormente se describen los conceptos necesarios para poder introducir el clasificador Random Ferns. Finalmente, se realiza una introducción a las nociones básicas del procesamiento de imágenes.

En el capítulo 3, se describe el trabajo realizado por Wang et al. en [1] y se explica qué partes del mismo se han implementado en la presente tesis.

En el capítulo 4, se abordan los experimentos realizados en el trabajo. Se describe tanto la implemetación del pipeline de procesamiento, como así también su diseño, el dataset usado y los resultados obtenidos. Por último se realiza un análisis completo de los resultados.

En el último capítulo, de conclusiones y trabajos futuros, se hace un resumen de lo logrado a lo largo de esta tesis así como un descripción de los objetivos futuros que se pretenden seguir en este trabajo.

2. Marco teórico

2.1. Aprendizaje automático

Desde la máquina construida por Blaise Pascal en 1642 para realizar sumas, la primera computadora programable electrónica ENIAC, hasta la actualidad, el hombre siempre ha demostrado interés por automatizar ciertos procesos. En particular, el tener una máquina que pudiera leer como una persona era un sueño, que tuvo sus inicios alrededor de 1914 cuando Edmund Fournier d'Albe desarrolló el *Optophone* (ver Figura 2), que era un escáner de mano que al moverlo a través de una página impresa, producía tonos que correspondían a letras o caracteres específicos [23].

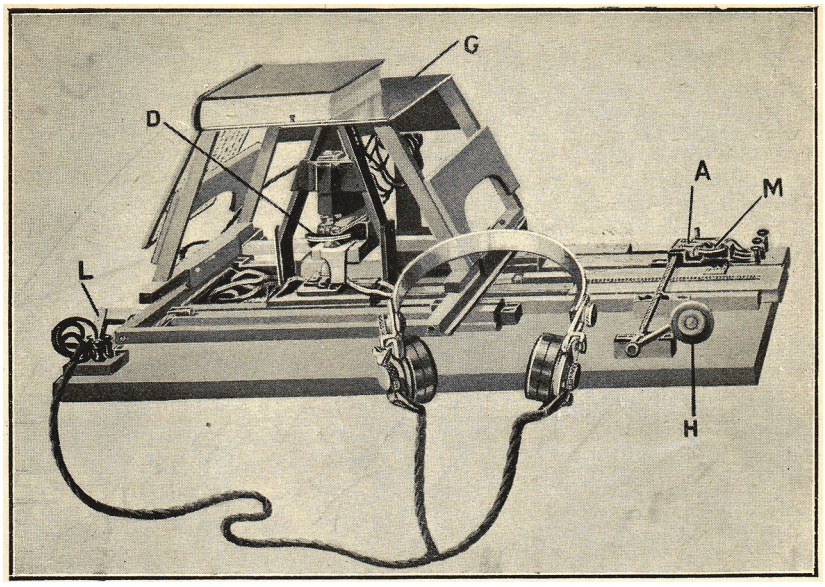


Figura 2: Imagen del *Optophone* en detalle. Extraida de la revista Vetenskapen och livet (1922).

Durante gran parte del siglo XX y hasta el día de hoy, la cantidad de información almacenada en libros y documentos escritos ha crecido de manera exponencial. Esto se debe a que cada día se vive en un mundo más globalizado, donde la información es una herramienta fundamental en cualquier ámbito. Con el surgimiento de las primeras computadoras y la digitalización de la información, se volvió más fácil compartir datos. Sin embargo, el proceso de digitalizar los documentos ya existentes era un proceso tedioso ya que se hacía manualmente. Era imperativo tener métodos automáticos que

ayudaran a clasificar y analizar toda esa información. Dada esta necesidad es que surge el campo del *aprendizaje automático* o *machine learning* que, en palabras de K. P. Murphy [22], estudia métodos que permiten detectar automáticamente patrones en los datos y luego usar esos patrones descubiertos para predecir datos futuros o poder tomar ciertas decisiones en condiciones de incertidumbre.

Uno de los temas que se tratan en este campo es el del reconocimiento de texto en imágenes. Las aplicaciones son muy numerosas, desde ayudar a las personas no videntes, [20], realizar detección de patentes de automóviles, [12], o hacer traducción automática de textos en diferentes idiomas [19].

El campo del *aprendizaje automático* tiene fuertes bases en la estadística por lo cual, los conceptos como los desarrollados en el apéndice A van a ser de utilidad para comprender las siguientes subsecciones.

De todas las técnicas existentes en este campo, las que se van a ver a continuación son la de: aprendizaje supervisado y no supervisado.

2.1.1. Aprendizaje supervisado

Como manifiesta Murphy, el aprendizaje supervisado es una técnica del aprendizaje automático cuyo objetivo es, dado un conjunto de entrada X y uno de salida Y, establecer un mapeo entre X e Y dado un conjunto etiquetado de pares de entrada-salida $M = \{(x_i, y_i)x_i \in X, y_i \in Y\}_{i=1}^{N}$ donde M es llamado el *conjunto de entrenamiento* y N es el número de ejemplos de entrenamiento [22].

El conjunto de entrenamiento es un elemento indispensable en cualquier algoritmo de aprendizaje supervisado, ya que representa la base fundamental de conocimiento necesaria para que el algoritmo pueda realizar futuras predicciones. En general, mientras más conocimiento se tenga sobre las características del objeto de interés que se esté analizando, más precisa va a ser la clasificación sobre nuevas entradas. Esto último hace referencia al concepto de *generalización*. Como expresa M. Mohri et. al. en [4], es la habilidad de interpretar con precisión nuevas entradas luego de haber "experimentado" un conjunto de entrenamiento. Este conocimiento se construye a partir de la variabilidad de los objetos conocidos, es decir el poder contar con un gran conjunto de muestras que reflejen las posibles variaciones del objeto de interés. Esto le otorga robustez a la clasificación. Por ejemplo, consideremos un clasificador de caracteres alfanuméricos que tiene como conjunto de entrenamiento un grupo de imágenes que representan a cada carácter individualmente. Luego, dado que un carácter puede estar representado de diferentes formas, es decir, pueden haber variaciones en la perspectiva del

mismo, en la iluminación del ambiente, el ruido de la imagen, entre otros, es necesario poder contar con la mayor cantidad de estas variaciones. Esto se debe a que mientras más conocimiento se tenga sobre las características del objeto de interés y las diferentes formas en que este puede aparecer, más certera va a ser la clasificación sobre nuevas entradas. Esto es lógico pues las nuevas entradas van a ser variaciones de los elementos que tenemos en el conjunto de entrenamiento.

Murphy dice que en la configuración más simple, cada entrada x_i del conjunto de entrenamiento es un vector D-dimensional de números. Estas son llamadas *características* (*features*, de su traducción al inglés). En general, sin embargo, x_i puede ser un objeto con una estructura compleja, como una imagen, un mensaje de correo, etc.

El autor destaca que, dependiendo del tipo de problema a tratar, la salida y_i puede ser una variable categórica, donde $y_i \in \{1, \ldots, C\}$ (conjunto finito de clases), o puede ser un valor real. Cuando y_i es una variable categórica, estamos frente a un problema de *clasificación* donde el objetivo es "etiquetar" o nombrar los objetos observados. Cuando y_i es una variable real, estamos en presencia de un problema de regresión, que es similar a la clasificación excepto que la respuesta es en general una variable continua.

2.1.2. Aprendizaje no supervisado

En [22], se describe al aprendizaje no supervisado como una técnica del aprendizaje automático cuyo objetivo es encontrar "estructuras interesantes" en los datos. Murphy resalta que, a diferencia del aprendizaje supervisado, no se establece qué salida se tiene que dar para cada entrada. En cambio, se busca construir modelos de la forma $p(x_i|\theta)$ donde θ es un vector de parámetros y x_i es un dato de entrada. La diferencia con el aprendizaje supervisado, es que hemos establecido $p(x_i|\theta)$ en vez de $p(x_i|y_i, \theta)$; es decir, el aprendizaje supervisado es una estimación condicional en la variable de interés, mientras que el aprendizaje no supervisado es una estimación no condicional.

Según el autor, el aprendizaje no supervisado no requiere que haya una persona que etiquete los datos manualmente. No solo es costoso, sino que además contiene relativamente poca información, sin duda no es suficiente para estimar de forma fiable los parámetros en modelos más complejos.

2.2. Clasificación

Como explica de manera sencilla Murphy K. P. en [22], el objetivo del aprendizaje supervisado es aprender un mapeo desde las entradas x a las salidas y. En clasificación, $y_i \in \{1, \ldots, C\}$ con C siendo el número de clases. Si $C = 2$, estamos frente al problema de *clasificación binaria*; mientras que si $C > 2$, la clasificación pasa a ser *multiclase*. Existe otro tipo de clasificación denominada *clasificación multi-etiqueta*, que difiere de la multiclase en cuanto a que las clases no son mutuamente excluyentes, es decir, una muestra puede pertenecer a dos o más categorías o clases. En este último caso, el mapeo se realiza desde la entrada x a un vector z, más que a una salida escalar. La elección de cual usar esta directamente asociada al tipo de problema que se quiera resolver.

Tal como expresa P. Domingos en [15], el objetivo del aprendizaje automático es *generalizar* más allá de los ejemplos en el conjunto de entrenamiento. Ya que, no importa cuantos datos tengamos, es muy poco probable que vayamos a ver los mismos ejemplos al momento de evaluar.

Para Murphy, una manera de formalizar el problema es a partir de una función de aproximación. Se asume $y = f_\theta(x)$ para cierta función desconocida f_θ, y el objetivo del aprendizaje es estimar los parámetros θ de la función f dado un conjunto de entrenamiento etiquetado. Posteriormente, se realizan predicciones usando $\hat{y} = f_{\hat{\theta}}(x)$ (usamos el símbolo $\hat{}$ para denotar estimación). El objetivo principal es realizar predicciones en entradas nuevas, es decir, que no se hayan visto durante el entrenamiento.

Como se habrá podido observar hasta acá, la clasificación es un tema de gran relevancia en el aprendizaje automático. Hacer que un programa aprenda en base a ejemplos y pueda posteriormente reconocer elementos similares es un gran desafío. Por eso es importante encontrar la mejor forma de entrenar a estos programas también denominados clasificadores.

En la próxima subsección, se va a realizar una introducción a los clasificadores probabilísticos y a la teoría de decisión. También, se presentaran varios clasificadores importantes para la comprensión del presente trabajo como Naïve Bayes, Random Forest y Random Ferns, siendo los dos primeros de relevancia para comprender al último.

2.2.1. Clasificadores Probabilísticos

Dado un conjuntos de clases Y y una entrada X, un clasificador probabilístico determina la probabilidad de que X pertenezca al conjunto de clases Y. Como expresa Bishop en [3], un clasificador probabilístico es una

distribución condicional $p(Y|X)$ sobre un conjunto finito de clases Y dada X de entrada. Una forma de determinar cuál es la mejor clase $\hat{y} \in Y$ para X sería elegir la clase con la probabilidad más alta

$$\hat{y} = \underset{y}{argmax} p(Y = y|X)$$

Uno de los clasificadores más populares es *Naïve Bayes* (explicado en las próximas secciones). Este clasificador, deriva de modelos de probabilidad generativos que proporcionan un principio para el estudio de la clasificación estadística en dominios complejos tales como el lenguaje natural y el procesamiento visual [21].

Un problema que surge al momento de clasificar una entrada a través de la probabilidad, es establecer cúal es el criterio a tomar al momento de asignar una clase a dicha entrada. Dada una entrada x, si decimos que está pertenece a una clase y con probabilidad 0,80. ¿En base a qué criterio se toma esa decisión?. De ahí surge la siguiente pregunta: ¿Existe alguna forma efectiva de decidir a que clase corresponde cada entrada?.

A continuación se introduce el concepto de la teoría de decisión. Este concepto ayuda, dado un modelo probabilístico, a establecer los mejores criterios para poder asignar una muestra a una clase.

Teoría de decisión

La teoría de decisión es el estudio de los principios necesarios para tomar decisiones correctas [35]. Nos ayuda a tomar decisiones óptimas en situaciones que involucran incertidumbre. La incertidumbre hace referencia a un estado de conocimiento limitado donde es imposible describir con exactitud el estado existente, la salida futura, o más de una posible salida.

Supongamos, como explica Bishop en [3], que tenemos un vector x como entrada junto con el correspondiente vector t de variables objetivo. La meta es predecir t dado un nuevo valor de x. En problemas de regresión, t comprenderá variables continuas, mientras que en problemas de clasificación t representará etiquetas de clase. La determinación de la probabilidad conjunta de x, t denotada como $p(x, t)$ dado un conjunto de datos de entrenamiento es un ejemplo de *inferencia* y es típicamente un problema muy difícil. La inferencia, en este caso, hace referencia a la estadística inferencial que es una parte de la estadística que comprende los métodos y procedimientos que por medio de la inducción determina propiedades de una población estadística, a partir de una pequeña parte de la misma.

Consideremos el siguiente ejemplo. Sea $C = \{C_1, \ldots, C_k\}$ un conjunto de etiquetas de clase y sea x un vector de entrada nuevo. Se desea determinar a qué clase pertenece x. El problema de inferencia involucra determinar la distribución conjunta $p(x, C_k)$, o equivalentemente $p(x, t)$.

El objetivo es decidir a cual de las k clases pertenece el vector de entrada x . Estamos interesados entonces, en las probabilidades de las k clases dado x, es decir $p(C_k|x)$, $k = 1, \ldots, K$. Usando el Teorema de Bayes, estas probabilidades pueden expresarse de la forma:

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)}$$

Luego se toma a $p(C_k)$ como la probabilidad a priori para la clase C_k, y $p(C_k|x)$ como la correspondiente probabilidad a posteriori. Por ejemplo, $p(C_1)$ representa la probabilidad de pertenecer a la clase C_1, antes de observar la muestra x.

Según Bishop, se pueden distinguir dos etapas en el problema de clasificación, la *etapa de inferencia* (entrenamiento) en el cual se usan los datos para entrenar el modelo para $p(C_k|x)$, y la subsecuente *etapa de decisión* (evaluación) en la cual se usan las probabilidades a posteriori para poder asignar las clases de manera óptima. Una alternativa es la de resolver ambos problemas en conjunto y simplemente entrenar una función que mapee las entradas x directamente con las decisiones. Dicha función es llamada *función discriminante*.

De hecho, el autor identifica tres enfoques diferentes al momento de resolver problemas de decisión. En orden decreciente de complejidad, estos son:

1. Primero, resolver el problema para determinar las densidades condicionales $p(x|C_k)$ para cada clase C_k individualmente. También de forma separada, inferir las probabilidades de clases a priori $p(C_k)$. Después, usar el teorema de Bayes en la forma:

   $$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)}$$

   para encontrar las probabilidades a posteriori $p(C_k|x)$. Como es usual, el denominador en el teorema de Bayes puede ser encontrado en término de las cantidades que aparecen en el numerador, como:

   $$p(x) = \sum_k p(x|C_k)p(C_k)$$

Equivalentemente, se puede modelar la distribución conjunta $p(x, C_k)$ directamente y después normalizar para obtener las probabilidades a priori. Habiendo encontrado las probabilidades a posteriori, se puede usar la teoría de decisión para determinar la pertenencia a una clase para cada entrada nueva x. Los enfoques que explícitamente o implícitamente modelan la distribución de las entradas, así también como las salidas, son conocidos como *modelos generativos*, debido a que tomando muestras de ellos es posible generar puntos de datos sintéticos en el espacio de entrada.

2. Inicialmente, resolver el problema de inferencia para determinar las probabilidades de clase a posteriori $p(C_k|x)$, y luego, subsecuentemente, usar la teoría de decisión para asignar a cada x nueva una de estas clases. Los enfoques que modelan las probabilidades a posteriori directas son llamados *modelos discriminativos*.

3. Encontrar una función $f(x)$, llamada función discriminante, que mapea directamente cada entrada x con una etiqueta de clase. Por ejemplo, en el caso del problema de dos clases, $f(\cdot)$ puede ser valuada de manera binaria, de manera que $f = 0$ represente a la clase C_1 y $f = 1$ represente a la clase C_2. En este caso, las probabilidades no toman partido.

En [3] se consideran los méritos relativos a estas tres alternativas. El enfoque (1) es el más demandante debido a que involucra encontrar la distribución conjunta tanto de x como de C_k. Para muchas aplicaciones, x tendrá alta dimensionalidad, y por consiguiente puede ser necesario un conjunto de entrenamiento grande con el fin de ser capaz de determinar las densidades de clase condicional con una exactitud razonable. Hay que tener en cuenta que la probabilidades a priori $p(C_k)$ de la clase a menudo pueden estimarse simplemente a partir de las proporciones de los punto de datos del conjunto de entrenamiento en cada una de las clases.

Sin embargo, si sólo deseamos realizar decisiones de clasificación, esto conlleva a un gasto de recursos computacionales y una demanda de datos excesiva para encontrar la distribución conjunta $p(x, C_k)$ cuando de hecho, solamente se necesitan las probabilidades a posteriori $p(C_k|x)$, las cuales pueden ser obtenidas a través del enfoque (2).

Un enfoque mucho más simple es (3) en el cual se usa un conjunto de entrenamiento para encontrar una función discriminante $f(x)$ que mapea cada x directamente a una etiqueta de clase, así combinando la inferencia y las etapas de decisión en un simple problema de aprendizaje.

Con la opción (3), sin embargo, se pierde el acceso a las probabilidades a porteriori $p(C_k|x)$.

2.2.2. Vectores de características

Hasta el momento, se ha introducido la idea general de qué es un clasificador probabilístico. Se detalló que estos clasificadores toman una "entrada X" y le asignan una probabildiad de que dicha entrada pertenezca a cada una de las clases asociadas. El proceso interno para realizar esto varia entre clasificadores, sin embargo, un concepto necesario para comprender cómo trabajan cuando "procesan" una entrada es el de *característica*.

Una *característica* o *feature*, es un aspecto o cualidad distintiva de un objeto (clase de interés). Las características son importantes dado que al representar los aspectos o cualidades más significativas de un objeto, facilitan el reconocimiento posterior de objetos similares [26]. Esto es fundamental por ejemplo, en los esquemas de clasificación en el aprendizaje supervisado ya que permite, dada una muestra desconocida, identificar a qué clase pertenece si anteriormente sabemos las características particulares de cada clase (a partir de instancias o muestras analizadas con anterioridad). Esto conlleva a que si se realiza una buena selección de características, impacte positivamente en la precisión de los clasificadores y por ende en el reconocimiento. Por ejemplo, en los algoritmos de detección de spam, las características pueden incluir el lenguaje en el que está escrito el email, la ausencia o presencia de ciertos encabezados, la corrección gramatical del texto, entre otros [10].

Un *vector de características* o "feature vector" se lo puede definir como un conjunto de características que definen a un objeto. Dicho objeto, en adelante X, es representado por un vector D-dimensional tal que $X = (x_1, \ldots, x_D)$ donde los x_i representan a las características del objeto. En general, $x_i, i = 1, \ldots, D$ son numéricas dado que dicha representación facilita el análisis estadístico y el procesamiento de datos.

En base a lo explicado anteriormente, se presenta a continuación al primer clasificador probabilístico que va a servir de ayuda para comprender los siguientes clasificadores.

2.2.3. Clasificador Naïve Bayes

En [27], Narasimha y Susheela definen a Naïve Bayes como un clasificador probabilístico basado en la aplicación del Teorema de Bayes con una fuerte suposición de independencia, naïve. En términos simples, los autores

detallan que un clasificador Naïve Bayes asume que la presencia o ausencia de una característica particular no está relacionada con la presencia o ausencia de cualquier otra característica. Este tipo de clasificador considera que cada una de estas características contribuye independientemente a la probabilidad de que un elemento sea de una clase particular, independientemente de la presencia o ausencia de otras características. Por ejemplo, una fruta puede ser considerada una manzana si es roja, redonda y de 7cm de diámetro aproximadamente.

El modelo general para un clasificador es:

$$p(C|F_1, \ldots, F_n)$$

sobre una variable dependiente C, con un pequeño número de resultados, o clases. Esta variable está condicionada por varias variables dependientes, *features*, desde F_1 a F_n [27]. El problema es que si el número n de variables dependientes es grande, o cuando éstas pueden tomar muchos valores, entonces basar este modelo en tablas de probabilidad se vuelve computacionalmente imposible. Por ejemplo, si hubiesen 35 variables independientes con 2 valores posibles cada una entonces habrían 34,359,738,368 valores posibles distribuidos en múltiples tablas. Si se usaran decimales de punto flotante simple (4 bytes), se necesitarían 32 gigabytes de memoria para almacenar todo, lo que lo hace muy poco manipulable. Por lo tanto, Narasimha et. al en [27], reformulan el modelo para hacerlo más manejable: Usando el teorema de Bayes se escribe:

$$p(C|F_1, \ldots, F_n) = \frac{p(C)\ p(F_1, \ldots, F_n|C)}{p(F_1, \ldots, F_n)}.$$

que puede ser reescrita como sigue, aplicando repetidamente la definición de probabilidad condicional:

$$\begin{aligned} p(C, F_1, \ldots, F_n) &= p(C)\ p(F_1, \ldots, F_n|C) && (2.2.1)\\ &= p(C)\ p(F_1|C)\ p(F_2, \ldots, F_n|C, F_1) && (2.2.2)\\ &= p(C)\ p(F_1|C)\ p(F_2|C, F_1)\ p(F_3, \ldots, F_n|C, F_1, F_2) && (2.2.3) \end{aligned}$$

y así sucesivamente. Ahora es cuando la asunción "naïve"de independencia condicional entra en juego: se asume que cada F_i es independiente de cualquier otra F_j para $j \neq i$. Esto significa que

$$p(F_i|C, F_j) = p(F_i|C)$$

por lo que la probabilidad compuesta puede expresarse como

$$p(C, F_1, \ldots, F_n) = p(C)\ p(F_1|C)\ p(F_2|C)\ p(F_3|C) \ldots$$
$$= p(C) \prod_{i=1}^{n} p(F_i|C).$$

Esto significa que haciendo estas presunciones, la distribución condicional sobre C puede expresarse de la siguiente manera:

$$p(C|F_1, \ldots, F_n) = \frac{1}{Z} p(C) \prod_{i=1}^{n} p(F_i|C)$$

donde Z es un factor que depende sólo de $F_1, \ldots, F_n$, es decir, constante si los valores de F_i son conocidos [27].

La siguiente subsección, si bien no está relacionada directamente con Naïve Bayes, va a ser de ayuda para entender al clasificador *Random Forest*. Lo que se ha visto hasta ahora sobre Naïve Bayes junto con lo se va a ver en la sección 2.2.5 sobre Random Forest va a servir para comprender al clasificador Random Fern.

2.2.4. Árboles de decisión

Un árbol de decisión es una estructura jerárquica compuesta por nodos internos que representan evaluaciones sobre ciertos atributos, características del problema que se busca resolver, y nodos hoja que representan las decisiones finales en base a las evaluaciones de los nodos. La rama que conecta a un par de nodos refleja uno de los posible valores para el atributo del nodo padre.

Los árboles, en el área de la computación, son estructuras de datos capaces de almacenar y representar información de forma jerárquica (a través de sus nodos). En general y en especial en el área de aprendizaje supervisado son de mucha utilidad ya que computacionalmente son eficientes a la hora de buscar información dentro de ellos (a diferencia de otras estructuras de datos). Además, su representación es fácil de interpretar y su implementación es sencilla.

Consideremos la Figura 3 que contempla el siguiente ejemplo: ¿Es conveniente ir a jugar al tenis basado en las condiciones climáticas?. La figura es una representación de un árbol de decisión que tiene un estructura similar a un diagrama de flujo. Como se puede observar, en cada nodo del árbol se evalúa un atributo diferente. En este caso, cada atributo hace referencia a

una condición climática. En base a las respuestas de los nodos, se avanza en el árbol hasta llegar a una hoja donde se toma una decisión. En el ejemplo de la figura, si el clima está despejado y la humedad es normal entonces es conveniente ir a jugar al tenis. Los árboles de decisión pueden ser usados para la clasificación, variables discretas, o regresión, variables continuas. Cuando son usados para la clasificación, cada nodo representa una característica, cada rama que sale de un nodo representa un valor posible para la característica que representa al mismo y por último las hojas representan la etiqueta de clase (decisión tomada luego de computar todos los nodos). La clasificación comienza en la raíz, donde se pregunta sobre algún valor de una característica en particular del objeto a analizar. Las diferentes ramas que salen de la raíz corresponden a las diferentes valores posibles. Basado en la respuesta, se continúa por la rama hasta el nodo siguiente. La siguiente etapa, es realizar una decisión en el nodo en cuestión que puede ser considerado como la raíz del sub-árbol y se continua de esta manera hasta que se alcanza una hoja, la cual no contiene más preguntas. Cada hoja contiene una etiqueta categórica y al objeto, se le asigna la etiqueta de la hoja que ha alcanzado.

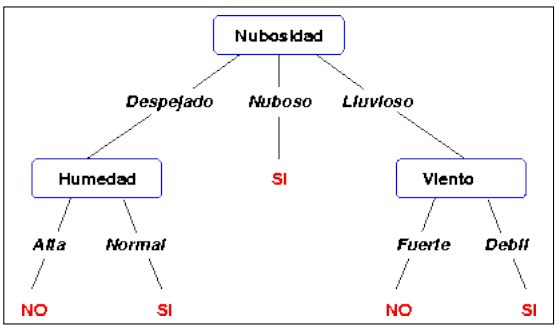


Figura 3: Árbol de decisión. Imagen extraida de Mitchell, T.M. (1997) *Machine Learning*, McGraw-Hill.

Los algoritmos para entrenar un árbol de decisión usualmente trabajan de *arriba hacia abajo*(*top-down*, por su denominación en inglés) [25]. Dado un conjunto de características iniciales, en cada nodo del árbol, los algoritmos utilizan una función que evalúa cúal es el atributo más apropiado para representar al nodo y en base a eso dividen el conjunto inicial en dos o más partes. La elección de que función utilizar para realizar estas decisiones no es sencilla. En algunos casos, lo que se busca es generar un árbol de decisión óptimo que reduzca el error de generalización, en otros, se busca reducir

el número de nodos o la profundida del árbol [24]. Este proceso continúa de manera recursiva hasta que el subconjunto de características en un nodo tiene el mismo valor para la variable objetivo, sea una clase o un valor numérico, o cuando las divisiones no agregan ningún valor a las predicciones. Este proceso es una estrategia muy común al momento de entrenar un árbol de decisión a partir de datos y es un ejemplo de *algoritmo ambicioso*(*greedy algorithm*, por su traducción al inglés). Si se considera nuevamente el ejemplo de la Figura 3, se puede ver que los atributos, si bien son pocos, están bien ubicados en los nodos del árbol. En cambio, si se intercambian los nodos con los atributos *humedad* y *viento*, entonces el preguntar por la humedad sabiendo que el día está lluvioso no aporta información al momento de tomar una decisión.

Cada algoritmo tiene su propio método al momento de dividir los nodos y decidir cuál es la mejor división que conlleve a disminuir los errores de clasificación. Por ejemplo, los algoritmos *ID3 y C4.5*, [31, 32], el primero precursor del segundo, utilizan el concepto de *entropía* o *entropía de información* para elegir al atributo que va a dividir el conjunto de características y que obviamente va a representar al nodo en el árbol. La entropía se la define como la medida de incertidumbre de una variable aleatoria. Básicamente, dado un conjunto de características S, se itera sobre cada atributo no usado del mismo y se calcula la entropía sobre el mismo de la siguiente manera:

$$\eta(S) = -\sum_{i} P(s_i) log_2 P(s_i) \tag{2.2.4}$$

donde S es nuestro conjunto conformado con los posibles valores $\{s_1, \ldots, s_n\}$ y $P(\cdot)$ es la función de probabilidad. Posteriormente, se elige aquel atributo que tenga menor entropía, mayor ganancia de información, y se lo utiliza para partir el conjunto S. Construir un árbol de decisión se basa en encontrar atributos que retornen la mayor ganancia de información con lo cual se obtienen ramas más homogéneas.

Los árboles de decisión crecen hasta que se cumplen ciertos criterios: que se haya alcanzado la máxima profundidad del árbol, que todas las instancias del conjunto de entrenamiento pertenezcan a un solo valor de salida, entre otros [24].

Como expresan O. Maimon y L. Rokach en [24], se puede dar el caso de que el árbol resultante sea excesivamente complejo, por ejemplo, al tener demasiados parámetros relativos al número de observaciones. Esto es lo que se conoce como *sobreajuste* (*overfitting*, de su traducción al inglés). Un modelo que ha sido sobreajustado, tendrá generalmente una baja tasa de predicción.

Este tipo de problemas se generan cuando se eligen pobremente los criterios de parada. El caso contrario generaría árboles pequeños o débilmente ajustados. Los autores destacan que una forma de solucionar el sobreajuste es utilizando la técnica de *poda* (*pruning*, de su traducción al inglés). La poda reduce el tamaño de los árboles de decisión eliminando secciones del mismo que provean poca capacidad para clasificar instancias. El objetivo es tanto la reducción de la complejidad del clasificador como así también mejorar la performance a través de la reducción del sobreajuste.

2.2.5. Random Forest

Un *Random Forest* o *bosque aleatorio* es un método de aprendizaje conjunto o *ensemble learning* para la clasificación o regresión. El aprendizaje conjunto, es una técnica que consiste en combinar varios predictores con el objetivo de obtener uno más "fuerte" que pueda realizar mejores predicciones. En el caso de clasificación, consiste en una colección de clasificadores con estructura de árbol, $\{h(x, \Theta_k), k = 1, \dots\}$, donde $\{\Theta_k\}$ son vectores aleatorios independientes e idénticamente distribuidos, Θ_k representa los parámetros para la construcción del k-esimo árbol, y $h(x, \Theta_k)$ es un clasificador donde x es un vector de entrada. Luego, dada una entrada x, cada árbol emite un único voto para la elección de la clase más popular para x [11].

Random Forest es un árbol aleatorio. El mismo, es construido a través de un proceso no determinístico. A diferencia de los algoritmos *ID3* y *C4.5*, explicados en la sección anterior, donde la elección del valor del nodo se da a través de la función de entropía, en los árboles aleatorios cada nodo del árbol se construye a partir de un proceso aleatorio que le asigna su valor.

El algoritmo de entrenamiento para Random Forest aplica la técnica general de *bootstrap aggregating*(agregación bootstrap) o *bagging*(embolsado). Esta técnica fue desarrollada por Breiman en 1996, [18], y es un método para generar múltiples versiones de un predictor y usar estos para obtener un predictor agregado. Para tener una idea más clara del concepto, consideremos el siguiente ejemplo. Supongamos que tenemos un conjunto de entrenamiento $L = \{(x_n, y_n), n = 1, \dots, N\}$, donde x_n son valores de entrada e y_n son etiquetas de clases, (o valores numéricos en el caso de regresión). Supongamos también que tenemos una forma de generar un predictor de la forma $\varphi(x, L)$ a partir de L tal que $\varphi(x, L) = y$. Por último, supongamos que nos dan una secuencia de conjuntos de aprendizaje, $\{L_k\}$, donde cada uno consiste en N observaciones independiente bajo la misma distribución de L. El objetivo es que usando $\{L_k\}$ se obtenga un predictor mejor que el

establecido anteriormente $\varphi(x, L)$ que usa solamente un conjunto de entrenamiento. La única restricción, es que se nos obliga a trabajar solamente con el conjunto de predictores $\{\varphi(x, L_k)\}$.

Para resolver este problema, Breiman estableció el siguiente criterio. Si la respuesta y era un valor numérico luego, se reemplaza a $\varphi(x, L)$ por el promedio del conjunto de predictores $\{\varphi(x, L_k)\}$ sobre k. Es decir, $\varphi_A(x) = \mathbb{E}_L\varphi(x, L)$, donde $\mathbb{E}_L$ denota la esperanza sobre L, y el subíndice A en φ_A denota la agregación. En cambio, si $\varphi(x, L)$ predecía una etiqueta de clase $j \in \{1, \ldots, J\}$, luego un método para agregar todos los predictores era a través del voto. Es decir, $N_j = \{k; \varphi(x, L_k) = j\}$ y se toma a $\varphi_A(x) = \underset{j}{argmax}|N_j|$.

El problema principal, es que generalmente se tiene sólo un conjunto de aprendizaje L. Para esto, Breiman considera que se puede imitar el procedimiento anterior tomando repetidas muestras bootstrap, $\{L^B\}$, a partir de L, y formar $\{\varphi(x, L^B)\}$. Formalmente dado L, una muestra boostrap $\{L^B\}$ se genera tomando b muestras de L uniformemente con reemplazo. Dado que hay reemplazo, se puede dar el caso de que se repitan elementos en $\{L^B\}$. Si y es numérica luego, toma φ_B como

$$\varphi_B(x) = av_B\varphi(x, L^B).$$

donde av denota la media sobre el conjunto de predictores. Si y es una etiqueta de clase, luego el conjunto $\{\varphi(x, L^B)\}$ vota para formar $\varphi_B(x)$. El autor a este procedimiento lo llama *bootstrap aggregating* o *bagging*.

Cabe aclarar, que cada $L_i \in \{L^B\}$ consta de N muestras obtenidas al azar, pero con reemplazo, de L. Cada (x_n, y_n) puede aparecer repetido una cierta cantidad de veces o no en L_i.

Se puede aplicar *bagging* para generar un algoritmo para árboles de decisión o regresión. Dado un conjunto de aprendizaje L como el explicado anteriormente, la técnica de bagging selecciona repetidamente muestras de bootstrap del conjunto de aprendizaje L y ajusta los árboles a estas muestras:

Para $b = 1$ hasta B:

- Se realiza un muestreo, con reemplazo, de n ejemplos de entrenamiento a partir de L; llamemos a esta muestra L_b (muestra bootstrap).
- Entrena un árbol de decisión o regresión f_b a partir de L_b.

Después del entrenamiento, las predicciones para ejemplos no vistos x' se pueden realizar a través del voto de todos los predicciones de la siguiente

manera:

$$N_j = \{k; \varphi(x', L_k) = j\}$$

y se toma a

$$\varphi_A(x') = \underset{j}{argmax}|N_j|$$

o el promedio en caso de árboles de regresión: $\bar{f} = \frac{1}{B}\sum_{b=1}^{B} \bar{f}_b(x')$.

En el algoritmo de arriba, B es un parámetro libre que indica la cantidad de árboles predictores que se van a emplear. Típicamente, algunos cientos o miles de árboles son usados, dependiendo del tamaño y naturaleza del conjunto de entrenamiento.

El procedimiento anterior describe el algoritmo original de *bagging* para árboles. Desafortunadamente, volver a correr el mismo algoritmo de aprendizaje en diferentes subconjuntos de los datos puede resultar en predictores altamente correlacionados, lo cual limita la reducción de varianza. La técnica conocida como *random forest*, construye árboles basados en un subconjunto de variables de entrada elegidas al azar.

Cada árbol es construido siguiendo el siguiente algoritmo:

- Si el número de muestras en el conjunto de entrenamiento es P, muestrear N casos aleatoriamente - pero con reemplazo, a partir de los datos originales. Esta muestra va a ser el conjunto de entrenamiento para la construcción del árbol.
- Si hay M variables de entrada, se especifica un número $m << M$, constante durante el crecimiento del bosque o forest, tal que en cada nodo se seleccionen m variables al azar de las M. Posteriormente se eligen entre las m variables aquellas que mejor dividan al nodo, es decir, aquellas que generen al final un árbol compacto y simple.
- Cada árbol se construye hasta su máxima extensión posible. No hay pruning(poda).

Las ventajas de Random forests son:

- Correr eficientemente en grandes bases de datos.
- Poder manejar cientos de variables entrantes sin excluir ninguna.
- Dar estimaciones de qué variables son importantes en la clasificación.
- Ofrecer un método experimental para detectar las interacciones de las variables.

Las desventajas de este algoritmo se pueden resumir en estos puntos:

- A diferencia de los árboles de decisión, la clasificación hecha por random forests es difícil de interpretar por el hombre.
- Si los datos contienen grupos de atributos correlacionados de relevancia similar para el rendimiento, entonces los grupos más pequeños están favorecidos por sobre los grupos más grandes.

2.2.6. Random Ferns

Random Ferns es un clasificador propuesto por Ozuysal et. al. [28]. Al igual que Random Forest, es un clasificador ensemble, compuesto de un determinado número de entidades o clasificadores y es una alternativa más rápida y simple que este último. En oposición a Random Forest, Ferns es una estructura no jerárquica donde cada entidad que constituye el clasificador es básicamente un conjunto de prueba binario. En Random Forest el conjunto de prueba de cada árbol es la colección de diferentes pruebas que se distribuyen a lo largo de los nodos que forman el árbol. Debido a la estructura plana de cada entidad en un clasificador Ferns, el conjunto de prueba es una sencilla lista ordenada de las posiciones de los features o características a ser evaluados.

Sea $c_i, i = 1, \ldots, H$, el conjunto de clases y sea $f_j, j = 1, \ldots, N$, el conjunto de características binarias. Formalmente, se busca:

$$\underset{c_i}{argmax}\ P(C = c_i | f_1, f_2, \ldots, f_N),$$

donde C es una variable aleatoria que representa a la clase. La fórmula de Bayes produce:

$$P(C = c_i | f_1, f_2, \ldots, f_N) = \frac{P(f_1, f_2, \ldots, f_N | C = c_i) P(C = c_i)}{P(f_1, f_2, \ldots, f_N)}$$

Asumiendo una probabilidad uniforme a priori $P(C)$, dado que el denominador es simplemente un factor de escala que es independiente de la clase, el problema se reduce a encontrar:

$$c_i = \underset{c_i}{argmax}\ P(f_1, f_2, \ldots, f_N | C = c_i) \tag{2.2.5}$$

Una representación completa de la probabilidad conjunta mediante tablas en la ecuación 2.2.5 no es factible dado que requeriría estimar y almacenar 2^N entradas por cada clase. Una forma de comprimir la representación es

asumir independencia entre características como Naïve Bayes. Una versión extrema es la de asumir independencia completa, es decir

$$P(f_1, f_2, \ldots, f_N | C = c_i) = \prod_{j=1}^{N} P(f_j | C = c_i)$$

Sin embargo, esto ignora completamente la correlación entre las características. Para hacer el problema manejable, manteniendo alguna correlación, un buen método es partir las características en M grupos de tamaño $S = \lfloor \frac{N}{M} \rfloor$. Estos grupos son los que se denominan *ferns* y se calcula la probabilidad conjunta de cada característica en cada fern. La probabilidad condicional se vuelve

$$P(f_1, f_2, \ldots, f_N | C = c_i) = \prod_{k=1}^{M} P(F_k | C = c_i) \tag{2.2.6}$$

donde $F_k = \{f_{\sigma(k,1)}, f_{\sigma(k,2)}, \ldots, f_{\sigma(k,S)}, k = 1, \ldots, M\}$ representa el k^{th} fern y $\sigma(k, j)$ es una función de permutación con rango $1, \ldots, N$. De ahí que se sigue un enfoque bayesiano Semi-Naïve modelando algunas de las dependencias entre características.

La fase de entrenamiento estima la probabilidad condicional de clase $P(F_m | C = c_i)$ para cada fern F_m y cada clase c_i, como está descrito en la ecuación 2.2.6. Para cada fern F_m escribimos estos términos como:

$$p_{k,c_i} = P(F_m = k | C = c_i) \tag{2.2.7}$$

donde se simplifica la notación considerando que F_m es igual a k si el número en base 2 formado por los features binarios de F_m tomados en secuencia es igual a k. Con esta convención, cada "fern" puede tomar $K = 2^S$ valores, y para cada uno, es necesario estimar $p_{k,c_i}, k = 1, 2, \ldots, K$ bajo la restricción

$$\sum_{k=1}^{K} p_{k,c_i} = 1$$

El enfoque más simple sería asignar la estimación de máxima verosimilitud a estos parámetros a partir de las muestras de entrenamiento. Para el parámetro p_{k,c_i} es

$$p_{k,c_i} = \frac{N_{k,c_i}}{N_{c_i}}$$

donde N_{k,c_i} es el número de muestras de entrenamiento de la clase c_i y N_{c_i} es el número total de muestras para la clase c_i. Estos parámetros por lo tanto, pueden ser estimados para cada fern independientemente.

En la práctica sin embargo, este simple esquema da pobres resultados porque si ninguna muestra de entrenamiento para la clase c_i evalúa a k, lo cual puede pasar con facilidad cuando el número de muestras no es infinitamente grande, ambos N_{k,c_i} y p_{k,c_i} serán cero. Dado que se multiplica p_{k,c_j} por todos los ferns, esto implica que, si el fern evalúa a k, la correspondiente muestra nunca va a ser asociada a la clase c_i, sin importar la respuestas de los otros ferns. Para superar este problema, se considera p_{k,c_i} de la siguiente manera

$$p_{k,c_i} = \frac{N_{k,c_i} + N_r}{N_{c_i} + K \times N_r}, \tag{2.2.8}$$

donde N_r representa un término de regularización. La ecuación 2.2.8 usa la técnica de *suavizado de Laplace* (*Laplace smoothing*, por su traducción al inglés) que es usada en estadística para suavizar datos categóricos. Si una muestra con un valor específico de fern no se encuentra durante el entrenamiento, este esquema, aún le va a asignar un valor distinto de cero a la probabilidad correspondiente.

3. Reconocimiento de texto en escenas naturales

En el capítulo anterior se desarrollaron los conceptos necesarios para entender las bases del aprendizaje supervisado y los clasificadores probabilísticos. Todos estos conceptos sirven para entender los principios sobre los que se asienta el trabajo realizado por Wang et. al. en [1], que abarca el reconocimiento de texto en escenas naturales en todas sus etapas. En particular, una de ellas es el reconocimiento de caracteres.

En este capítulo se van a presentar dos implementaciones: la provista originalmente por Wang et al. en [1] y la realizada en este trabajo basándose en la primera. Solamente se van a desarrollar los temas referentes al reconocimiento de caracteres ya que el presente trabajo se enfoca únicamente en ese problema.

3.1. Introducción

En el trabajo que realizaron Wang et al., los autores identificaron el problema que había al detectar y reconocer palabras en imágenes naturales. Si bien las actuales aplicaciones de OCR se manejan bien con documentos escaneados, todavía encuentran problemas cuando tratan de procesar texto adquirido en entornos naturales, también referido como texto de escena. Este tipo de texto se ha vuelto más frecuente debido al aumento de dispositivos que son capaces de extraer dicha información, sean estos celulares, tabletas o cámaras.

Durante la competencia de ICDAR (*International Conference on Document Analysis and Recognition*) en 2003, los organizadores tenían como objetivo ver cuál era el estado del arte en las diferentes etapas del reconocimiento de texto en escenas naturales. Observaron que habían imágenes con texto que los motores de OCR del momento no podían procesar y por lo tanto, decidieron dividir el problema general de reconocer palabras en escenas naturales en tres subproblemas:

- **La clasificación de caracteres recortados.**

  En este problema se consideran imágenes de caracteres extraídas de escenas naturales. Las imágenes contienen exclusivamente un sólo carácter y sus tamaños se ajustan a las dimensiones del caracter, tal como se puede apreciar en la Figura 4. El objetivo es identificar qué carácter está reflejado en la imagen a partir de un conjunto predefinido de caracteres que hacen al problema.

Una de las dificultades de este problema es que algunos caracteres son muy difíciles de dicernir entre sí, por ejemplo, si consideramos las letras mayúsculas y las minúsculas por separado, una "Z" es muy difícil de dicernir de una "z". Incluso se puede dar lugar a la confusión entre distintos símbolos, por ejemplo, la "O", letra o mayúscula, y el número "0" o el número "6" y la "G", letra "g" mayúscula.

Otro elemento que se tiene que tener presente al momento de clasificar caracteres, es el tipo de características locales que se van a obtener de cada imagen. Como se ha explicado en la sección 2.2.2, las características son importantes dado que representan los aspectos o cualidades más significativas de un objeto. Si se hace una buena elección de las características locales, se va a ver reflejado positivamente en la performance de clasificación.

Las condiciones en que fueron tomadas las imágenes donde se extrajeron los caracteres influye posteriormente en su reconocimiento. Por ejemplo, las variaciones en las condiciones de iluminación, oclusión, posicionamiento al momento de tomar la imagen, etc. Para resolver esto, se puede realizar un pre-procesamiento que ayude a "limpiar" la imagen para poder facilitar su reconocimiento posterior.

- **Detección de zonas con texto en la imagen.**

  Este problema involucra la detección de zonas de la imagen que pueden contener texto. Esto se realiza con el objetivo de priorizar estas regiones en procesamientos posteriores para reducir la complejidad del análisis de texto. La Figura 5 expone un ejemplo de esto.

  Para poder resolver esta dificultad, se debe considerar que las palabras que conforman el texto a detectar pueden haber sido adquiridas a distintas escalas. Tal es el caso del texto de casi la mayoría de los carteles publicitarios que es posible encontrar en las calles de una ciudad. Otro factor a considerar es la inclinación del texto que puede darse por la posición del observador. Además, esta tarea se dificulta si consideramos, al igual que en el reconocimiento de caracteres, las condiciones de la imagen: iluminación, distorsiones, estilo y fuente de las palabras en el texto, etc.

  En el trabajo de Chen H. et. al. [30], los autores destacan que hay dos categorías al momento de diferenciar las técnicas de reconocimiento de texto. La primera categoría, son las técnicas *basadas en textura* (*texture based* de su traducción al inglés) que destacan al texto como una "textura" especial que es distinguible del fondo. Las características

Figura 4: Conjunto de caracteres recortados. Imagen extraída de T. E. de Campos et. al. [5]

se extraen de ciertas regiones de la imagen y se utiliza un clasificador para identificar la existencia de texto. La segunda categoría, son las técnicas basadas en *componentes conectados* (*connected component* de su traducción al inglés), donde se extraen regiones de la imagen y se utilizan restricciones geométricas para descartar candidatos que no sean texto.

- **El reconocimiento de palabras recortadas.**

  En este problema se busca identificar las palabras que se encuentran dentro de las imágenes recortadas. Al igual que el problema del reconocimiento de caracteres, en este problema, cada imagen contiene una sola palabra y la dimensiones de estas imágenes se ajustan a las palabras tal como se puede ver en la Figura 6.

  Una de las dificultades que surgen en este problema al manipular imágenes naturales, son las condiciones en que estas fueron adquiridas. Como se explicó anteriormente, las variaciones en la iluminación, la oclusión, entre otros, generan problemas en el reconocimiento.

  Uno de los métodos que se utiliza en la actualidad y se considera

Figura 5: Imagen natural donde las zonas con texto están encasilladas. Fuente: `http://libccv.org/post/introducing-ccv-milestone/`

estado del arte son las estructuras pictóricas. Este método fue usado por Wang y Belongie en [2], donde en base a un lexicón, conjunto de palabras, miden la configuración de cada carácter de cada palabra en la imagen. Básicamente toma la ubicación y el puntaje de los caracteres detectados como entrada y encuentra la configuración óptima para una palabra en particular [1]. Una de las particularidades de este trabajo, es que requiere de un clasificador de caracteres para poder posteriormente realizar el reconocimiento de palabras.

Figura 6: Conjunto de palabras recortadas de diferentes escenas naturales. Imagen extraida del sitio de *Graphics and Media Lab*, `http://graphics.cs.msu.ru/en/science/research/machinelearning/text`

Dada esta problemática, Wang et al. se enfocaron en un caso especial

del reconocimiento de texto de escena donde tenían a disposición un listado de palabras, i.e, un lexicón, para detectar y leer.

Para esto, ellos construyen y evalúan dos sistemas. En el primero, evalúan la performance en la detección y el reconocimiento de palabras de un enfoque con dos etapas que consiste en un detector de texto considerado estado del arte y un destacado motor de OCR. El segundo es un sistema arraigado en el reconocimiento de objetos genéricos, el cual es una extensión de un trabajo que realizaron anteriormente, [2]. En [2], los autores consideran a las palabras como categorías de objetos en sí mismas y realizan reconocimiento sobre las categorías de las palabras. Utilizan técnicas propias del reconocimiento de categorías genéricas y las aplican al reconocimiento de palabras. Por ejemplo, así como podemos tener muchas imágenes que representen el objeto "vehículo", también es posible aplicar la misma idea para la palabra "door" como muestra la Figura 7. En la figura se muestra una analogía con el reconocimiento de objetos genéricos.

En [1], los autores remarcan que para poder lograr el reconocimiento de palabras en imágenes, es necesario en primera instancia tener un clasificador de caracteres. Este trabajo se enfoca en el reconocimiento de caracteres en ventanas, es decir, imágenes de caracteres recortados de escenas naturales.

Figura 7: Reconocimiento de palabras. Se considera a la palabra como una categoría de objeto al igual que la categoría "vehiculo" presentada en la parte superior de la imagen.

3.2. Reconocimiento de caracteres como un problema de clasificación de imágenes naturales

El reconocimiento de caracteres es la primera etapa en el pipeline de procesamiento que desarrollaron Wang et al. Dada una imagen, inicialmente es necesario detectar las potenciales ubicaciones de los caracteres dentro

de la misma. Para lograr esto, los autores realizan una detección a múltiple escala usando un algoritmo de clasificación de ventana deslizante. Esta ventana al comienzo tiene un tamaño fijo, y recorre la imagen detectando potenciales caracteres contenidos en ella. Luego aumenta su tamaño con el objetivo de poder detectar caracteres más grandes. Supongamos que la ventana está ubicada en una zona de la imagen donde hay un carácter a detectar. Si quisieramos reconocer ese carácter, sería necesario compararlo con los 62 posibles caracteres existentes. Dado que hay muchos símbolos involucrados, los autores tienen que lograr clasificar cada nueva entrada dentro de las 62 posibles clases. Estas clases estan formadas por los caracteres alfabéticos, en minúscula y mayúscula, 52 en total, y numéricos, 10 en total). Dada la gran cantidad de clases, y que los sistemas que emplean esquemas de reconocimiento de caracteres usualmente deben funcionar en tiempo real y evaluar una gran cantidad de ventanas, se tiene asociado un gran costo computacional. El clasificador *Random Ferns* (ver 2.2.6), ha demostrado ser una buena opción por su eficiencia y capacidad de manejar múltiples clases.

Figura 8: Arriba se puede ver el conjunto de imágenes sintéticas creadas por Wang et. al. Estas imágenes tienen una dimensión de 48x48 píxeles e intentan imitar a las imágenes reales. Abajo se puede observar un conjunto de imágenes reales recortadas que fueron extraidas del dataset ICDAR. Fuente: Wang et. al. [1].

Para poder reconocer un símbolo, se tienen que extraer aquellas características que lo distingue de otros símbolos (sección 2.2.2), es decir, aquello que lo hace único. Una forma de lograr esto es a través de los gradientes de la imagen (ver apéndice B). En la literatura de clasificación y reconocimiento en imágenes naturales, un vector de características muy empleado es

HOG (*Histogram of Oriented Gradients*). Es un descriptor en $\mathbb{R}^N$ que modela la distribución del gradiente de una imagen. HOG ha demostrado ser una representación robusta de las imágenes en el problema relacionado a la clasificación de imágenes naturales [7].

Para el entrenamiento de *Random Ferns*, se tienen los descriptores HOG de cada imagen para cada una de las 62 clases. Sin embargo, no es suficiente para poder empezar a entrenar ya que *Random Ferns* requiere del uso de descriptores binarios. Es por eso que, siguiendo la propuesta de Wang et. al., se propone binarizar los descriptores HOG en lugar de generar descriptores binarios directamente a partir de imágenes.

El pipeline de reconocimiento finaliza una vez que se ha detectado un carácter, con una técnica denominada *supresión de los máximos* (*NMS* por sus siglas en inglés). Esta técnica se aplica sobre el carácter detectado y es una técnica muy usada en los algoritmos de visión por computadora. Básicamente, suprime los píxeles en la imagen que no son máximos locales, con respecto a los píxeles vecinos, a lo largo de la dirección del gradiente.

3.2.1. Histograma de Gradientes Orientados

Los Histogramas de Gradientes Orientados o HOG, por sus siglas en inglés, son descriptores de características utilizados en visión por computadora y en el procesamiento de imágenes con el objetivo de realizar detección de objetos. Fueron introducidos por N. Dalal y B. Triggs en [7] con el propósito de realizar detección de personas. Sin embargo, su uso no se limita solamente a esa área, sino que pueden ser utilizados en otras áreas como la detección de caracteres tal como hicieron Wang et al. en [1].

Todas las imágenes, como por ejemplo la presentada en la Figura $9A$, contienen estructuras locales cuyas apariencias y formas pueden ser descriptas por la distribución de los gradientes de intensidad, como se puede observar en la Figura $9B$. Un descriptor HOG es un vector compuesto por una combinación de histogramas que representan los gradientes de intensidad en distintas regiones de una imagen. Estos descriptores se obtienen dividiendo a la imagen en regiones de tamaño fijo llamadas celdas, como se puede observar en la Figura $9B$, y posteriormente, por cada celda, calculando un histograma de gradientes para los píxeles en esa celda, Figura $9C$. Luego, se agrupan las celdas en bloques, Figura $9C$, y se normaliza cada uno utilizando la norma L_2. Esto se hace con el objetivo de obtener un descriptor robusto ante los cambios en la iluminación, entre otros. Por ejemplo, consideremos la Figura $9C$. Sea b_{nm} $n, m = 1, \ldots, 3$, un bloque 2×2 tal que

$$b_{nm} = (h_{nm}, h_{n(m+1)}, h_{(n+1)m}, h_{(n+1)(m+1)})$$

donde h_{ij} representa la celda ubicada en la fila i columna j. Como se puede observar, en la figura aparece un área resaltada que hace referencia al bloque b_{11}. La norma L_2 para $b_{11} = (h_{11}, h_{12}, h_{21}, h_{22})$ se obtiene de la siguiente manera:

$$||b_{11}||_2 = \sqrt{h_{11}^2 + h_{12}^2 + h_{21}^2 + h_{22}^2}$$

luego se normaliza el bloque b_{11}

$$b'_{11} = \frac{b_{11}}{||b_{11}||_2}$$

Finalmente, el descriptor HOG se obtiene de concatenar los histogramas obtenidos como muestra la Figura 9*D*.

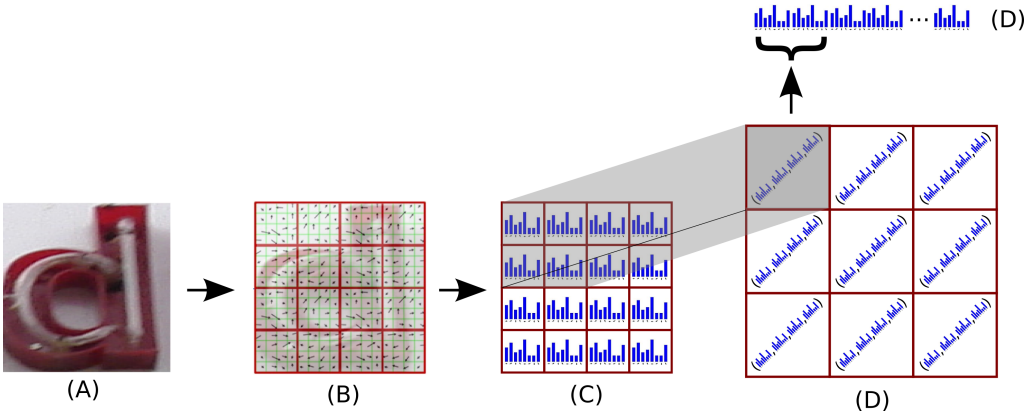


Figura 9: Formación del vector de características HOG

En el área de visión por computadora, los descriptores HOG son considerados estado del arte. Los mismos han demostrado ser útiles en la clasificación de imágenes como se puede apreciar en el trabajo de Wang et al. [1] a donde se ha entrenado el clasificador Random Ferns con estos. Incluso la performance obtenida con estos descriptores en dicho trabajo supera a la mayoría de los descriptores evaluados en el trabajo de De Campos et al. [5] bajo las mismas condiciones de entrenamiento.

3.2.2. Binarización

Una de las condiciones para poder usar los descriptores HOG en el clasificador *Random Ferns*, es que los mismos tienen que estar binarizados. Los descriptores binarios son fáciles de computar, son compactos, se pueden almacenar fácilmente y son fáciles de comparar. En cambio, los descriptores originales tienen alta dimensionalidad y requieren sistemas con más memoria, capacidad de almacenamiento y procesamiento. Muchos sistemas en tiempo real como el reconocimiento de objetos, [13], y en el agrupamiento de puntos clave, [14], han incorporado este enfoque por su utilidad.

Para binarizar los descriptores HOG se requiere de un vector umbral. El mismo, una vez calculado, se encarga de binarizar todos los descriptores tanto del conjunto de entrenamiento como del conjunto de evaluación. Dicho umbral se calcula utilizando solamente el conjunto de entrenamiento y se obtiene de la siguiente manera:

- Dado N descriptores HOG de dimensión D, se forma una matriz $Q \in \mathbb{R}^{N \times D}$ donde cada fila representa un vector.
- Por cada uno de los N vectores originales $v = (v_1, \ldots, v_D)$, se seleccionan X dimensiones al azar de v, con reemplazo, obteniendo de esta manera un vector $z = (z_1, \ldots, z_X)$, *con* $z_i = x_{\hat{i}}$, $\hat{i} \in \{1, \ldots, D\}$, donde $\hat{i}$ es un número al azar en el rango $\{1, \ldots, D\}$.
- Posteriormente, se procede a binarizar z de la siguiente manera. Dado una dimensión z_i de z, se aplica una función de umbralización sobre la *i-esima* columna de Q, la función puede ser el calculo de la mediana, la media, bootstrap, entre otros. Se obtiene de esta manera $w_i \in \mathbb{R}$. Luego, si $z_i \geq w_i$ se le asigna 1 a esa dimensión y 0 en caso contrario. El vector umbral W surge de aplicar X veces la función de umbralización sobre la matriz Q, recordar que hay dimensiones repetidas en z.

A la binarización se la puede expresar con la siguiente composición de funciones:

$$h \circ g : \mathbb{R}^D \to \{0,1\}^X$$

donde

$$h\text{:}\ \mathbb{R}^X \to \{0,1\}^X$$

$$g\text{:}\ \mathbb{R}^D \to \mathbb{R}^X$$

g: $\mathbb{R}^D \to \mathbb{R}^X$ es una función definida de la siguiente manera. Dado un vector $x = (x_1, \ldots, x_D)$:

$$g(x) = z = (z_1, \ldots, z_X),\ con\ z_i = x_{\hat{i}},\ \hat{i} \in \{1, \ldots, D\} \tag{3.2.1}$$

En 3.2.1 $\hat{i}$ representa un entero elegido de manera aleatoria en el rango $\{1, \ldots, D\}$. Luego, una vez obtenido z, aplicamos la función h

$$(y_1, \ldots, y_X) = h(z) = (h_1(z_1), \ldots, h_X(z_X)) \tag{3.2.2}$$

donde

$$h_i(z_i) = \begin{cases} 1 & \text{si } z_i \geq w_i \\ 0 & \text{caso contrario} \end{cases}$$

$w_i \in \mathbb{R}$ representa al valor obtenido de haber aplicado una función de binarización a la columna *i-esima* de la matriz Q. Las funciones de binarización que se utilizan en este trabajo y que forman los vectores umbrales son: *media*, *mediana*, *exponencial* y *bootstrap*. A continuación se procede a explicar en detalle cada función.

Dada la columna "*i-esima*" de la matriz Q representada por $C_i = (c_{i_1}, \ldots, c_{i_j})$, con $j = 1, \ldots, N$ luego:

- **Media:**

  La **media** es una función *b:* $\mathbb{R}^N \to \mathbb{R}$ tal que:

  $$b(C_i) = \frac{\sum_{j=1}^{N} c_{i_j}}{N}$$

- **Mediana:**

  La **mediana** es una función *b:* $\mathbb{R}^N \to \mathbb{R}$ cuyo único requisito es que el vector esté ordenado. Sea *dim(C)* la dimensión del vector C, luego:

  $$b(C_i) = \begin{cases} \frac{c_{i\frac{dim(C)}{2}} + c_{i\frac{dim(C)}{2}+1}}{2} & \text{si dim(C) es par} \\ \frac{c_{i\frac{dim(C)}{2}}}{2} & \text{caso contrario} \end{cases}$$

- **Bootstrap**

  Al igual que la *media* y la *mediana*, **bootstrap** es una función *b:* $\mathbb{R}^N \to \mathbb{R}$ tal que, dado $c_{i_k} \in C_i$ una dimensión aleatoria de C_i luego:

  $$b(C_i) = c_{i_k}$$

- **Exponencial**

  La elección de la distribución exponencial se basa en las observaciones de que las dimensiones, de manera individual, se las puede aproximar con una distribución exponencial, ver Figura 10. Para una dimensión en particular, el parámetro lambda fue estimado en base a un criterio de máxima verosimilitud a partir de las muestras para esa dimensión.

$$\hat{\lambda} = \frac{\sum_{j=1}^{N} c_{i_j}}{N}$$

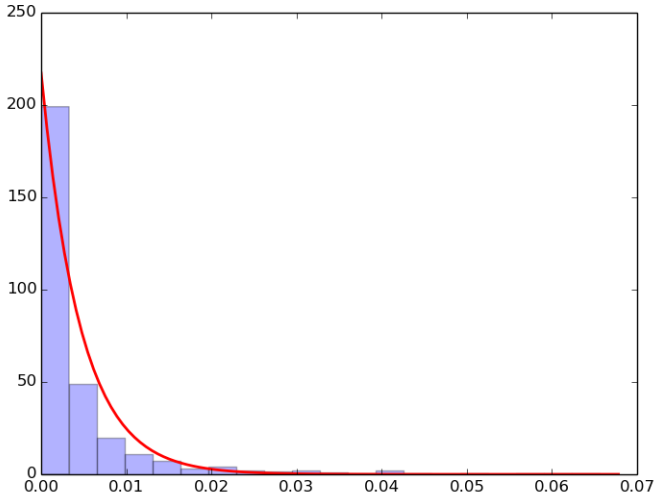


Figura 10: Histograma obtenido para una dimensión individual y la aproximación de esta a una distribución exponencial a través de la curva en rojo.

3.3. Arquitectura del sistema

En las próximas subsecciones, se procederá a explicar cuestiones de implementación que se tuvieron en cuenta, en este trabajo, al momento de resolver el problema del reconocimiento de caracteres.

3.3.1. Reconocimiento de caracteres

Pipeline de procesamiento

La implementación que se realizó en este trabajo, está basada en un pipeline similar al que realizaron Wang et al. que comprende de dos etapas: la de entrenamiento y la de evaluación. La primera esta constituida por la generación de datos sintéticos que forman el conjunto de entrenamiento, la extracción de las características, la binarización y el entrenamiento del clasificador. La segunda consiste en la evaluación del clasificador que abarca desde la extracción y binarización de las características del conjunto de prueba hasta la evaluación del clasificador para cada imagen de este conjunto. Los pipelines se pueden apreciar en las Figuras 11 y 12.

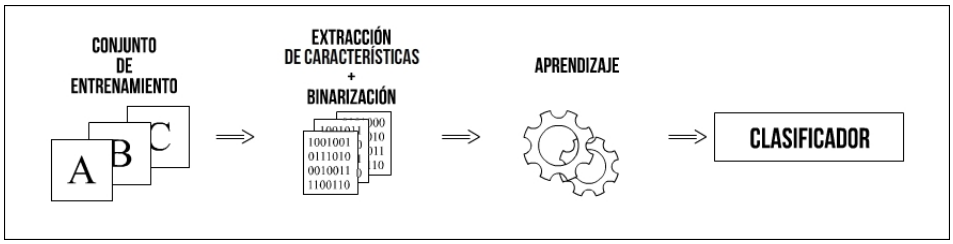


Figura 11: Pipeline de entrenamiento que se sigue en el trabajo a la hora de entrenar el clasificador.

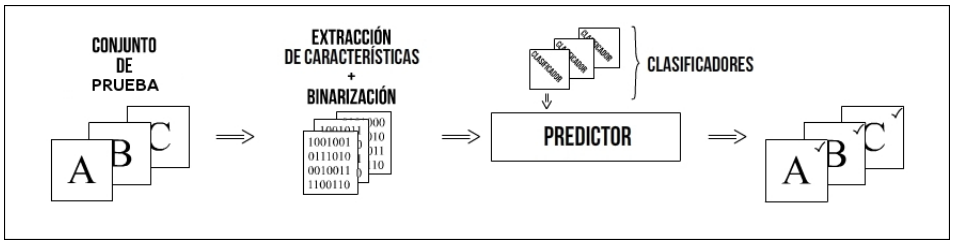


Figura 12: Pipeline de evaluación que se sigue en el trabajo al momento de evaluar el clasificador.

Conjunto de entrenamiento

La primera etapa de este pipeline de procesamiento consiste en la obtención de un conjunto de entrenamiento para utilizar en el entrenamiento del clasificador. Se destacan dos grupos de imágenes al momento de entrenar, el de caracteres segmentados de escenas naturales y el de caracteres sintéticos. A continuación se procederá a explicar cómo se genera el grupo de datos sintéticos.

Generación de datos sintéticos

Supongamos que se tiene un conjunto limitado de imágenes reales de caracteres. Uno querría poder trabajar con más imágenes. Sin embargo, el tiempo y el esfuerzo que conlleva la recolección de este tipo de datos lo hacen una tarea poco práctica. Se busca una forma de automatizar la forma en que se pueden obtener este tipo de datos. El objetivo detrás de la generación de los datos sintéticos es agregar vistas de los caracteres que no están reflejadas en el conjunto de imágenes reales original pero que son plausibles de ser observadas en la realidad.

Inicialmente, se tiene como base un conjunto que contiene imágenes de caracteres de diversas tipografías generadas de forma artificial. Lo que se busca es aplicar a cada imagen un conjunto de transformaciones afines aleatorias con el objetivo final de obtener imagenes nuevas cuyas apariencias sean cercanas a una real. Una transformación afín consiste en una transformación lineal seguida de una traslación.

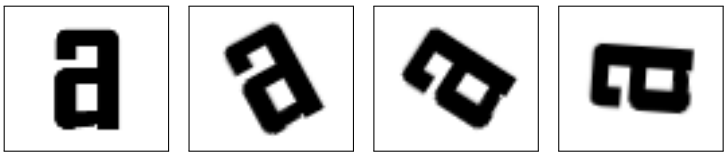
Existen dos tipos de transformaciones: *geométricas* y *fotométricas*. Las primeras son de la forma $x \mapsto Mx+b$, donde M es una transformación lineal y b es un vector. Se usan las multiplicaciones entre matrices para representar las transformaciones lineales y la suma de vectores para representar las traslaciones. M es una matriz 2×2 y permite realizar transformaciones en imágenes de 2 dimensiones, x es un vector en $\mathbb{R}^2$ que representa las coordenadas (x, y) de un píxel en la imagen. En este trabajo, para generar los datos sintéticos se hacen uso de las siguientes transformaciones geométricas:

Rotación La rotación es una transformación que consta de rotar la imagen en un ángulo θ en torno al origen, el centro de la imagen. El parámetro usado son los radianes y las diferentes variaciones se pueden apreciar en la Figura 13. Teniendo en cuenta la definición formal de una transformación afín anterior, la rotación se puede representar formalmente de la siguiente manera:

$$M = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} b = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

a donde se puede observar el parámetro θ que representa el grado de inclinación. El vector b es nulo ya que no hay traslación alguna.

Escala La escala es una transformación que busca modificar el tamaño del objeto relativo a la ventana. Formalmente, se puede representar por la



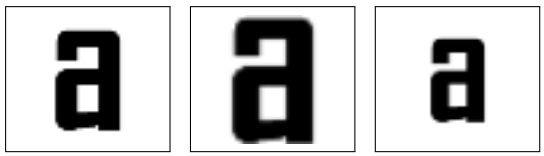
(a) (b) (c) (d)

Figura 13: Rotación de un caracter. (a) Imagen original. (b) Imagen rotada 0.5 radianes. (c) Imagen rotada 1 radian. (d) Imagen rotada 1.5 radianes

siguiente configuración de la matriz M:

$$M = \begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix} \tag{3.3.1}$$

donde a y d representan los cambios en los ejes x e y de la imagen respectivamente. Se puede apreciar la aplicación de esta transformación en la Figura 14.



(a) (b) (c)

Figura 14: Cambio de escala de un carácter. (a) Imagen original. (b) Imagen ampliada 1,2 veces su tamaño (c) Imagen reducida a 0,8 veces su tamaño .

Transvección La transvección es una transformación donde la imagen se inclina en un solo eje. Es una función que toma un punto con coordenadas (x, y) al punto $(x + ny, y)$, donde n es un parámetro fijo, denominado el factor de inclinación. El efecto es el desplazamiento de todos los puntos horizontalmente en una cantidad proporcional a su coordenada y. Todo punto encima del eje x es desplazado a la derecha si $n > 0$, y a la izquierda si $n < 0$. La matriz M va a ser de la forma:

$$M = \begin{bmatrix} 1 & n \\ 0 & 1 \end{bmatrix}. \tag{3.3.2}$$

Se puede observar en la Figura 15 los diferentes efectos de aplicar diferentes valores de n en la aplicación de la transvección en una imagen.

(a) (b) (c) (d) (e)

Figura 15: Transvección de un carácter.

Traslación La traslación es el último tipo de transformación afín que se usa en este trabajo y a diferencia del resto de las transformaciones, esta no hace uso de M, sino de b que es el vector de desplazamiento. Formalmente:

$$M = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

donde b_1 y b_2 representan el movimiento en el eje x e y respectivamente.

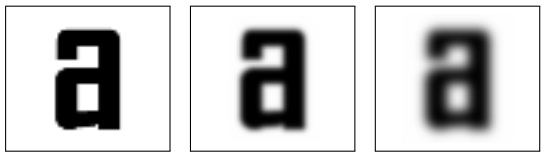
El siguiente tipo de transformaciones son las *fotométricas*. Este tipo de transformaciones se caracterizan por surgir de variaciones en la iluminación de una imagen, es decir, una variación en la función de intensidad de la misma. En general, este tipo de transformaciones no cambian la forma de la imagen. Por ejemplo, el ruido Gaussiano se origina, entre otras cosas, por la pobre iluminación del entorno, en el caso de las imágenes digitales. Dentro de este grupo podemos encontrar las siguientes transformaciones:

Suavizado Gaussiano Es una técnica que consta de la difuminación de una imagen a partir de la convolución con una función Gaussiana. Se usa principalmente para reducir el ruido en una imagen simulando el efecto de "blur". Dadas las coordenadas de un punto (x, y) en una imagen, la función de suavizado es la siguiente:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

donde σ es la desviación estándard de la distribución gaussiana. En la figura 16, se puede observar el efecto para valores crecientes de σ.

Ruido Gaussiano El ruido en una imagen es una variación en la información sobre el color o la iluminación en la misma. En imágenes reales, puede ser producido por un sensor, la circuitería de un escáner o una cámara digital. Condiciones de poca iluminación o interferencia electromagnética durante



(a) (b) (c)

Figura 16: Aplicación de blur o suavizado gaussiano a un carácter. (a) Imagen original. (b) Imagen suavizada con un valor de $\sigma = 1$. (c) Imagen aún más suavizada con un valor de $\sigma = 2$.

la transmisión de las imágenes son factores para que aparezcan este tipo de distorsiones. El ruido Gaussiano es un tipo de ruido que se caracteriza por tener una distribución Gaussiana. La Figura 17 presenta 2 ejemplos de imágenes de caracteres con esta transformación. A medida que el parámetro σ aumenta, el ruido en la imagen se hace más notorio.

Formalmente se puede expresar esta transformación de la siguiente manera:

$$x \rightarrow x + \epsilon$$

donde ϵ es una muestra asociada a una variable aleatoria con distribución $\mathcal{N}(0, \sigma^2)$

(a) (b) (c)

Figura 17: Aplicación de ruido gaussiano a un carácter. (a) Imagen original. (b) σ=15. (c) σ=30 .

Extracción de características y binarización

La siguiente etapa en el pipeline es la extracción del vector de características (ver sección 2.2.2), de cada imagen del conjunto de entrenamiento y la binarización posterior de dicho descriptor (ver sección 3.2.2).

Una de las condiciones para poder extraer el vector de características de una imagen con HOG, es que esta esté en escala de grises. Luego, dada una imagen, ventana que contiene un carácter, se extrae el descriptor de características a través de HOG y se obtiene un vector $v \in \mathbb{R}^K$, donde K depende de los parámetros del algoritmo. Posteriormente se procede como se detalla en la sección 3.2.2, a donde se obtiene el umbral que después se utiliza para binarizar.

Supongamos un conjunto de entrenamiento con N imágenes. Luego, para generar el vector umbral en los experimentos, se propuso la siguiente implementación basada en la detallada en la sección 3.2.2.

- Dado N descriptores HOG de dimensión D, se forma una matriz de tamaño $N \times D$ donde cada fila representa un vector.
- Se seleccionan X columnas al azar de la matriz, con reemplazo.
- Se aplica una función de umbralización sobre cada columna, la función es una de las detalladas en 3.2.2. Se obtiene de esta manera un vector nuevo W de dimensión X tal que cada elemento de W está compuesta por un par (z, y). y es un número $0 \leq y \leq D$ que representa una de las columnas elegidas de la matriz y z representa el valor resultante de haber aplicado la función elegida a dicha columna. Cabe aclarar que X puede ser mayor o menor a D por lo cual el umbral W puede tener mayor o menor dimensión al final.
- Posteriormente, dicho umbral W se utiliza para binarizar los vectores originales de manera sencilla: sea v_j, con $j \in \{1, \ldots, N\}$, uno de los N vectores originales y tal que $v_j = d_1, d_2, \ldots, d_D$. Luego se compara cada entrada del umbral W con la y-ésima entrada del vector v_j. Si $d_y \leq z$ se asigna 0, caso contrario 1. De esta manera binarizamos el vector v_j obteniendo un nuevo vector binario de dimensión X.

Entrenamiento del clasificador

En esta etapa, al igual que los autores del trabajo original, se utiliza el clasificador Random Ferns. Para poder entrenar el clasificador, los vectores modificados de la etapa anterior se almacenan en un diccionario. Este último está estructurado como un conjunto de diccionarios anidados y representa la "base de datos" del sistema.

Dado que los vectores binarizados pertenecen al conjunto $\{0, 1\}^N$, si los quisiéramos almacenar en una sola tabla por cada clase, cada una debería

tener 2^N entradas. Si N es muy grande esto sería imposible por la cantidad de memoria necesitada para mantener las tablas. Luego, basándonos en lo explicado sobre Random Ferns en 2.2.6, la solución pasa por dividir cada vector en M grupos de longitud $S = \frac{N}{M}$. Con esto, un vector completo se almacena en M tablas diferentes, tablas correspondiente a la clase del vector. En total estaríamos trabajando con $M \times 62$ tablas.

Dado un vector $v = (v_1, \ldots, v_S)$ dividido en M grupos de longitud S, es decir, $v = (w_1, \ldots, w_M)$, donde $w_i = (w_i, \ldots, w_{i+S})$. Sea z la clase de v y sean, t_i $i = 1, \ldots, S$, las tablas de z. Cada grupo w_i, $i = 1, \ldots, M$, va a tener una entrada en su tabla correspondiente z_i equivalente a su representación decimal. Dicha entrada en la tabla tiene un contador que se va incrementando en 1 a medida que se acceda a esta entrada. Luego, a medida que se van "entrenando" las tablas de cada clase, se puede dar la situación, más a medida que la dimensión de cada grupo crece, que algunas entradas no se hayan accedido nunca. Esto supone un problema para los cálculos posteriores del clasificador, por lo que se propone inicializar cada entrada de cada tabla con un valor $\alpha \neq 0$.

Evaluación del clasificador

Para la evaluación del clasificador, se toma como conjunto de test el descrito en la sección de descripción del dataset. A cada imagen del conjunto de test, se le extrae el descriptor HOG, se lo binariza y con el vector resultante se calcula la probabilidad de que dicho vector pertenezca a cada una de las clases involucradas. El cálculo para cada clase es el mismo y consiste en dividir el vector de prueba en M grupos y por cada grupo obtener el valor en la tabla correspondiente. Por cada clase, entonces, tendríamos M valores. Finalmente, se realiza la suma de los logaritmos de cada valor y el resultado es la probabilidad de que ese vector pertenezca a la clase evaluada. Al final se le asigna a la imagen de prueba la clase que haya obtenido la mayor probabilidad.

4. Experimentos

En el presente capítulo se abordan las diferentes etapas que engloban a los experimentos de esta tesis. En las primeras 2 secciones se describen tanto las métricas y protocolos de evaluación utilizados en el dataset como así también los conjuntos de entrenamiento que se utilizan en los experimentos. Posteriormente, se detallan los parámetros que se utilizan en los diferentes experimentos y los resultados obtenidos de los mismos. Por último, se brinda un análisis de los resultados con el fin de evaluar la eficiencia del clasificador en las diferentes pruebas.

4.1. Conjunto de Datos y Protocolo de Evaluación

El dataset utilizado para los experimentos es *Chars74k* creado por T. E. Campos et al. en [5]. El mismo consta de los siguientes tipos de imágenes de caracteres:

- **Escenas naturales:**

  Este conjunto de imágenes está dividido en 2 grupos: las imágenes buenas y las malas. A su vez, cada grupo contiene 62 subgrupos que representan a los caracteres alfanuméricos, en adelante clases, y que contienen varias imagenes representativas del carácter. En el presente trabajo, se hace uso de un subconjunto de todas estas imágenes creado por los autores de *Chars74k*, llamado *Chars74k-15*.

  Chars74k-15 consta de un conjunto de entrenamiento y uno de prueba, disjuntos entre si, con la misma estructura explicada para *Chars74k* y contienen 15 imágenes por clase cada uno. Las imágenes de ambos conjuntos son una mezcla de las imágenes buenas y malas explicadas anteriormente. En los experimentos de este trabajo, se ajustaron las dimensiones de cada imágen a 48×48 píxeles. De esta forma, los descriptores de características de cada imágen tienen la misma dimensión. Cabe aclarar que el conjunto de evaluación descrito, se mantiene fijo durante todos los experimentos que se realizan en este trabajo. En la Figura 18 se muestra un ejemplo del conjunto de imágenes, buenas y malas, para el caracter "K".

- **Sintéticos:**

  El conjunto de caracteres sintéticos, proviene de diversas fuentes de computadora. Este conjunto tiene una estructura similar a la anterior pero con algunas diferencias. La primera es que no se diferencian

Figura 18: Imágenes buenas (arriba) y malas (abajo) del carácter “K”.

caracteres buenos de malos. La segunda es sobre la estructura en que estan representadas en el dataset *Chars74k*. Cada clase contiene 1000 imágenes de caracteres de más de 40 fuentes diferentes en 3 estilos distintos: normal, itálica y negrita.

En este trabajo, se utilizan estas imágenes para generar los datos sintéticos que se pueden observar en la Figura 19. Como se detalló en la sección 3.3, para poder realizar esto se hacen uso de diferentes transformaciones fotométricas y geométricas. El objetivo, es intentar dar a la imagen el aspecto lo más parecido a una imagen natural.

Figura 19: Imágenes sintéticas. Arriba se pueden ver las imágenes sintéticas originales. Abajo se aprecia un conjunto de imágenes luego de aplicar las diferentes transformaciones.

- **Manuscritos:**

  Este conjunto esta compuesto por más de 3400 caracteres escritos a mano. Tiene la misma estructura de grupos que los caracteres sintéticos

y fueron extraidas utilizando una tablet-PC. Un ejemplo de este conjunto se puede observar en la Figura 20. Cabe aclarar que en este trabajo no se utilizan este tipo de imágenes.

Figura 20: Imágenes de caracteres manuscritos.

Lo que calcula sobre el conjunto de evaluación, compuesto por imágenes naturales, es el grado de precisión del clasificador al haber entrenado al mismo con los conjuntos de entrenamiento explicados. Es decir, se busca ver la cantidad de instancias correctamente clasificadas sobre el total de evaluadas.

4.2. Baselines

En esta sección se detallan los experimentos y los resultados obtenidos de realizar una reimplementación del trabajo de Wang et. al. en [1]. El objetivo es definir el baseline para el resto de los experimentos con imágenes reales y sintéticas. Así también, se definen los parámetros comunes a todos los experimentos como la configuración de HOG, α, parámetro de inicialización de las tablas, entre otros.

Posteriormente se van a mostrar y explicar los resultados de los experimentos con imágenes reales. Se detalla el impacto de los diferentes esquemas de binarización con el fin de seleccionar el más adecuado para el resto de los experimentos.

4.2.1. Reimplementación de Wang et. al.

En este experimento se utilizan, para el entrenamiento del clasificador, solamente imágenes de caracteres segmentados de escenas naturales e imágenes sintéticas, ver 4.1, siguiendo lo detallado por Wang et. al. en [1]. La evaluación del clasificador se realiza sobre un conjunto de prueba proporcionado por el dataset *Chars74k-15* y se mantiene fijo durante todos los experimentos del presente trabajo.

De la implementación de los autores, se pudieron obtener los siguientes parámetros para los experimentos:

Implementación	**Score**
Wang NATIVE+FERNS	54 %
Impl. propia NATIVE+FERNS	50 %
Wang SYNTH+FERNS	47 %
Impl. propia SYNTH+FERNS	43 %

Cuadro 1: Resultados obtenidos en la reimplementación del trabajo de Wang et al.

- **Cantidad de grupos** (M): 256.
- **Bits por grupo** (bpg): 6.

Estos dos valores generan vectores de características de longitud 1536.

Para la función HOG, se utilizan dos parámetros, la cantidad de *orientaciones* y la cantidad de *celdas por bloque*. En los experimentos realizados, se prueban los siguientes conjuntos de valores:

- Orientaciones: $o \in \{8, 9\}$
- Celdas por bloque: $cpb \in \{4, 9\}$

En cuanto al método para binarizar los descriptores HOG, luego de haber leído la implementación provista por los autores, se establece como método la generación de variables aleatorias uniformes. Este enfoque, resultó ser el más adecuado debido a que es el que más se asemeja a lo que hicieron los autores en [1] .

Finalmente, el valor α para la inicialización de las tablas es un parámetro que no está especificado pero es necesario para poder entrenar el clasificador. En este experimento se probaron los siguientes valores de α con el fin de establecer el mejor valor.

- $\alpha \in \{0,01; 0,1; 1\}$.

Teniendo en cuenta estos parámetros, se obtuvieron los resultados que se pueden apreciar en el cuadro 1. En la misma, expresamos como "NATIVE+FERNS" a los resultados obtenidos al haber entrenado y evaluado el clasificador Random Ferns con imágenes reales. De la misma manera, "SYNTH+FERNS" se refiere a los experimentos donde se usaron 1000 imágenes sintéticas durante el entrenamiento.

Como se puede ver en base a los resultados expuestos en el cuadro 1, hay diferencias significativas entre los resultados de ambas implementaciones

que se deben a varios factores. Existen ciertos parámetros que no están especificados en el trabajo de Wang et al. y que se detallan a continuación:

- **Método de binarización**: El hecho de que en la reimplementación se utilizó un lenguaje diferente, en este caso se usó Python, genera ciertos problemas. El uso de funciones que generan valores numéricos aleatorios depende mucho de la implementación de dichas funciones. Estas pueden variar entre los lenguajes por lo cual los resultados obtenidos son diferentes. La binarización hace uso de valores aleatorios con lo que se generan estas diferencias.

- α: es el parámetro para la inicialización de las tablas que influye en la performance.

- **Implementación de HOG**: la implementación de HOG que usan los autores es la desarrollada por Piotr Dollár, [17], la cual es una variación del método original propuesto por Dalal & Triggs en [7]. En la reimplementación se utiliza la función HOG extraída de la librería de python scikit-images, que también está basada en [7] pero difiere de la usada por los autores. Una de las diferencias entre ambas funciones está en que la desarrollada por Piotr Dollar acepta imágenes a color y la que se usa en este trabajo sólo acepta imágenes en escala de grises. Otra diferencia entre estas implementaciones está en la longitud del vector resultante. Usando los mismos parámetros en ambas funciones, la desarrollada por Piotr Dollar devuelve vectores con una longitud 4 veces mayor a los devueltos por la función HOG usada en el presente trabajo. Esto se debe a que usa un esquema diferente de normalización.

Ante esta situación se decide por establecer como baselines los resultados obtenidos por la configuración más cercana a la usada en [1]. Teniendo en cuenta los parámetros conocidos, se establece además las siguiente configuración:

- **HOG**: se consideran 8 orientaciones y 9 celdas por bloque como los parámetros que mejores resultados produjeron.

- α: se establece en 0,01 pues es el valor que más impacto positivo produjo en estos experimentos.

El esquema de binarización utilizado en este experimento sirve solamente para establecer el baseline para el resto de los experimentos. En la siguiente subsección se evalúan, a través de experimentos con imágenes

reales, nuevos métodos de binarización, explicados en 3.2.2, y se va a considerar como baseline el valor 0,50. De la misma manera, en la sección de los experimentos con imágenes sintéticas, el baseline pasará a ser 0,43.

4.2.2. Evaluación de Esquemas de Binarización

Esta subsección tiene como objetivo evaluar los 4 esquemas de binarización detallados en 3.2.2. Además, como resultado de los experimentos, se establecerá cúal es el tamaño óptimo para los grupos o *ferns* y cúal es la mejor dimensión para los vectores de características. Para esto, se entrena y evalua al clasificador *Random Ferns* con imágenes reales de la misma manera que en la subsección anterior.

Inicialmente, se van a mostrar 4 gráficos, que corresponden a los métodos utilizados en la binarización. Con esto se busca ver qué método devuelve los mejores resultados. Estas figuras, reflejan la diferencia en performance al considerar grupos de diferente dimensión $B/G \in \{1, 4, 8, 10\}$. Con esto se busca establecer, para cada caso, qué dimensión arroja los mejores resultados. Cabe aclarar que cada gráfico refleja la media de los resultados de 5 corridas del experimento junto con las barras de error a 1 desviación estándar. La Figura 25 reúne los mejores resultados de clasificación de las primeras cuatro figuras para establecer una comparación más precisa. En la tabla 2 se muestra un resumen de los mejores valores.

Dada la gran cantidad de parámetros que se manejan en estos experimentos, se va a dejar detallado en el análisis cuales son los mejores y se los va a usar para los siguientes experimentos.

Teniendo en cuenta los resultados de los primeros 4 gráficos, a simple vista se puede observar que los resultados obtenidos usando la media como método para binarizar los vectores son los mejores, llegando estos a un máximo cercano al 55 % de precisión. En el caso opuesto, el usar la mediana arrojó los peores resultados, ya que, en el mejor de los casos, se supera el 47 % a diferencia de bootstrap que llega a 51 % . El uso de la distribución exponencial en los casos donde la longitud de los grupos es baja, 1 bit por grupo, no supera a la media en performance pero aumenta a medida que aumentamos este valor, casos $\{4, 8, 10\}$ bits por grupo. Los valores que podemos encontrar en 23 se acercan a los de la media llegando a un máximo en la precisión de 53 %. En cuanto al uso de bootstrap, se puede apreciar que los resultados se mantienen por debajo de los obtenidos con la distribución exponencial y la media con valores que rondan entre 45 % y 52 %, en el mejor de los casos.

En cuanto a las dimensiones del vector binario evaluadas en el experi-

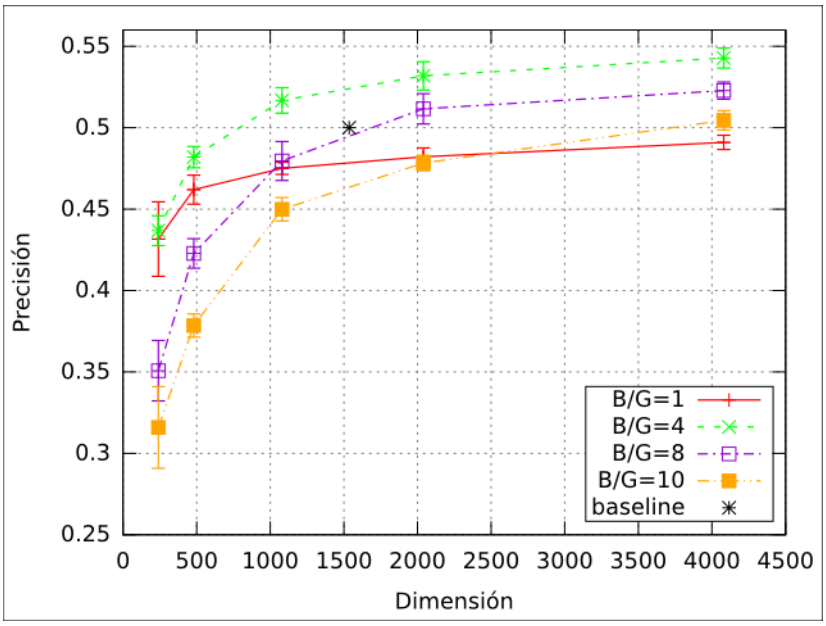


Figura 21: Resultados usando la media

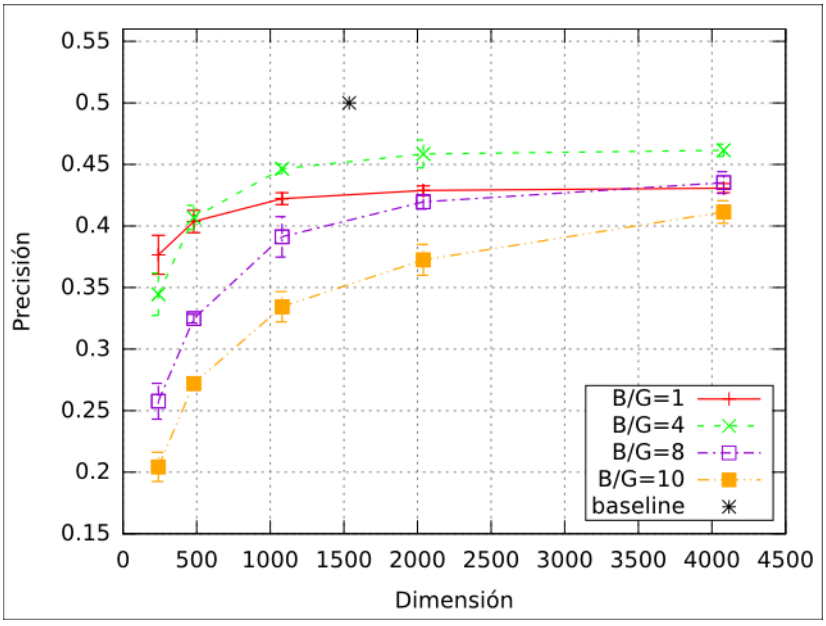


Figura 22: Resultados usando la mediana

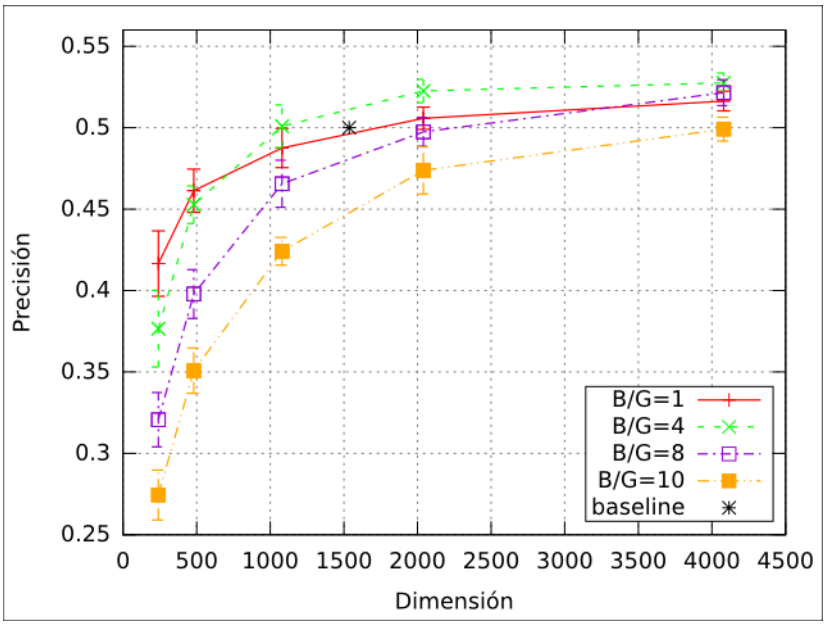


Figura 23: Resultados usando la distribución exponencial

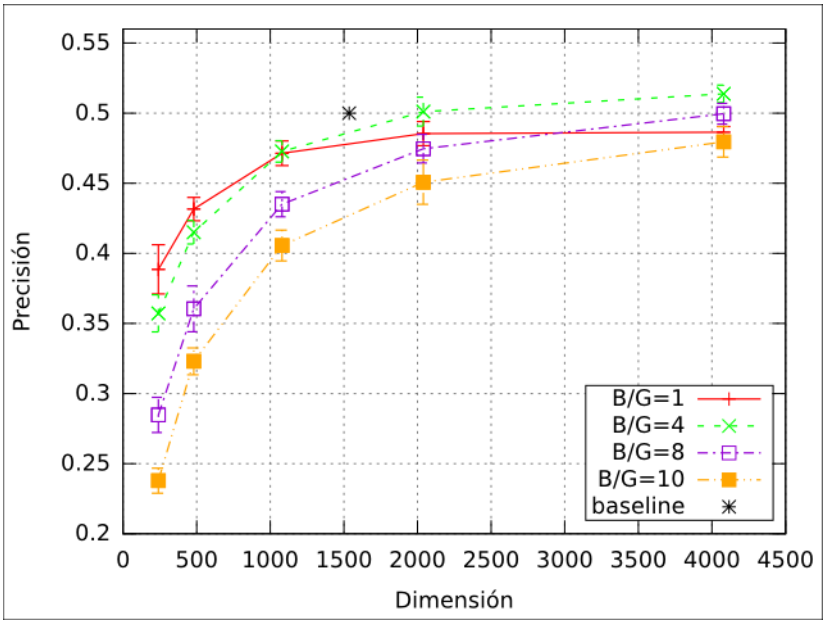


Figura 24: Resultados usando bootstrap

mento $\{240, 480, 1080, 2040, 4080\}$, es claro que a medida que aumentamos la longitud de los vectores aumenta la performance. Se puede ver que, cuando se usan vectores de longitud reducida, en este caso 240, se marca una diferencia importante en la precisión entre los experimentos realizados con grupos de dimensionalidad baja $\{1, 4\}$ y alta $\{8, 10\}$. Es decir, dejando fija la dimensión de los vectores en 240, mientras más chico es el tamaño de los grupos mejor es el resultado, se puede observar en cualquiera de las figuras. En el otro extremo, si usamos grupos de 10 bits la precisión disminuye considerablemente. La misma relación se mantiene a medida que aumentamos el tamaño de los vectores de características hasta cierto punto. Por ejemplo, en la Figura 21 es claro que a partir del uso de 1080 como tamaño de vector en adelante, el uso de grupos de longitud mayor arroja mejores resultados. Lo mismo sucede en las otras figuras en diferentes puntos.

En cuanto al uso del valor 4080, se puede ver que no hay un aumento considerable en la performance que lo diferencie del uso de 2040. Con esto en mente, es conveniente hacer uso de este último valor ya que usa vectores mucho más chicos incrementando la eficiencia en el cómputo.

El enfoque del clasificador Ferns es semi-naïve, es decir, a diferencia del clasificador Naïve Bayes, se considera que hay cierta relación de dependencia o correlación entre las dimensiones del vector de características. Por lo tanto, tal y como se explica en 2.2.6, los vectores se dividen en grupos de dimensión S. Esto con el objetivo de ir un paso más allá de la independencia propuesta en Naïve Bayes donde cada dimensión es independiente y así poder capturar estas correlaciones. El problema con los vectores binarizados de dimensión reducida, es que almacenan menos información sobre los datos de la imagen a comparación de los vectores cuya longitud es mayor. En cada grupo, se capturan las correlaciones entre las dimensiones involucradas en estos, es decir, el nivel de dependencia o relación que existen entre las mismas. Es por eso que al aumentar la dimensionalidad de los vectores en el proceso de binarización, capturamos más estructuras de correlación de los datos. Basándonos en esto, y como se puede observar en los gráficos expuestos, los grupos de 4 bits son los más adecuados en la clasificación en la mayoría de los casos.

Del análisis realizado surge que la mejor configuración está dada por: *bits por grupo* $= 4$ y *dimensión del vector* $= 2040$. Se puede observar en la Figura 25 las mejores curvas para cada método, utilizando la mejor configuración, a excepción de la longitud del vector con el objetivo de ver la curva de crecimiento. Queda claro que la media se puede establecer como el mejor método para la binarización por lo que en los próximos experimentos se procederá a trabajar con este método de binarización. La diferencia en precisión

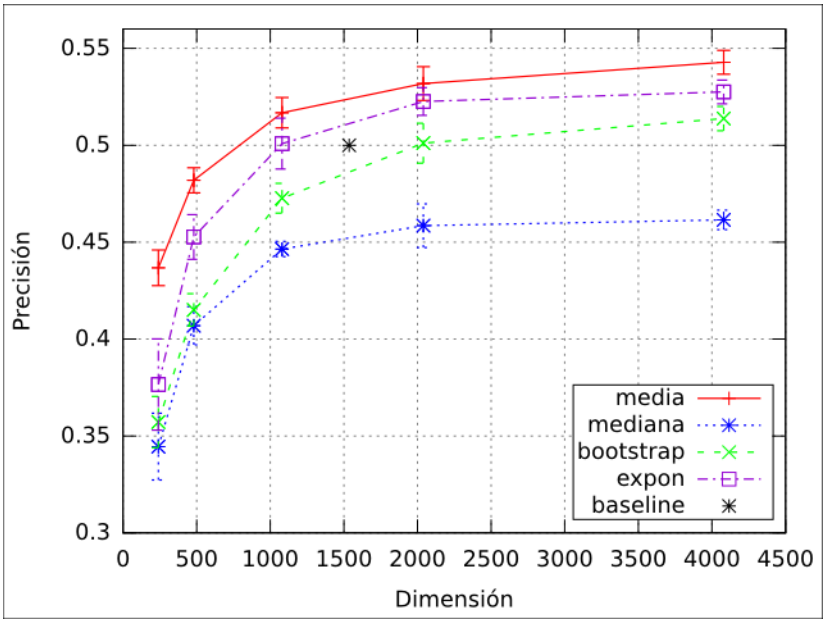


Figura 25: El gráfico muestra las mejores curvas de los gráficos presentados con la mejor configuración

entre usar vectores de longitud 2040 y 4080 no es notable en los 4 casos por lo cual como dijimos anteriormente, el uso de 2040 dimensiones permite obtener buenos resultados con un menor costo computacional. En todos los casos, salvo cuando se usa la mediana como método de binarización, se supera el baseline propuesto en diferentes puntos. La media logra una diferencia de 4 % con respecto al baseline pero usando vectores de dimensión 4080, la misma diferencia baja a un 3 % si consideramos 2040.

La tabla 2 muestra un resumen de los valores obtenidos con la mejor configuración para cada método.

4.3. Uso de imágenes sintéticas

En esta sección se busca ver el impacto producido por las imágenes sintéticas en la clasificación. Es decir, se buscar entrenar al clasificador con dos grupos de imágenes. El primer conjunto está conformado por imágenes sintéticas alteradas cuyo proceso de creación fue detallado en la sección 3.3. El segundo esta conformado por una combinación del primer grupo con el conjunto de imágenes reales usado en 4.2.

El grupo de imágenes de fuentes sintéticas de computadora que se usa

NATIVE + FERNS	Score
Media	53 %
Mediana	46 %
Exponencial	52 %
Bootstrap	50 %

Cuadro 2: Tabla comparativa entre los diferentes métodos propuestos para la binarización en la clasificación de caracteres en escenas naturales usando la mejor configuración de parámetros.

para crear los caracteres sintéticos fue extraído del dataset mencionado en 4.1 y consiste en 1000 imágenes por clase. De este conjunto se crearon 6 variaciones diferentes, con el objetivo de observar el impacto que tiene la cantidad de caracteres sintéticos al momento de clasificar imágenes de caracteres naturales. La cantidad de muestras por clase (variaciones) en cada conjunto de entrenamiento es la siguiente:

- Muestras por clase : $mpc \in \{50, 100, 250, 500, 1000, 2000\}$

Estas variaciones son puestas a prueba en el primer experimento de la sección.

El segundo conjunto de imágenes sintéticas se creó con la finalidad de observar el impacto en la clasificación de integrar, en diferentes proporciones, imágenes sintéticas y reales al conjunto de entrenamiento. Dada la escasa cantidad de imágenes naturales para el entrenamiento, 15 en total por cada clase, se busca complementar esto integrando datos sintéticos. Por consiguiente, se decide crear 9 conjuntos de entrenamiento donde en cada uno se incorporan diferentes proporciones de imágenes sintéticas a las 15 reales existentes. La proporción de cada conjunto de entrenamiento es:

$$(15, x) \ con \ x \in \{0, 3, 8, 15, 30, 75, 150, 300, 750, 1500\}$$

El primer elemento de la tupla $(15, x)$ corresponde a la cantidad de imágenes reales, cuyo valor es fijo. El segundo elemento de la tupla hace referencia a la proporción de caracteres sintéticos. Es fácil observar que a medida que aumentamos la proporción, la cantidad de caracteres naturales se va volviendo cada vez más despreciable. El hecho de no considerar imágenes sintéticas inicialmente $(15, 0)$ tiene como finalidad el observar el cambio que se produce desde el inicio cuando no hay caracteres sintéticos en el entrenamiento.

Este segundo grupo de imágenes es puesto a prueba en el último experimento de la sección.

Los parámetros utilizados durante la creación de las imágenes sintéticas son, como se explicó en 3.3, escala, rotación, transvección, traslación, blur y ruido Gaussiano. Dado que hay un abanico bastante grande de valores para asignarles a estas transformaciones, se decidió por asignarle a cada una un rango de valores razonable. Dado que los autores en [1] no especifican el rango de valores para las transformaciones que utilizan, se procedió a establecer los siguientes intervalos para todas las transformaciones utilizadas:

- Transvección: $n \in [-0{,}20; 0{,}20]$
- Suavizado gaussiano (blur): $\sigma \in [0; 2]$
- Escala: $a = d \in [0{,}8; 1{,}25]$
- Rotación (en radianes): $\theta \in (-0{,}1; 0{,}1)$
- Ruido Gaussiano: $\sigma \in [1; 30]$

4.3.1. Primer Experimento: Conjunto de imágenes sintéticas

En el primer experimento, se procederán a mostrar los resultados obtenidos de haber utilizado el primer conjunto de imágenes. Los gráficos van a representar la relación entre la cantidad de imágenes sintéticas por clase, dado por *IPC* en las siguientes figuras, y la precisión de clasificación dada por los 4 valores a evaluar que son las dimensiones de los grupos. Teniendo en cuenta los resultados anteriores con las imágenes reales, se decidió por trabajar con la mejor configuración encontrada salvo por el parámetro $B/G(bits\ por\ grupo)$. Este último parámetro se mantuvo con el fin de seguir evaluándolo con estos nuevos experimentos.

El baseline con el que se comparan los resultados es el obtenido en 4.2 para las imágenes sintéticas cuyo valor es 0,43.

En cuanto a los experimentos con imágenes sintéticas, basándonos en los resultados plasmados en la figura 26, es interesante observar que a medida que agregamos más imágenes sintéticas por clase, aumenta la performance del clasificador en todos los casos. Inicialmente, el crecimiento es más pronunciado hasta llegar a un punto, a partir de las 1000 IPC, en donde el mismo no es tan notable y en algunos casos tiende a "aplanarse". En base a esto, no tiene sentido trabajar con más de 1000 imágenes por clase, pues computacionalmente es más costoso y no se obtiene una precisión que justifique su uso. En el mejor de los casos, al usar grupos de 10 bits, la diferencia

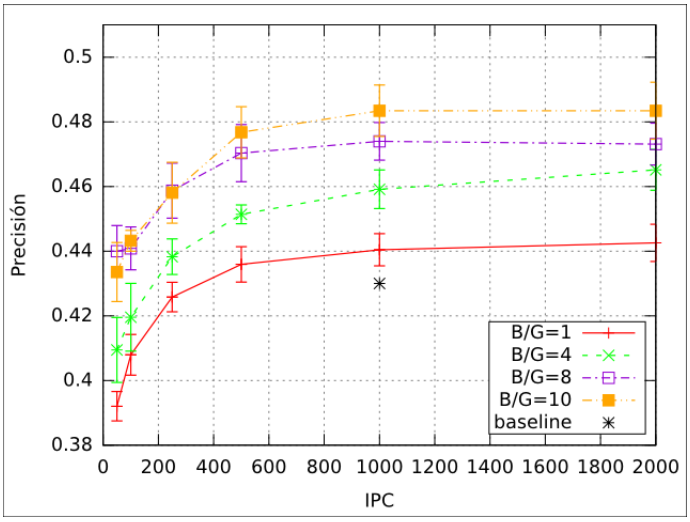


Figura 26: El gráfico muestra los resultados obtenidos de haber utilizado la mejor configuración analizada y haciendo uso de vectores binarizados de longitud 2040.

en performance entre usar 1000 ó 2000 imágenes por clase es nula. El hecho de que se hayan necesitado 2000 muestras por clase para alcanzar una performance cercana al 49 %, también refleja que las imágenes sintéticas no aportan la misma cantidad de información que las reales.

Una particularidad de este experimento con respecto a los anteriores, ver 4.2.2, es que los mejores resultados se obtienen cuando la cantidad de bit por grupo, dado por B/G en la Figura 26, es grande. Esta tendencia se ve en todos los niveles a medida que aumenta IPC.

Por último, en cuanto a la comparación con el baseline, los resultados en la mayoría de los casos son superiores incluso habiendo entrenado al clasificador con menos imágenes por clase, ≤ 1000. Por lo tanto, se puede decir que la configuración elegida hace un mejor uso de los datos de entrenamiento.

4.3.2. Segundo Experimento: Conjunto de imágenes mixto

En este último experimento, se van a mostrar los resultados correspondientes de haber entrenado al clasificador *Random Ferns* con conjuntos de entrenamiento mixtos. Se procederá a mostrar la figura con la mejor configuración. También se mostrarán las matrices de confusión para este caso. Al igual que en 4.3.1, se va a utilizar la mejor configuración obtenida en 4.2 descartando también al parámetro B/G.

El gráfico a continuación presenta el eje x en escala logarítmica para poder apreciar mejor las variaciones que se dan al experimentar con valores cercanos entre sí.

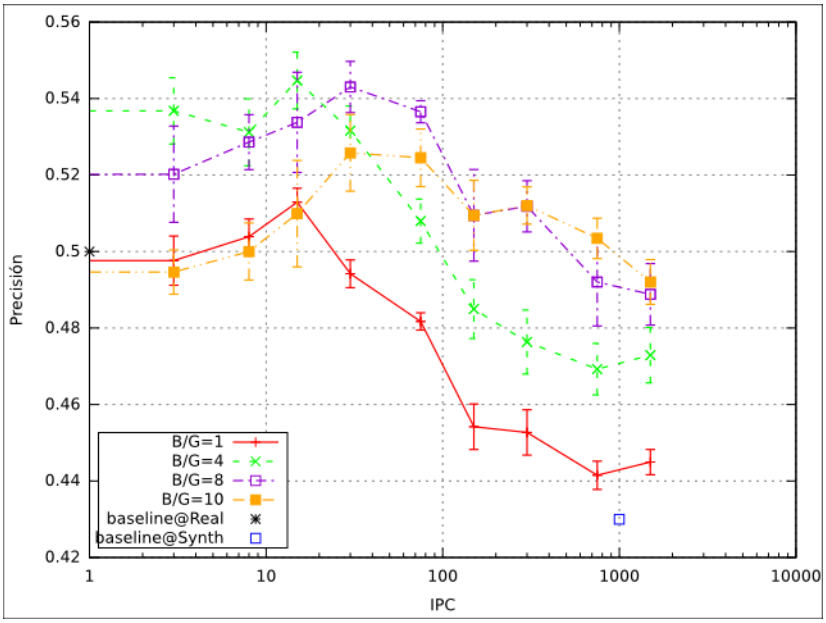


Figura 27: El gráfico muestra la configuración que devolvió los mejores resultados.

De la figura 27 se puede desprender el siguiente análisis. El primero, y uno de los más importantes, hace referencia a la influencia en la clasificación de ir agregando de manera incremental imágenes sintéticas durante la etapa de entrenamiento. Se puede observar que a nivel general todos los experimentos llegaron a un punto donde su performance se desplomó al seguir agregando imágenes. Sin embargo, este cambio ocurrió en diferentes momentos dependiendo del caso. Por ejemplo, cuando trabajamos con grupos de dimensionalidad baja, 1 y 4 bits por grupo, la precisión del clasificador va en aumento hasta que se llega a un máximo correspondiente al haber entrenado al sistema con igual cantidad de imágenes reales y sintéticas. A partir de ese punto, el seguir incrementando la proporción de sintéticas sobre reales trajo efectos negativos en los resultados como se puede apreciar en el gráfico presentado. En los casos donde consideramos 10 bits por grupo el rendimiento disminuye cuando consideramos más de 75 imágenes sintéticas. Cuando superamos esa proporción se puede ver que la performance decae.

Todo esto se debe a que llegado cierto punto, el clasificador aprende las particularidades de los datos sintéticos. Es decir, a medida que la cantidad de imágenes sintéticas superan en gran proporción a las imágenes reales, la información almacenada sobre las primeras adquiere más relevancia en la clasificación haciendo que la performance disminuya.

La mejor curva en el gráfico la obtenemos cuando consideramos 4 bits por grupo y entrenamos al clasificador con la misma cantidad de imágenes sintéticas que reales. Este resultado coincide respecto al análisis realizado para los experimentos con imágenes reales donde consideramos que el mejor parámetro era considerar grupos de 4 bits. En la mayoría de los casos se dan buenos resultados cuando la cantidad de imágenes sintéticas sobre las reales es la misma, 1 y 4, o se duplica, 8 y 10.

Teniendo en cuenta los baselines presentados en la tabla 1 y el hecho de que estos experimentos tienen una base inicial de imágenes reales, se puede observar que el baseline "SYNTH" es superado. En cuanto al baseline "NATIVE", a diferencia de los resultados obtenidos con las imágenes reales, que se pueden ver en el inicio de cada curva de la Figura 27, el incremento en la precisión del clasificador no es muy notorio; sin embargo, el hecho de poder incrementar la performance haciendo uso de imágenes sintéticas es un avance. Llama la atención que el usar la misma cantidad de imágenes reales y sintéticas no haya producido un incremento más marcado en la performance. Sin embargo, como pudimos ver en 26, ni con 50 imágenes sintéticas por clase se podía alcanzar un valor similar al del baseline.

Otro de los análisis realizado en el trabajo pasa por los errores de clasificación que se han observado. En las figuras 30 y 31, se pueden ver dos matrices de confusión. La primera corresponde al mejor resultado obtenido en la Figura 27 y la segunda es una versión de la primera donde no diferenciamos caracteres mayúsculas de minúsculas.

En la región central, dada por la diagonal de la matriz, se encuentran las muestras que fueron bien clasificadas durante la evaluación del clasificador. Existen zonas dentro de la diagonal que corresponden a clases que obtuvieron una tasa de acierto inferior al 30 %, los cuadrados en escala de azul dentro de la imagen. En estas clases son más comunes los siguientes errores:

- No poder distinguir mayúsculas de minúsculas para los caracteres involucrados. En la figura 28 se puede observar un ejemplo donde, incluso para las personas, es difícil notar la diferencia.
- Confundir un carácter por otro similar. La figura 29 muestra varios ejemplos donde se pueden confundir caracteres.

(a) c minúscula (b) C mayúscula (c) z minúscula (d) Z mayúscula

(e) x minúscula (f) X mayúscula (g) v minúscula (h) V mayúscula

Figura 28: Error de clasificación donde se confunde el carácter en mayúscula y en minúscula.

(a) o minúscula (b) O mayúscula (c) número 0

(d) I mayúscula (e) l minúscula (f) número 1

Figura 29: Error de apariencia entre caracteres.

El no poder distinguir mayúsculas de minúsculas se ve reflejado en la matriz 31 en dos regiones. Dichas regiones corresponden a dos submatrices, la primera ubicada en la esquina inferior izquierda y la segunda en la esquina superior derecha. Ambas submatrices, no involucran los caracteres numéricos y se puede ver en la diagonal de cada una, la cantidad de imágenes de cada clase donde se confundió la versión mayúscula y minúscula. Es muy difícil, incluso para las personas, clasificar imágenes de caracteres recortados donde no hay una clara distinción entre la versión mayúscula y minúscula.

Los errores de "apariencia" como los expuesto en la figura 29 están distribuidos por toda la matriz, habiendo casos donde la similitud es evidente y en otros casos más raros donde no existe directamente. Este último caso, donde no existe similitud, puede solventarse si se entrenase al clasificador con más imágenes.

La matriz 31 se creó con el objetivo de reflejar otro punto de vista de los resultados, donde no se consideren los caracteres en mayúscula. Se puede observar que la diagonal de la matriz mejora con respecto a la anterior y además se acentúan aún más los problemas de apariencia que surgen entre ciertos caracteres. Por ejemplo, los errores de clasificación en el carácter "i" que se confunde con una "l" en muchos casos, o el carácter "o" con el número "0". La mayoría de los errores de clasificación se podrían corregir si las imágenes de los caracteres sintéticos pudieran almacenar el mismo nivel de información que una imagen real. Esta diferencia se pudo observar en los resultados pues, el conjunto de imágenes sintéticas en este trabajo no pudo superar en precisión al conjunto de imágenes reales.

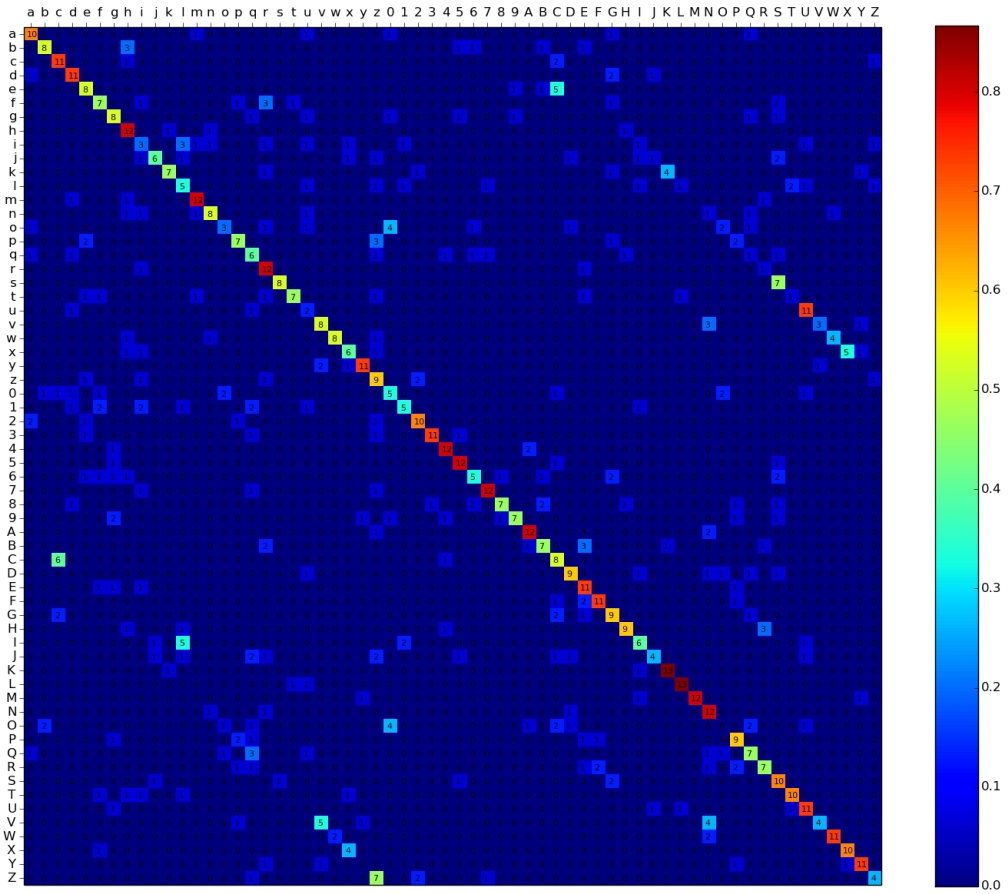


Figura 30: Matriz de correlación del gráfico 27 para el mejor resultado.

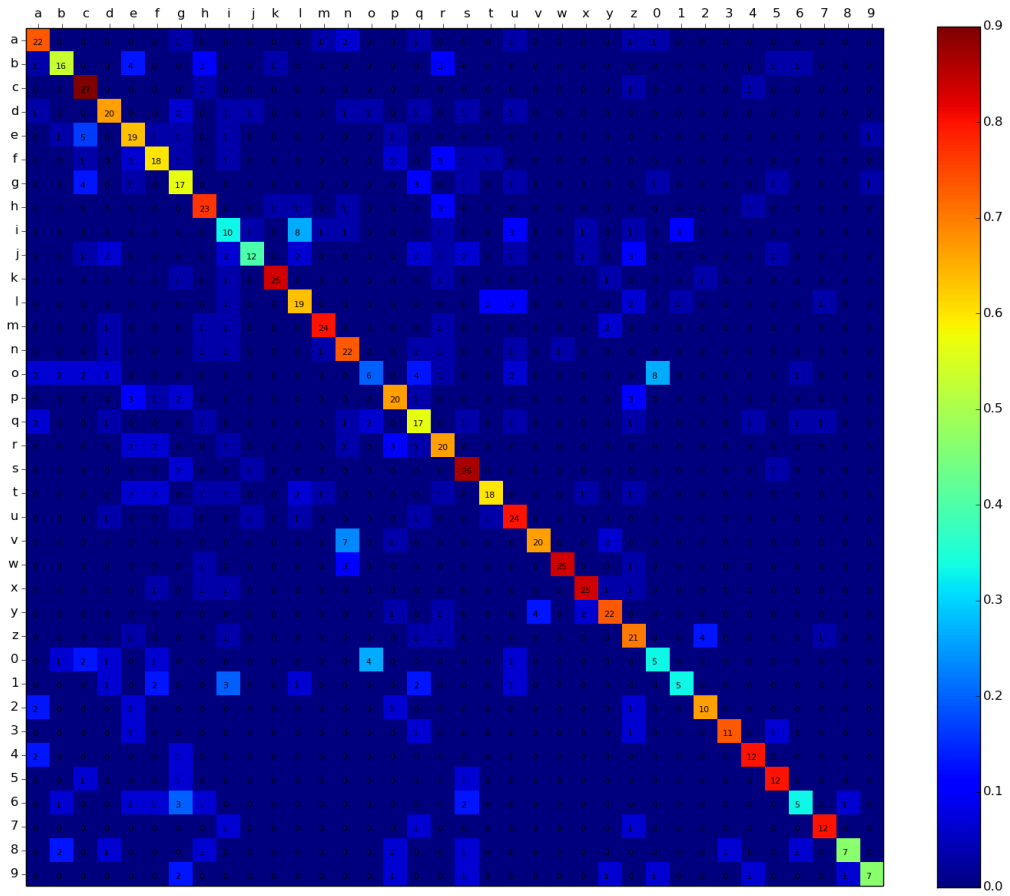


Figura 31: Matriz de correlación del gráfico 27 para el mejor resultado no teniendo en cuenta los caracteres en mayúscula.

5. Conclusiones y trabajos futuros

Para concluir con este trabajo y en base a los resultados expuestos, se puede destacar que el trabajo de reconocimiento de caracteres en imágenes estructuradas no es una tarea sencilla. Actualmente han habido bastantes avances en el campo pero todavía no se logra desarrollar un método que compita con los clasificadores que hacen reconocimiento de texto impreso.

Basados en los análisis del capítulo 4, se pudo observar que el uso de imágenes sintéticas proporcionó mejoras al momento de entrenar al clasificador si se las combina con imágenes reales. Si bien la mejora no es muy notoria, se puede seguir trabajando para mejorar esto y así poder evitar a futuro el tedioso trabajo de tener que recolectar imágenes reales. Una mejora interesante para agregar en la generación de imágenes sintéticas es tener un conjunto de fondos naturales para poder usar o darle color a los caracteres. Si bien todas las imágenes terminan en escala de grises, las pequeñas variaciones que se introducen al agregar estos detalles pueden ayudar a mejorar la clasificación.

El uso de Random Ferns como clasificador es una buena opción por lo expuesto en el capítulo 2; sin embargo, sería interesante probar con otros clasificadores para poder realizar una mejor comparación.

Sin duda alguna el desafío de reconocer texto en escenas naturales es un problema que vale la pena estudiar y pulir ya que las aplicaciones que tiene actualmente son enormes. Más aún con la proliferación de los dispositivos móviles y la necesidad de crear aplicaciones que asistan a usuarios con deficiencia visual entre otros.

Referencias

[1] Kai Wang, Boris Babenko and Serge Belongie, *"End-to-End Scene Text Recognition"*, IEEE International Conference on Computer Vision (ICCV), Barcelona, España, 2011.

[2] Kai Wang and Serge Belongie, *"Word spotting in the wild"*, In ECCV, 2010.

[3] Christopher M. Bishop. (2006). *"Pattern Recognition and Machine Learning, Information Science and Statistics"*. Springer.

[4] M. Mohri, A. Rostamizadeh, A. Talwalkar (2012). *"Foundations of Machine Learning"*. The MIT Press.

[5] T. E. de Campos, B. R. Babu, and M. Varma. Character recognition in natural images. In *"Proceedings of the International Conference on Computer Vision Theory and Applications, Lisbon, Portugal,"* February, 2009.

[6] Sunil Kumar, Rajat Gupta, Nitin Khanna, Santanu Chaudhury, and Shiv Dutt Joshi. *"Text extraction and document image segmentation using matched wavelets and mrf model." IEEE Transactions on Image Processing*, 16(8): 2117-2128, 2007.

[7] Navneet Dalal & Bill Triggs. *"Histograms of oriented gradients for human detection."* In Cordelia Schmid, Stefano Soatto, and Carlo Tomasi, editors, *International Conference on Computer Vision & Pattern Recognition*, volume 2, pages 886-893, INRIA Rhône-Alpes, ZIRST-655, av. de l'Europe, Montbonnot-38334, June 2005.

[8] ARH Inc. *Card Reader*, Computer Device. Obtenido de `http://www.passport-reader.com/card_reader.html`

[9] Google Inc. (2004). *Google Books*, Web Service. Obtenido de `http://books.google.com/`.

[10] T. S. Guzella, W. M. Caminhas. *"A review of machine learning approaches to Spam filtering"*. Expert Systems with Applications. 2009.

[11] Breiman, L. *"Random Forests"*. Machine Learning 45, (1), 5-32. January 2001.

[12] Donoser, M., Arth, C. & Bischof, H. *"Detecting, Tracking and Recognizing License Plates"*. Institute for Computer Graphic and Vision.

[13] J. Shotton, M. Johnson, and R. Cipolla. *"Semantic texton forests for image categorization and segmentation"*. In *CVPR*, 2008.

[14] M. Ozuysal, P. Fua, and V. Lepetit. *"Fast keypoint recognition in ten lines of code"*. In *CVPR*, 2007.

[15] P. Domingo. *"A Few Useful Things to Know about Machine Learning"*. Department of Computer Science and Engineering. University of Washington.

[16] L. Neumann and J. Matas. *"Real-Time Scene Text Localization and Recognition"*. In *CVPR*, 2012.

[17] Piotr Dollár. *"Piotr's Image and Video Matlab Toolbox (PMT)"*. Obtenido de `http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html`.

[18] L. Breiman. *"Bagging Predictors"*. Septiembre 1994.

[19] Quest Visual. *"Word Lens"*, 2014. Obtenido de `http://questvisual.com/`

[20] Optelec. *"Optelec ClearReader+"*, 2014. Obtenido de `http://us.optelec.com/products/crbaus-optelec-clearreader.html`

[21] A. Garg and D. Roth. (2001). *"Understanding Probabilistic Classifiers"*. In *EMCL*.

[22] K. P. Murphy. (2012). *"Machine Learning A Probabilistic Perspective"*. Cambridge: The MIT Press.

[23] d'Albe, E. E. F. (1 Julio 1914). *"On a Type-Reading Optophone"*. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 90 (619): 373–375.

[24] L. Rokach and O. Maimon (2010). *"Data Mining and Knowledge Discovery Handbook"*. 2nd ed. Springer Science.

[25] Rokach, L. and Maimon, O. (2005). *"Top-Down Induction of Decision Trees Classifiers – A Survey"*. IEEE Transactions on Systems, Man, and Cybernetics, Part C 35, (4): 476–487.

[26] O. D. Trier, A. K. Jain and T. Taxt. (1996). *"Feature Extraction methods for Character Recognition - A Survey"*. *Pattern Recognition*, Vol 29, No. 4, pp. 641-662, 1996.

[27] M. Narasimha Murty and V. Susheela Devi. (2011). *"Pattern Recognition: An Algorithmic Approach"* Springer. University Press.

[28] Ozuysal, M., Calonder, M., Lepetit, V. & Fua, P. *"Fast Keypoint Recognition using Random Ferns"*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32, 448-461, 2010.

[29] B. Gatos, I. Pratikakis and S. Perantonis. *"Towards Text Recognition in Natural Scene Images"*. Computational Intelligence Laboratory, Institute of Informatics and Telecommunications.

[30] H. Chen, S. S. Tsai, G. Schroth et. al. *"Robust text detection in natural images with edge-enhanced maximally stable extremal regions"*. In ICIP, 2011.

[31] J. R. Quinlan. *"Induction of Decision Trees"*. Machine Learning 1: 81-106. March 1986.

[32] J. R. Quinlan. *"C4.5: Programs for Machine Learning"*. Morgan Kaufmann Publishers, 1993.

[33] R. C. Gonzales and R. E. Woods. *"Digital Image Processing"*. Prentice Hall.

[34] D. Jacobs. (2005). *"Image Gradients"*.

[35] S. O. Hansson. (2005). *"Decision Theory: A Brief Introduction"*, Department of Philosophy and the History of Technology. Royal Institute of Technology (KTH). Stockholm.

A. Conceptos de probabilidad y notación

A continuación se introducirán algunos conceptos de probabilidad que serán de utilidad en el desarrollo de las secciones del capítulo 2. Los mismos, fueron extraidos de [22].

Variables aleatorias discretas

Dado un evento A, la expresión $p(A)$ denota la probabilidad de que A sea verdadero. Por ejemplo, A puede ser la expresión lógica "Va a llover mañana". Se requiere que $0 \leq p(A) \leq 1$, donde $p(A) = 0$ significa que el evento no va a ocurrir, y $p(A) = 1$ significa que el evento definitivamente va a suceder. Escribimos $p(\overline{A})$ para denotar la probabilidad del evento no A, es decir, de que el evento no ocurra; esto se define como $p(\overline{A}) = 1 - p(A)$. Se escribirá $A = 1$ para hacer referencia que el evento A es verdadero, y $A = 0$ cuando el evento A es falso.

Se puede extender la noción de eventos binarios definiendo una *variable aleatoria discreta* X, la cual puede tomar cualquier valor de un conjunto finito o infinito contable χ. Se denota la probabilidad de que $X = x$ por $p(X = x)$, o sólo $p(x)$ como abreviación. Ambas notaciones se van a usar indistintamente en el desarrollo del trabajo. Aquí $p : x \to \mathbb{R}$ se denomina *función de probabilidad*. Esta satisface las propiedades $0 \leq p(x) \leq 1$ y $\sum_{x \in \chi} p(x) = 1$.

Probabilidad de la unión de dos eventos

Dados dos eventos, A y B, se define la probabilidad de A o B de la siguiente manera:

$$p(A \vee B) = p(A) + p(B) - p(A \wedge B) \tag{A.0.1}$$

$$= p(A) + p(B) \text{ si A y B son mutuamente excluyentes} \tag{A.0.2}$$

Probabilidad conjunta

Se define la probabilidad del evento conjunto A y B como sigue:

$$p(A, B) = p(A \wedge B) \tag{A.0.3}$$

A esta ecuación se la llama *regla del producto*. Dada la distribución conjunta en dos eventos p(A,B), se define la "distribución marginal" de la

variable aleatoria A de la siguiente manera:

$$p(A) = \sum_b p(A, B = b) \tag{A.0.4}$$

donde se asumen todos los estados posibles de B. Se puede definir p(B) de forma similar. A esta ecuación se la denomina regla de la suma o regla de la probabilidad total.

Probabilidad condicional

Se define la probabilidad condicional del evento A, dado que el evento B es verdadero, como sigue:

$$p(A|B) = \frac{p(A, B)}{p(B)} \text{ si } p(B) > 0 \tag{A.0.5}$$

Regla de Bayes

Combinando la definición de probabilidad condicional con las reglas del producto y la suma se obtiene la *regla de Bayes*, también llamada *Teorema de Bayes*:

$$p(X = x|Y = y) = \frac{p(X = x, Y = y)}{p(Y = y)} \tag{A.0.6}$$

$$= \frac{p(X = x)p(Y = y|X = x)}{\sum_{x'} p(X = x')p(Y = y|X = x')} \tag{A.0.7}$$

Independencia e independencia condicional

Se dice que X e Y son independientes, denotado como $X \perp Y$, si se puede representar la unión como el producto de dos marginales, es decir,

$$X \perp Y \iff p(X, Y) = p(X)p(Y) \tag{A.0.8}$$

En general, se dice que un conjunto de variables es mutuamente independiente si la conjunta puede ser escrita como producto de marginales.

Desafortunadamente, la independencia es rara, porque la mayoría de las variables pueden influir en la mayoría de las otras variables. Sin embargo, usualmente esta influencia se da a través de otras variables en vez de ser directa. Por lo tanto se dice que X e Y son *condicionalmente independientes*,

dada Z, si y solo si la conjunta condicional puede ser escrita como producto de marginales condicionales:

$$X \perp Y|Z \iff p(X,Y|Z) = p(X|Z)p(Y|Z) \tag{A.0.9}$$

B. Gradiente

En este apéndice, se busca realizar una introducción básica al concepto de gradiente con el objetivo de poder comprender mejor como funcionan los descriptores HOG.

Sea $f(x_1, \ldots, x_n)$ una función escalar de múltiples variables. Como expresa Gonzales et. al. en [33], el gradiente de f es un vector que apunta en la dirección donde se registra la mayor tasa de incremento de la función. Su magnitud es la pendiente del gráfico en esa dirección. Es la generalización del concepto de derivada en funciones de múltiples variables.

El gradiente de la función f descrita anteriormente, es denotado como ∇f donde ∇, el símbolo nabla, denota el operador diferencial. El gradiente de f es definido como el único campo vectorial cuyo producto punto con cualquier vector v en cada punto x es la derivada direccional de f a lo largo de v. Es decir,

$$(\nabla f(x)) \cdot v = D_v f(x)$$

En un sistema de coordenadas rectangular, el gradiente es el campo vectorial cuyos componentes son las derivadas parciales de f:

$$\nabla f(x) = \frac{\partial f}{\partial x_1}\mathbf{e}_1 + \cdots + \frac{\partial f}{\partial x_n}\mathbf{e}_n$$

donde los $\mathbf{e}_i$ son vectores unitarios ortogonales que apuntan en la dirección de coordenadas.

En el procesamiento de imágenes, un gradiente es un cambio direccional en la intensidad o color de la imagen. En [34], Jacobs explica que el vector gradiente se forma combinando la derivada parcial de la imagen en las direcciones x e y. Se puede expresar de la siguiente forma:

$$\nabla I = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right) \tag{B.0.10}$$

donde I: $\mathbb{R}^2 \to [0,1]$, es la "función intensidad" que asigna un valor de intensidad a cada pixel, par (x,y), de la imagen. Según Jacobs, cuando

determinamos la derivada parcial de I respecto de x, determinamos la rapidez con que la imagen cambia de intensidad a medida que x cambia. Para funciones continuas, $I(x, y)$, podemos expresarlo de la siguiente manera:

$$\frac{\partial I(x,y)}{\partial x} = \lim_{\nabla x \to 0} \frac{I(x + \nabla x, y) - I(x, y)}{\nabla x} \tag{B.0.11}$$

El cálculo de los gradientes de una imagen es útil ya que sirve, por ejemplo, para realizar detección de bordes de un objeto. La detección de bordes busca identificar puntos en una imagen en donde el brillo de la misma cambie de manera abrupta o, más formalmente, tenga discontinuidades. El propósito de esto es capturar eventos importantes o cambios en las propiedades de una imagen. En este caso, después de que los gradientes han sido computados, los píxeles con alto valor de gradiente son elegido como posibles bordes. Los píxeles con el valor de gradiente más alto en la dirección del gradiente se convierten en píxeles de borde. Los gradientes, también pueden ser usados en aplicaciones que realizan reconocimiento de objetos o correspondencia de texturas.

C. Sistema Operativo, Hardware y Software

La mayor parte de los algoritmos fueron desarrollados en el Sistema Operativo *Ubuntu (Linux)*. Al haber elegido Python como lenguaje de programación permite utilizarlos en distintas plataformas, como *Mac* y *Windows*, sin problemas.

C.1. Datos Específicos sobre Hardware Utilizado

Computadora[2]:

- Procesador: Intel® Core™ i7-4700MQ CPU @ 2.40GHz × 8
- Memoria Ram: 16 GB 1333 Mhz DDR3
- Sistema Operativo: Ubuntu 14.04 LTS x64

Servidor "Ganesh"[3]:

- Micros: 4xDualCore AMD Opteron™ Processor 8212.

[2]Utilizada solamente para procesar los resultados obtenidos del servidor y generar los gráficos necesarios.

[3]Servidor proporcionado por la *FaMAF* para correr los experimentos.

- Placa madre: SuperMicro H8QM8.
- Memoria Ram: 32 GB DDR2.
- Sistema Operativo: Ubuntu 12.04.5 LTS x64.

C.2. Datos Específicos sobre Software Utilizado

- Python Versión 2.7.6
- Pillow Versión 2.4.0
- GCC Versión 4.8.2
- numpy Versión 1.8.1
- matplotlib Versión 1.3.1
- ipython Versión 2.1.0
- ipdb Versión 0.8
- git Versión 1.9.1
- scikit-image Versión 0.10.0
- scipy Versión 0.14.0

C.3. Herramientas de Software utilizadas

Durante la realización de la tesis, se usaron varias herramientas *de software libre* que fueron indispensables y de una utilidad crucial.

Algunas de estas herramientas fueron:

- SUBLIME TEXT: IDE que se utilizó para escribir todo el código Python del presente trabajo[4].
- TexMaker: IDE que se usó para escribir todo el código LaTeX de la tesis[5]. das
- GIT: Sistema de control de versiones usado para todo el código[6].

[4] http://www.sublimetext.com/
[5] http://www.xm1math.net/texmaker/
[6] http://git-scm.com/

Figura 32: SUBLIME-TEXT IDE.

Figura 33: Texmaker.

C.4. ¿Por qué Python?

El lenguaje que se ha elegido para programar al clasificador como así también todos los módulos de procesamiento de resultados es Python. Las razones por las cuales se elige este lenguaje y no otro son muchas, entre las cuales algunas de las más importantes son:

- Python tiene una comunidad muy activa y cooperativa: La comunidad de programadores que programan con Python es muy grande, y tiende a seguir creciendo, y se caracteriza fuertemente por una muy buena aceptación de nuevos miembros y una gran predisposición a la ayuda mutua en medios como foros y listas de correos, como la lista de correo de PyAr (Python Argentina). Esto muestra que Python es un lenguaje vivo y en crecimiento.
- Python es un lenguaje con una sintaxis sencilla y es fácil de entender y aprender.
- Python es rápido: Python cuenta con una enorme cantidad de librerías magníficas implementadas en C, como NumPy o SciPy, que permiten

Figura 34: GIT.

realizar operaciones complejas en cuanto a procesamiento de manera sumamente rápida y eficiente. Incluso si no existiera una librería en particular en C para la funcionalidad que se desea, se puede implementar la porción de código que realiza la función crítica en C y utilizarla desde Python.

- Python tiene un gran soporte para computación científica: con librerías sólidas, fuertemente mantenidas, bien documentadas y áltamente eficientes, como SciPy, NumPy, matplotlib, entre otros.
- Python es multiplataforma: el código Python de este trabajo podrá ser utilizado prácticamente en cualquier sistema operativo. Esto es una ventaja importantísima y no limita al usuario a un sistema operativo o hardware en particular.