PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

Comparación de métricas de distancia en el algoritmo K-Vecinos Más Cercanos para el problema de Reconocimiento Automático de Dígitos Manuscritos

Miguel Arriagada Rodríguez

INFORME DEL PROYECTO
PARA OPTAR AL TÍTULO PROFESIONAL
INGENIERO DE EJECUCIÓN EN INFORMÁTICA

Agosto 2015

PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

Comparación de métricas de distancia en el algoritmo K-Vecinos Más Cercanos para el problema de Reconocimiento Automático de Dígitos Manuscritos

Miguel Arriagada Rodríguez

Profesor Guía: Dr. Héctor Allende Cid
Profesor Co-referente: Rodrigo Alfaro

Agosto 2015

Dedicado a mi familia, quienes me brindaron su apoyo y cariño incondicional. A mis amigos, quienes me animaron a continuar cuando las noches se tornaban largas. Y a nuestros profesores, quienes nos entregaron las herramientas para formarnos como profesionales.

Índice

Resumen

El Reconocimiento de Patrones, es un área de la Inteligencia Computacional, que se encarga de encontrar patrones y regularidades en los datos, los cuales se obtienen de los objetos de estudio. Si bien la problemática no es nueva, aun en la actualidad presenta desafíos: la precisión en el reconocimiento de caracteres manuscritos es aún insuficiente, existiendo errores en el reconocimiento. Además el problema tiene aplicaciones de interés, tales como el reconocimiento automático de códigos postales, el reconocimiento de importes en cheques bancarios y procesamiento automático de formularios. Por esta razón, es tomado como referencia para la aplicación y testeo de nuevas teorías y algoritmos del área de Reconocimiento de Patrones en general.

En el presente informe se muestra el método de procesamiento Vecinos Más Cercanos (K Nearest Neighbors) para el reconocimiento de dígitos manuscritos, para lo cual se utilizará la base de datos MNIST, con la finalidad de realizar un estudio comparativo en base a medidas de desempeño de clasificación multi-clase. Para lo cual se utilizaran 4 métricas de distancia: Euclidiana, Chebyshev, Manhattan y Kullback Leibler; para evaluar cuál de estos obtiene un mejor desempeño de un conjunto de medidas de desempeño multi-clase. Este modelo será implementado en un nuevo lenguaje de programación llamado JULIA que posee una vasta librería matemática, las cuales facilitan la labor.

Palabras-claves: Reconocimiento de Patrones, Reconocimiento automático de dígitos manuscritos, base de datos MNIST, Vecinos más cercanos, Euclidiana, Chebyshev, Manhattan, Kullback Leibler, JULIA.

Abstract

Pattern Recognition is the study of how computational models can perceive the environment, learn how to distinguish patterns of interest from experience, and make reasonable decisions regarding the categories to which such patterns belong. Is a well-known problem, even today there are still challenges present: the accuracy in handwritten character recognition is still insufficient, there are errors in the recognition. In addition the problem has applications of interest, such as the automatic recognition of postal codes, recognition of amounts on cheques and automatic processing of forms. For this reason, it is taken as a reference for the implementation and testing new theories and algorithms of pattern recognition area in general.

In this report the method of nearest neighbor processing for recognition of handwritten digits, for which MNIST data base, in order to conduct a comparative study based on performance measures used multiclass classification is shown. For which four distance metrics were used: Euclidean, Chebyshev, Manhattan and Kullback Leibler; to assess which of these gets a better performance of a package of multi-class performance. This model will be implemented in a new programming language called JULIA that has extensive math library with which to work.

Key-words: Pattern Recognition, Automatic recognition of handwritten digits, MNIST data base, Nearest Neighbors, Euclidean, Chebyshev, Manhattan, Kullback Leibler, JULIA.

Listas de Figuras

Lista de Tablas

1 Introducción

El Reconocimiento de Patrones, es un área de la Inteligencia Computacional, que se encarga de encontrar patrones y regularidades en los datos, los cuales se obtienen de los objetos de estudio. Aunque las investigaciones en este campo llevan más de cincuenta años, el diseño de un reconocedor de patrones automático de propósito general cuyo desempeño sea perfecto permanece como una meta lejana. El mejor reconocedor de patrones conocido hasta ahora es el ser humano, no sabiéndose a ciencia cierta cuál es el proceso mediante el cual los humanos reconocemos patrones. Podríamos decir que un patrón es una entidad u objeto de estudio que posee regularidades en su representación [1][2], como por ejemplo, una imagen de huella digital, una palabra manuscrita, un rostro, una señal representando voz hablada, manuscrito de dígitos, entre otros muchísimos ejemplos posibles. Dado un patrón, la tarea de su reconocimiento o clasificación puede ser resuelta, en un principio, de dos maneras: de forma supervisada, en la cual el patrón será identificado como miembro de una clase predefinida, o de forma no supervisada, en la cual el patrón será asignado a una clase desconocida previamente y aprendida en base a la similitud entre patrones.

El área de Reconocimiento de Patrones representa un desafío en si misma dentro de la Inteligencia Artificial. Debido a esto, el interés por la misma se ha visto incrementado sobre todo por la demanda de aplicaciones computacionales relacionadas con diversas áreas como por ejemplo, Minería de Datos (*Data Mining*), clasificación automática de documentos, pronósticos financieros, organización y recuperación en bases de datos multimedia, Biometría (identificación de personas), herramientas para la toma de decisiones, por ejemplo, para diagnósticos médicos, entre muchas otras.

El Reconocimiento Óptico de Caracteres (OCR) es uno de los tópicos más antiguos dentro del Reconocimiento de Patrones. En sus comienzos, la problemática OCR parecía de fácil tratamiento y resolución. Sin embargo, el problema del reconocimiento de caracteres, aunque ha sido estudiado durante varios años obteniéndose una alta precisión en las respuestas, está lejos de considerarse resuelto: la precisión en el reconocimiento asociada tanto a caracteres impresos en una imagen degradada o a caracteres manuscritos, es aún insuficiente. Los métodos existentes basados en aprendizaje no funcionan bien sobre grandes conjuntos de datos categorizados, o sobre conjuntos que crecen, es decir, que van incorporando nuevas muestras; los errores en el reconocimiento aún existen.

El reconocimiento de caracteres puede dividirse en dos grandes áreas: la escritura en línea u *on-line* y la escritura fuera de línea u *off-line.* Los reconocedores *on-line* reciben datos a reconocer a medida que los usuarios escriben. En general, tienen que procesar y reconocer la escritura manuscrita en tiempo real o casi en tiempo real. Los reconocedores *off-line* actúan una vez que los datos han sido recolectados y en donde, por ejemplo, las imágenes de la escritura manuscrita se ingresan al sistema como mapas de bits. En estos sistemas la velocidad de procesamiento no depende de la velocidad de escritura del usuario, sino de las mismas especificaciones del sistema. Los reconocedores *on-line* están

ampliamente difundidos en computadores manuales, también denominada *Tablet*, donde no hay lugar para un teclado, y en cierta manera son más fáciles de construir que los *off-line*, ya que en la escritura en tiempo real se cuenta con una información importante como es el orden de los trazos.

En cambio, los reconocedores de escritura manuscrita *off-line* juegan un rol importante a partir del hecho de la existencia de grandes cantidades de papel escrito a procesar en nuestra sociedad. Un ejemplo típico de aplicación es el reconocimiento de códigos postales; inclusive bases de datos ampliamente utilizadas para testear sistemas de reconocimiento en el ámbito científico han surgido a partir de esta problemática, como por ejemplo, la base de datos MNIST.

El Reconocimiento de Dígitos Manuscritos es un tópico destacado dentro del Reconocimiento de Caracteres, que desde sus comienzos ha recibido una importante atención del área científica. Con la aparición de las nuevas tecnologías en el campo de las Ciencias de la Computación, y con la explosión de las aplicaciones relacionadas con la manipulación de datos vía Internet, la conversión automática de información escrita y hablada a formularios que puedan ser leídos por una computadora se ha convertido en una tarea de creciente importancia.

Para finalizar, el Reconocimiento de Dígitos Manuscritos constituye un problema modelo que incluye desafíos comunes con otros tópicos del área de Reconocimiento de Patrones. Por esta razón, es tomado como referencia para la aplicación y testeo de nuevas teorías y algoritmos del área de Reconocimiento de Patrones en general.

2 Definición de Objetivos

La investigación aplicada que se desarrollará, busca la comparación de algoritmos K- Nearest Neighbors con distintas métricas de distancia para el reconocimiento de dígitos manuscritos, utilizando la base de datos MNIST, en el lenguaje de programación de alto nivel, JULIA.

2.1 Objetivo general

El objetivo principal es la comparación de métricas de distancias en algoritmos K-NN para la clasificación automática de dígitos manuscritos en base a medidas de desempeño para clasificación, con la implementación en el lenguaje JULIA.

2.2 Objetivos específicos

La solución a desarrollar satisface los siguientes objetivos particulares, desprendidos del objetivo general, los cuales consisten en:

- Estudiar el estado del arte del problema a tratar.
- Implementar las metodologías para el reconocimiento de dígitos manuscritos.
- Implementación del modelo K-NN de reconocimiento de patrones en el lenguaje de alto nivel JULIA.
- Implementación de las distintas métricas de distancia.
- Definir la metodología experimental.
- Validación con base de datos de dígitos manuscritos MNIST.
- Evaluar los resultados obtenidos con la finalidad de validar las conclusiones.

3 Estado del Arte

La disponibilidad de técnicas de aprendizaje ha sido un factor fundamental para el desarrollo y éxito de ciertas aplicaciones de Reconocimiento de Patrones tales como el reconocimiento del discurso y el reconocimiento de la escritura manuscrita [3].

Los métodos utilizados en los comienzos de las investigaciones en OCR fueron *Template Matching y Análisis Estructural,* también conocidos como *Feature Matching.* Los templates o prototipos eran diseñados artificialmente y también seleccionados o promediados de las pocas muestras de datos. A medida que el número de muestras se incrementaba, estos métodos se volvían insuficientes para representar la variabilidad de las formas de las muestras, y por lo tanto no permitían construir un clasificador de alta precisión que no presente errores. Para poder aprovechar las ventajas de usar grandes conjuntos de datos, la comunidad científica dedicada al Reconocimiento de Caracteres orientó sus investigaciones a los métodos de clasificación basados en aprendizaje, especialmente las Redes Neuronales Artificiales (RNA) a finales de los 80 y durante la década de 1990. Debido a la estrecha relación entre las RNA y los métodos estadísticos de reconocimiento de patrones, estos últimos también fueron considerados ya que permitían mejorar los resultados obtenidos. Actualmente, los métodos de aprendizaje más nuevos, en particular las Maquinas de Soporte vectorial y en forma más general los métodos usados en *Kernel* (Ensamblado de Clasificadores), así como también los métodos basados en la combinación de múltiples clasificadores, son activamente estudiados y aplicados en el área de Reconocimiento de Patrones [4].

El aprendizaje mediante un conjunto de ejemplos, es una característica deseable en la mayoría de los sistemas del área. Los cuatro enfoques más importantes para el Reconocimiento de Patrones son: Template Matching, Clasificación Estadística, Correspondencia Sintáctica o Estructural y Redes Neuronales Artificiales [2] [5]. Estos enfoques no son necesariamente independientes y muchas veces, el mismo método puede ser visto con diferentes interpretaciones desde distintos enfoques. Enmarcados en estos cuatro modelos, por lo cual, se han presentado numerosas técnicas para el problema del reconocimiento de caracteres *off-line.*

El enfoque denominado Template Matching es uno de los más sencillos y está orientado a determinar el grado de similitud entre dos entidades del mismo tipo, que pueden ser puntos, curvas u otras formas. Para esto se debe disponer de un prototipo asociado con el patrón a reconocer (en general de dos dimensiones) y que este pueda aprender a partir de los datos de entrenamiento. Ademas las técnicas de Template Matching pueden agruparse en tres categorías: Correspondencia Directa, Prototipos Deformables y Correspondencia Elástica, y Correspondencia Relajada [6].

Por otra parte en el enfoque Estadístico, cada patrón está representado en términos de *d* características o mediciones, constituyendo un punto en el espacio de *d* dimensiones. El objetivo es seleccionar aquellas características que permitan que los patrones que

pertenezcan a distintas categorías ocupen regiones compactas y disjuntas en el espacio de características *d*-dimensional, de forma tal de poder separar los elementos de cada clase adecuadamente. Para esto, y en base a un conjunto de patrones de entrenamiento, se establecen límites de decisión en este espacio de características, por ejemplo, en base a las distribuciones de probabilidad de los patrones de cada clase [2]. Varias de las técnicas más utilizadas para el problema de la escritura manuscrita pertenecen a este grupo, como por ejemplo, Vecinos Más Cercanos (*K-Nearest-Neighbor*) [7], el clasificador Bayesiano [8], el clasificador discriminante polinomial [9], Modelos Markovianos Ocultos (*Hidden Markov Model* - HMM) [10] [11], Maquinas de soporte Vectorial (*Suport Vector Machines*) [12] [13], entre otras [5].

Por otro lado en las técnicas correspondientes al Reconocimiento de Patrones Sintáctico, se establece una analogía formal entre la estructura de los patrones y la sintaxis del lenguaje. Los patrones se consideran como estructuras u oraciones del lenguaje, mientras que las primitivas o subpatrones elementales constituyen el alfabeto, de forma tal que estas estructuras u oraciones son generadas de acuerdo a una gramática. Así, un conjunto de patrones complejos se puede describir utilizando un pequeño número de primitivas y reglas gramaticales. La gramática asociada a cada clase se infiere del conjunto de patrones de entrenamiento. El enfoque sintáctico presenta algunas dificultades como, por ejemplo, la necesidad de utilizar grandes conjuntos de datos y estar asociado a altos costos computacionales [14] [15].

Por ultimo, las Redes Neuronales Artificiales tienen la característica de poder aprender correspondencias no-lineales complejas entre valores de entrada y salida, entrenarse de forma automática mediantes ejemplos y poder aprender a partir de grandes bases de datos, presentando un muy buen rendimiento frente a datos con ruido. Las RNA han sido ampliamente utilizadas en el Reconocimiento de Patrones y en particular para el problema de Dígitos Manuscritos obteniéndose un alto rendimiento. El Perceptrón Multicapa [16] [17] entrenado con el algoritmo de *Backpropagation* [18] es uno de los modelos clásicos de RNA más estudiados y utilizados [19]. Otras arquitecturas exitosas son las Redes Convolucionales [20], Mapas Auto-organizados de Kohonen o *SOM* [21], *Radial Basis Functions* [22], *Time Delay Neural Networks* [23], entre otros [5].

Todos los enfoques mencionados tienen sus ventajas e inconvenientes. Con el objeto de mejorar los resultados en el reconocimiento aprovechando los beneficios de cada técnica, se han desarrollado distintas estrategias de combinación de clasificadores demostrándose experimentalmente que algunas de ellas realmente mejoran el rendimiento del mejor clasificador individual [24] [25] [5]. El uso de un módulo verificador que refina la elección de la clase de salida entre los mejores candidatos, es otra de las estrategias orientadas a incrementar el porcentaje de patrones correctamente clasificados [10] [26].

Para concluir, el Reconocimiento de Dígitos Manuscritos se enmarca dentro de OCR, y como ya hemos mencionado, es un campo de investigación en continuo desarrollo, debido no solo a las aplicaciones potenciales sino también a que las soluciones encontradas pueden aplicarse a otros problemas de Reconocimiento de Patrones.

3.1 Estructura de un Sistema Reconocedor de aprendizaje

El enfoque tradicional para la construcción de un sistema reconocedor de patrones consiste en dividir al sistema en dos módulos principales: un módulo encargado de la extracción de características y el otro dedicado a la clasificación, como se muestra en la figura.

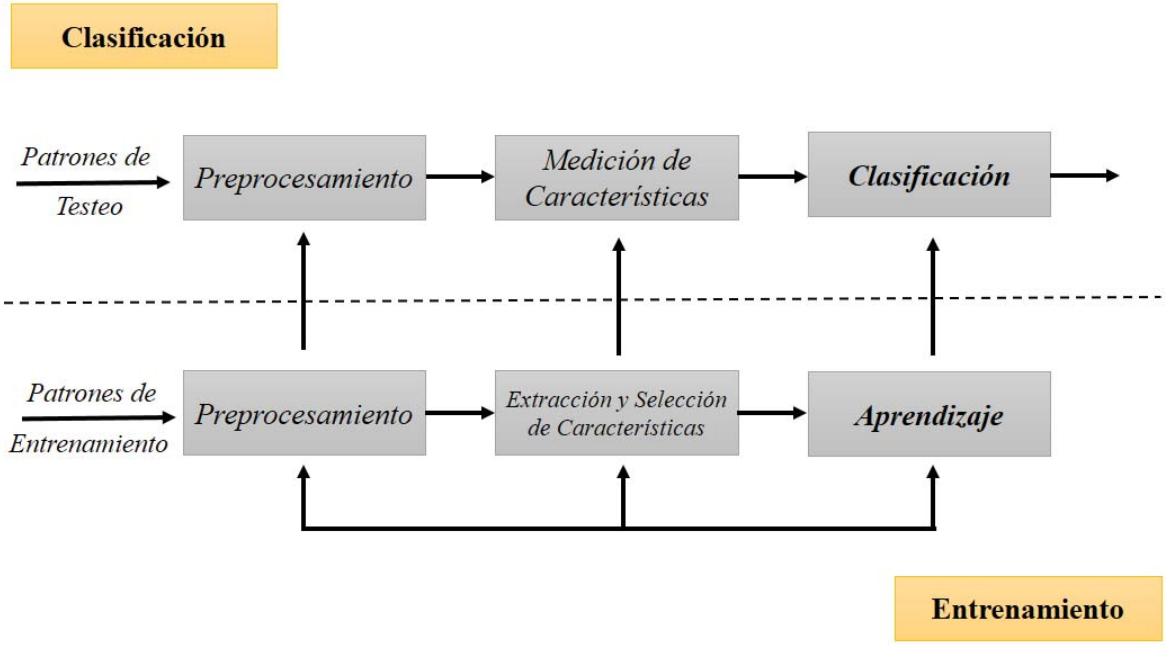

Figura 1: Etapas en la elaboración del modelo de aprendizaje [2].

Preprocesamiento: El preprocesamiento implica etapas de normalización y limpieza de los datos que serán utilizados como entrada para la siguiente etapa.

Extracción, selección y medición de características: En el entrenamiento, la extracción y selección de características permitirá entrenar el modelo con el conjunto de datos de entrenamiento. La selección realizada se basa en la experimentación con varias configuraciones hasta encontrar el menor error posible.

Aprendizaje y clasificación: Durante el entrenamiento, las características seleccionadas serán utilizadas para que el sistema aprenda e infiera un modelo que luego, durante la prueba, servirá para predecir a que clase (clasificar) pertenece cada elemento que se esté evaluando.

4 Definición del Modelo

4.1 Vecinos Más Cercanos

Los Vecinos Más Cercanos o K-NN (K Nearest Neighbors) es un método de clasificación supervisada (Aprendizaje, estimación basada en un conjunto de entrenamiento y prototipos) que sirve para estimar la función de densidad $F(x/\mathcal{C}_j)$ de las predictoras x por cada clase $\mathcal{C}_j$.

Este es un método de clasificación no paramétrico. Que estima el valor de la función de densidad de probabilidad o directamente la probabilidad a posteriori de que un elemento x pertenezca a la clase $\mathcal{C}_j$ a partir de la información proporcionada por el conjunto de prototipos. En el proceso de aprendizaje no se hace ninguna suposición acerca de la distribución de las variables predictoras.

En el reconocimiento de patrones, el algoritmo K-NN es usado como método de clasificación de objetos basados en un entrenamiento mediante ejemplos cercanos en el espacio de los elementos K-NN es un tipo de "Lazy Learning", donde la función se aproxima solo localmente y todo el computo es diferido a la clasificación.

Los ejemplos de entrenamiento son vectores en un espacio característico multidimensional, cada ejemplo esta descrito en términos de $\mathcal{P}$ atributos considerando q clases para la clasificación. Los valores de los atributos del i-esimo ejemplo (donde $1 \leq i \leq n$) se representan por el vector $\mathcal{P}$-dimensional $\mathcal{X}_i = (\mathcal{X}_{1i}, \mathcal{X}_{2i}, \dots, \mathcal{X}_{pi}) \in X$.

El espacio es particionado en regiones por localizaciones y etiquetas de los ejemplos de entrenamiento. Un punto en el espacio es asignado a la clase $\mathcal{C}$ si esta es la clase más frecuente entre los k ejemplos de entrenamiento más cercano. Generalmente se usa la distancia euclidiana.

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{p} (x_{ri} - x_{rj})^2}$$

La fase de entrenamiento del algoritmo consiste en almacenar los vectores característicos y las etiquetas de las clases de los ejemplos de entrenamiento. En la fase de clasificación, la evaluación del ejemplo (del que no se conoce su clase) es representada por un vector en el espacio característico. Se calcula la distancia entre los vectores almacenados y el nuevo vector, y se seleccionan los K ejemplos más cercanos. El nuevo ejemplo es clasificado con la clase que más se repite en los vectores seleccionados.

Este método supone que los vecinos más cercanos nos dan la mejor clasificación y esto se hace utilizando todos los atributos; el problema de dicha suposición es que es posible que

se tengan muchos atributos irrelevantes que dominen sobre la clasificación: dos atributos relevantes perderían peso entre un conjunto de atributos irrelevantes.

Para corregir el posible sesgo se puede asignar un peso a las distancias de cada atributo, dándole así mayor importancia a los atributos más relevantes. Otra posibilidad consiste en tratar de determinar o ajustar los pesos con ejemplos conocidos de entrenamiento. Finalmente, antes de asignar pesos es recomendable identificar y eliminar los atributos que se consideran irrelevantes [29].

En síntesis, el método K-NN se resumen en dos algoritmos:

4.1.1 Algoritmo de entrenamiento

Para cada ejemplo $< x, f(x) >$ donde $x \in X$, agregar el ejemplo a la estructura representando los ejemplos de aprendizaje.

4.1.2 Algoritmo de clasificación

Dado un ejemplar $\mathcal{X}_q$ que debe ser clasificado, sean $\mathcal{X}_1, \ldots, \mathcal{X}_k$ los K vecinos más cercanos a $\mathcal{X}_q$ en los ejemplos de aprendizaje, regresar

$$\hat{f}(x) \leftarrow argmax_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i))$$

Donde

$\delta(a, b) = 1 \; si \; a = b; y \; 0$ en cualquier otro caso.

El valor $\hat{f}(\mathcal{X}_q)$ devuelto por el algoritmo como un estimador de $f(\mathcal{X}_q)$ es solo el valor más común de f entre los K vecinos más cercanos a $\mathcal{X}_q$. Si elegimos $K = 1$; entonces el vecino más cercano a $\mathcal{X}_i$ determina su valor.

4.1.3 Elección del K

La mejor elección de K depende fundamentalmente de los datos; generalmente, valores grandes de K reducen el efecto de ruido en la clasificación, pero crean límites entre clases parecidas. Un buen K puede ser seleccionado mediante una optimización de uso. El caso especial en que la clase es predicha para ser la clase más cercana al ejemplo de entrenamiento (cuando $K = 1$) es llamada Nearest Neighbor Algorithm, Algoritmo del vecino más cercano.

El desempeño de este algoritmo puede ser severamente degradada por la presencia de ruido o características irrelevantes, o si las escalas de características no son consistentes con lo que uno considera importante.

4.2 Métricas de Distancias

A continuación definiremos las 4 Métricas de Distancias que se utilizaran para testear el modelo K-NN:

4.2.1 Distancia Euclidiana

Es la distancia en línea recta o la trayectoria más corta posible entre dos puntos:

$$\mathcal{D}_{Euc}(x_i, x_j) = \sqrt{\sum_{r=1}^{p} (x_{ri} - x_{rj})^2}$$

4.2.2 Distancia Chebyshev

Es una métrica definida en un espacio vectorial donde la distancia entre dos vectores es el mayor de sus diferencias a lo largo de cualquier dimensión de coordenadas:

$$\mathcal{D}_{Cheb}(i, j) = \max_k \left| \mathcal{X}_{ik} - \mathcal{X}_{jk} \right|$$

4.2.3 Distancia Manhattan

La función de la distancia Manhattan calcula la distancia que se puede recorrer para llegar de un punto de datos a la otra si un camino en forma de rejilla es seguido. La distancia Manhattan entre dos elementos es la suma de las diferencias de sus correspondientes componentes:

$$\mathcal{D}_{Man}(\mathcal{X}, Y) = \sum_{i=1}^{k} |\mathcal{X}_i - Y_i|$$

4.2.4 Distancia Kullback-Leibler

Es una medida no simétrica de la similitud o diferencia entre dos funciones de distribución de probabilidad P y Q. KL mide el número esperado de extra bits requeridos en muestras de código de P cuando se usa un código basado en Q, en lugar de un código basado en P. Generalmente P representa la "verdadera" distribución de los datos, observaciones, o cualquier distribución teórica. La medida Q generalmente representa una teoría, modelo, descripción o aproximación de P:

$$\mathcal{D}_{Kullback}(p, q) = \sum_{i=1}^{n} p(x_i) \log \frac{p(x_i)}{q(x_i)}$$

5 Validación del Estudio

Los algoritmos de Máquinas de Aprendizaje o Machine Learning (ML) dividen su clasificación en tareas binarias, multi-clase, multi-etiquetada y jerárquica. Para ello consideraremos una serie de indicadores derivados de la matriz de confusión que corresponden a las características específicas de los datos.

Las Maquinas de Aprendizaje supervisadas permite el acceso a las etiquetas de los datos durante las etapas de entrenamiento. Consideremos etiquetas categóricas cuando los datos de entrada son $\mathcal{X}_1, \ldots, \mathcal{X}_n$ tienen que ser asignados en clases predefinidas $\mathcal{C}_1, \ldots, \mathcal{C}_\ell$. Para nuestro estudio utilizaremos la clasificación de Multi-Clase.

Multi-Clase: La entrada debe ser clasificada en uno y solo uno, las ℓ clases no se superponen. Los problemas Multi-Clase incluyen la identificación de dígitos manuscritos en un conjunto de datos, en el reconocimiento de patrones [30]. En cuanto al caso binario, la categorización Multi-Clase puede ser objetiva o subjetiva, definida o ambigua.

5.1 Medidas de desempeño para la clasificación

La exactitud de una clasificación puede ser evaluado calculando el número de ejemplos de la clase correctamente reconocidos (true positives), el número de ejemplos correctamente reconocidos que no pertenecen a la clase (true negatives), los ejemplos que fueron incorrectamente asignados a la clase (false positives) y los que no fueron reconocidos como ejemplos de la clase (false negatives). Estos cuatro cargos constituyen una matriz de confusión.

Clases de Datos	Clasificado como *Positivo*	Clasificado como *Negativo*
Positivos	*Verdadero Positivo* (tp)	*Falso Negativo* (fn)
Negativos	*Falso Positivo* (fp)	*Verdadero Negativo* (tn)

$$\begin{bmatrix} tp & fn \\ fp & tn \end{bmatrix}$$

Figura 2: Matriz de Confusión [34].

5.1.1 Medidas de desempeño para la clasificación Multi-Clase

Para una clase individual de $\mathcal{C}_i$, la evaluación es definida por tp_i, fn_i, tn_i, fp_i, $Accuracy_i$, $Precision_i$, $Recall_i$ y *Error Rate* son calculados a partir del valor para $\mathcal{C}_i$. [31]

Accuracy: El promedio por clase eficacia de un clasificador.

$$\frac{\sum_{i=1}^{\ell} \frac{tp_i + tn_i}{tp_i + fp_i + fn_i + tn_i}}{\ell}$$

Error Rate: El error de clasificación por clase media.

$$\frac{\sum_{i=1}^{\ell} \frac{fp_i + fn_i}{tp_i + fn_i + fp_i + tn_i}}{\ell}$$

Precision: Acuerdo de las etiquetas de la clase de datos con los de un clasificador es calculada a partir de las sumas.

$$\frac{\sum_{i=1}^{\ell} tp_i}{\sum_{i=1}^{\ell} (tp_i + fp_i)}$$

Recall: Eficacia de un clasificador para identificar las etiquetas de clase si se calcula a partir de las sumas.

$$\frac{\sum_{i=1}^{\ell} tp_i}{\sum_{i=1}^{\ell} (tp_i + fn_i)}$$

5.2 Corridas experimentales

Se realizara el experimento 20 veces por cada configuración, para así poder obtener la media y la varianza de las medidas de desempeño. De los resultados obtendremos los primeros 2 momentos, y asumiremos normalidad. Con ello se podrá realizar un Test de Hipótesis con el fin que los resultados tengan validez estadística.

5.2.1 Test de Hipótesis

Contrastar una Hipótesis Estadísticamente es juzgar si cierta propiedad supuesta para una población es compatible con lo observado en una muestra de ella [37].

Hipótesis Nula: ($\mathbf{H_0}$) es la hipótesis que se contrasta. Esta hipótesis se mantendrá a no ser que los datos indiquen lo contrario. Esta hipótesis nunca se considera probada aunque puede ser rechazada por los datos.

Hipotesis Alternativa: ($\mathbf{H_1}$) es la hipótesis contrapuesta a H_0.

Elementos de un Test de Hipótesis:

1. **Hipótesis Nula (H_0) e Hipótesis Alternativa (H_1).**
2. **Estadística de Prueba (Discrepancia):** Es la función de la muestra. Interesa que contenga el máximo de información sobre H_0. En base a la información contenida en esta función que decidiremos respecto de la aceptación o rechazo de H_0.
3. **Región de Rechazo (Región Crítica):** Define los valores del estadístico de Prueba para los cuales se contradice H_0.
4. **Regla de Decisión:** Procedimiento que acepta o rechaza H_0, dependiendo del valor del estadístico de Prueba.

6 Problema de Reconocimiento de Dígitos Manuscritos

Las imágenes como se mencionó anteriormente provienen de la base de datos MNIST [28], la cual contiene 42.000 ejemplos de números manuscritos con 785 columnas, la primera columna corresponde a la etiqueta la cual indica el numero al cual corresponde, el resto de las columnas representan el numero en una escala de pixeles entre 0 y 255.

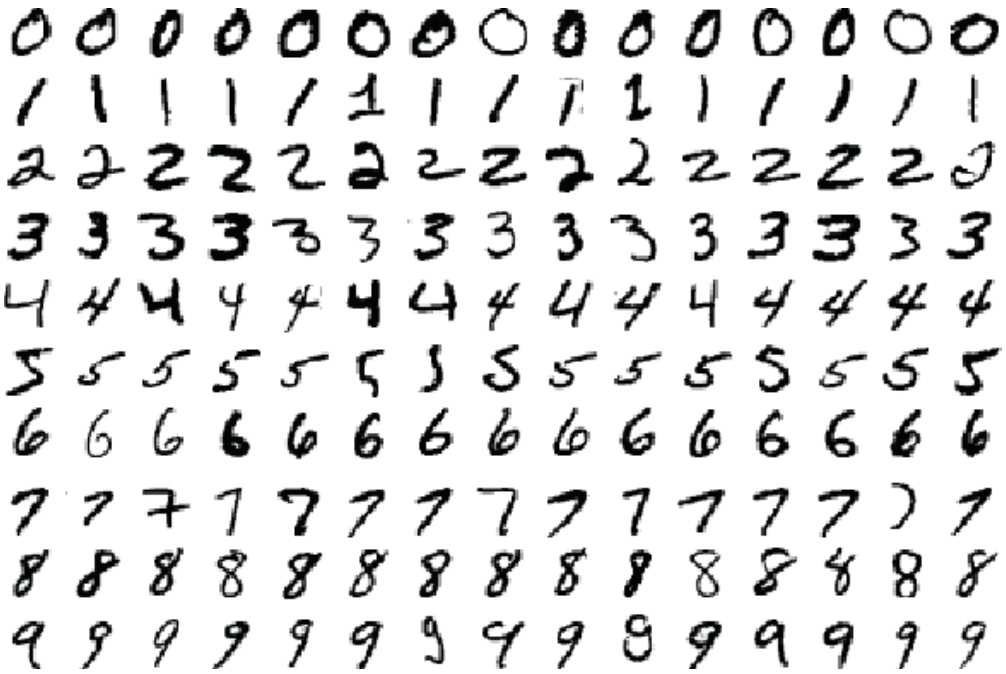

Figura 3: Muestra de la base de datos MINIST [28].

Para poder trabajarla lo primero es hacerle una permutación de las filas con la finalidad de conseguir más variabilidad entre las pruebas para luego separarla en una parte de testeo y otra parte de prueba. Luego se separa la primera columna (etiqueta) del resto (numero) para poder trabajarla.

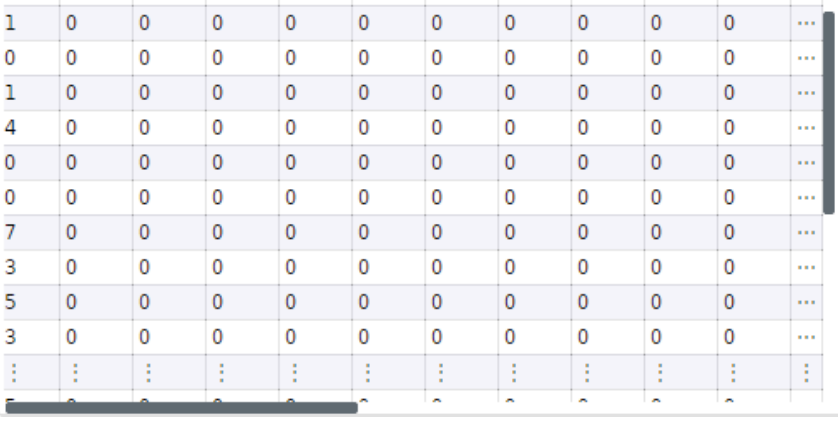

Figura 4: Matriz a Trabajar en Julia [27].

Una vez aplicado el algoritmo K-NN se obtienen dos vectores, uno correspondiente a las etiquetas de test y el otro corresponde a las etiquetas que genera el algoritmo K-NN, con estos vectores se procede a calcular las Medidas de Desempeño a través de la Matriz de Confusión.

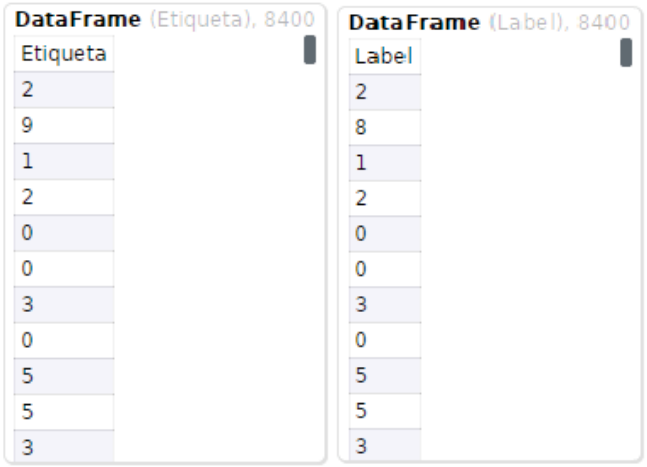

Figura 5: Ejemplo de Vectores correspondientes a las etiquetas.

Con la matriz de confusión se procede al cálculo de las medidas de desempeño: Accuracy, Error Rate, Precision y Recall:

	A	B	C	D	E	F	G	H	I	J
A	tp_A	e_{AB}	e_{AC}	e_{AD}	e_{AE}	e_{AF}	e_{AG}	e_{AH}	e_{AI}	e_{AJ}
B	e_{BA}	tp_B	e_{BC}	e_{BD}	e_{BE}	e_{BF}	e_{BG}	e_{BH}	e_{BI}	e_{BJ}
C	e_{CA}	e_{CB}	tp_C	e_{CD}	e_{CE}	e_{CF}	e_{CG}	e_{CH}	e_{CI}	e_{CJ}
D	e_{DA}	e_{DB}	e_{DC}	tp_D	e_{DE}	e_{DF}	e_{DG}	e_{DH}	e_{DI}	e_{DJ}
E	e_{EA}	e_{EB}	e_{EC}	e_{ED}	tp_E	e_{EF}	e_{EG}	e_{EH}	e_{EI}	e_{EJ}
F	e_{FA}	e_{FB}	e_{FC}	e_{FD}	e_{FE}	tp_F	e_{FG}	e_{FH}	e_{FI}	e_{FJ}
G	e_{GA}	e_{GB}	e_{GC}	e_{GD}	e_{GE}	e_{GF}	tp_G	e_{GH}	e_{GI}	e_{GJ}
H	e_{HA}	e_{HB}	e_{HC}	e_{HD}	e_{HE}	e_{HF}	e_{HG}	tp_H	e_{HI}	e_{HJ}
I	e_{IA}	e_{IB}	e_{IC}	e_{ID}	e_{IE}	e_{IF}	e_{IG}	e_{IH}	tp_I	e_{IJ}
J	e_{JA}	e_{JB}	e_{JC}	e_{JD}	e_{JE}	e_{JF}	e_{JG}	e_{JH}	e_{JI}	tp_J

Tabla 1: Matriz de Confusión para el cálculo de Medidas de Desempeño [35].

Accuracy: Es la suma de las clasificaciones correctas dividida por el número total de clasificaciones:

$$accuracy = \frac{\sum \mathrm{tp}}{\sum \mathrm{tp} + \sum \mathrm{e}}$$

En Julia: accuracy = trace(MC) / sum(MC)

Error Rate: Es la suma de las clasificaciones incorrectas o errores, dividida por el número total de clasificaciones:

$$Error = \frac{\sum \mathrm{e}}{\sum \mathrm{tp} + \sum \mathrm{e}}$$

En Julia: Error_Rate = (sum(MC) – trace(MC)) / sum(MC)

Precisión: Es una medida de la Accuracy siempre que una clase específica ha sido predicha, esta se define por:

$$Precision = \frac{\mathrm{tp}}{(\mathrm{tp} + \mathrm{fp})}$$

En la matriz de confusión, por ejemplo la Precisión para la clase A se calcula como:

$$Prec_{\mathrm{A}} = \mathrm{tp_A}/(\mathrm{tp_A} + \mathrm{e_{BA}} + \mathrm{e_{CA}} + \mathrm{e_{DA}} + \mathrm{e_{EA}} + \mathrm{e_{FA}} + \mathrm{e_{GA}} + \mathrm{e_{HA}} + \mathrm{e_{IA}} + \mathrm{e_{JA}})$$

En Julia: puede ver código en Anexo.

Recall: Es una medida de la capacidad del modelo de predicción para seleccionar instancias de una clase determinada de un conjunto de datos, y corresponde a la tasa de true positive, se define por:

$$Recall = \frac{\mathrm{tp}}{(\mathrm{tp} + \mathrm{fn})}$$

En la matriz de confusión, por ejemplo el Recall para la clase A se calcula como:

$$Recall_{\mathrm{A}} = \mathrm{tp_A}/(\mathrm{tp_A} + \mathrm{e_{AB}} + \mathrm{e_{AC}} + \mathrm{e_{AD}} + \mathrm{e_{AE}} + \mathrm{e_{AF}} + \mathrm{e_{AG}} + \mathrm{e_{AH}} + \mathrm{e_{AI}} + \mathrm{e_{AJ}})$$

En Julia: puede ver codigo en Anexo.

F-Score: Es la medida de precisión que tiene una prueba. Para ello se consideran tanto la Precisión como el Recall de una prueba para calcular su score. En este estudio se considerara el uso de **F1-Score** [36].

El F1-Score puede ser interpretado como el promedio ponderado de la Precisión y el Recall, en el cual el F1-Score alcanza su mejor valor en 1 y su peor valor en 0, el cual se define por:

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

7 Algoritmo de K-NN

A continuación se presentara el Algoritmo o código de K-NN implementados en el Lenguaje de Programación Julia [27]:

Ver código yPredictionseuc en Anexo:

Este ejemplo es utilizando la métrica de distancia Euclidiana, el cual busca generar la yPredictionseuc que corresponde a las etiquetas que predice el modelo K-NN utilizando la data que previamente fue separada en xTrain, yTrain, xTest. Con esto invoca la función assign_label para el uso del modelo.

Ver código assign_label en Anexo:

Esta función primeramente invoca a la función get_k_nearest_neighbors para obtener el valor que se obtendrá de acuerdo a cada métrica de distancia, para luego asignar la etiqueta con el valor (o número) más popular de acuerdo al valor obtenido con la función anterior, de esta manera el modelo asigna el valor de la etiqueta del vecino más cercano.

Ver código get_k_nearest_neighbors en Anexo:

Esta función calcula la distancia según la métrica que se utilice (con el valor recibido dist), este calcula la distancia con todos los vectores de entrenamiento con el vector de test para poder obtener el valor de la distancia entre los vectores para posteriormente comparar el más popular y así asignar la etiqueta que el modelo crea correspondiente.

8 Análisis de Resultados

En esta sección se presentaran los resultados obtenidos tras las 20 corridas experimentales realizadas con las diferentes métricas de distancias y con la variación del K, para ello se dividió la base de datos MINIST en un 80% para el entrenamiento del modelo y en 20% para el testeo del modelo, por lo cual a continuación se presentara primeramente las Medidas de Desempeño para posteriormente realizar un Test de Hipótesis, para los resultados del test.

8.1 Resultados Obtenidos

Como se mencionó anteriormente se realizaron 20 corridas experimentales para cada métrica de distancia y para cada valor de K, de estos se obtuvieron las medias de dichos resultados que compraremos a continuación:

8.1.1 Métricas de Distancias

A continuación se presentaran las diferentes tablas de las métricas de distancias utilizadas, correspondientes a D. Euclidiana, D. Chebyshev, D. Manhattan, D. Kullback Leibler:

D. Euclidiana	Accuracy	Error Rate	Precision	Recall	F1-Score
K=2	0,9672	0,0328	0,9668	0,9675	0,9671
K=3	0,9682	0,0318	0,9677	0,9685	0,9681
K=4	**0,9684**	**0,0316**	**0,9680**	**0,9689**	**0,9684**
K=5	0,9676	0,0324	0,9671	0,9681	0,9676
K=6	0,9678	0,0322	0,9674	0,9684	0,9679
K=7	0,9666	0,0334	0,9662	0,9673	0,9667
K=8	0,9655	0,0345	0,9650	0,9663	0,9657
K=9	0,9664	0,0336	0,9659	0,9671	0,9665
K=10	0,9637	0,0363	0,9632	0,9646	0,9639
K=15	0,9604	0,0396	0,9599	0,9616	0,9607
K=20	0,9565	0,0435	0,9559	0,9580	0,9570
K=25	0,9538	0,0462	0,9531	0,9556	0,9543
K=30	0,9507	0,0493	0,9500	0,9529	0,9514

Tabla 2: Métrica de Distancia Euclidiana, Medias de acuerdo al K.

Como podemos observar el mejor resultado usando la métrica de distancia Euclidiana se obtuvo con el valor de K=4 alcanzando un 96.8% de Accuracy, un Error Rate de 3.16%, una Precision de 96.8%, un Recall de 96.89% y un F1-Score de 96.8%.

Cabe destacar que esta métrica de distancia obtuvo los mejores resultados de la experimentación alcanzando el mayor número de aciertos positivos en el reconocimiento de dígitos manuscritos.

D. Chebyshev	Accuracy	Error Rate	Precision	Recall	F1-Score
K=2	**0,7935**	**0,2065**	**0,7897**	**0,8023**	**0,7960**
K=3	0,7910	0,2090	0,7870	0,8033	0,7951
K=4	0,7893	0,2107	0,7857	0,8004	0,7930
K=5	0,7850	0,2150	0,7808	0,7940	0,7874
K=6	0,7832	0,2168	0,7787	0,7946	0,7866
K=7	0,7820	0,2180	0,7781	0,7934	0,7856
K=8	0,7795	0,2205	0,7747	0,7912	0,7829
K=9	0,7822	0,2178	0,7770	0,7949	0,7859
K=10	0,7751	0,2249	0,7702	0,7878	0,7789
K=15	0,7667	0,2333	0,7621	0,7820	0,7720
K=20	0,7609	0,2391	0,7562	0,7750	0,7655
K=25	0,7586	0,2414	0,7528	0,7711	0,7619
K=30	0,7512	0,2488	0,7452	0,7657	0,7553

Tabla 3: Métrica de Distancia Chebyshev, Medias de acuerdo al K.

Con la métrica de distancia Chebyshev se alcanzó su mejor resultado con K=2 obteniéndose una Accuracy de 79.35%, un Error Rate de 20.65%, una Precision del 78.9%, Recall de 80.2% y un F1-Score del 79.6%, lo cual no la convierte en la mejor opción a la hora de evaluar modelos de reconocimiento de dígitos manuscritos.

D. Manhattan	Accuracy	Error Rate	Precision	Recall	F1-Score
K=2	0,9611	0,0389	0,9605	0,9617	0,9611
K=3	**0,9621**	**0,0379**	**0,9614**	**0,9628**	**0,9621**
K=4	0,9618	0,0382	0,9613	0,9627	0,9620
K=5	0,9608	0,0392	0,9601	0,9618	0,9610
K=6	0,9614	0,0386	0,9608	0,9625	0,9617
K=7	0,9595	0,0408	0,9587	0,9606	0,9596
K=8	0,9584	0,0416	0,9577	0,9600	0,9589
K=9	0,9593	0,0407	0,9587	0,9608	0,9598
K=10	0,9564	0,0436	0,9557	0,9581	0,9569
K=15	0,9517	0,0483	0,9511	0,9542	0,9526
K=20	0,9477	0,0523	0,9469	0,9506	0,9487
K=25	0,9447	0,0553	0,9437	0,9481	0,9459
K=30	0,9421	0,0579	0,9411	0,9459	0,9435

Tabla 4: Métrica de Distancia Manhattan, Medias de acuerdo al K.

Como podemos observar con la métrica de distancia Manhattan se obtuvo su mejor resultado con K=3 alcanzando un Accuracy de 96.2%, un Error Rate de 3.8%, una Precision del 96.14%, un Recall del 96.2% y un F1-Score del 96.2%.

Cabe destacar que esta métrica de distancia fue la segunda mejor, con una diferencia de menos 1% con la D. Euclidiana, obteniéndose valores muy similares entre ambas, la cual la convierte en una buena opción de distancia para el uso de modelos de reconocimiento de dígitos manuscritos.

D. Kullback Leibler	Accuracy	Error Rate	Precision	Recall	F1-Score
K=2	**0,2869**	**0,7131**	**0,2807**	**0,7241**	**0,4046**
K=3	0,2805	0,7195	0,2744	0,7261	0,3983
K=4	0,2731	0,7269	0,2676	0,7286	0,3915
K=5	0,2652	0,7348	0,2590	0,7376	0,3834
K=6	0,2520	0,7480	0,2454	0,7310	0,3674
K=7	0,2491	0,7509	0,2438	0,7352	0,3662
K=8	0,2372	0,7628	0,2297	0,7479	0,3515
K=9	0,2543	0,7457	0,2465	0,7385	0,3696
K=10	0,2247	0,7753	0,2178	0,7445	0,3370
K=15	0,2231	0,7769	0,2160	0,7508	0,3354
K=20	0,2082	0,7918	0,2030	0,7679	0,3211
K=25	0,2095	0,7905	0,2029	0,7442	0,3189
K=30	0,2026	0,7974	0,1975	0,7367	0,3115

Tabla 5: Métrica de Distancia Euclidiana, Medias de acuerdo al K.

Con esta métrica de distancia se obtuvo su mejor resultado con K=2 obteniéndose un Accuracy de 28.69%, un Error Rate de 71.31%, una Precision del 28.07%, un Recall de 72.41% y un F1-Score del 40.46%, siendo esta métrica con la que se obtuvieron los peores resultados de la experimentación, convirtiéndose en una mala opción a la hora de evaluar modelos de reconocimiento de dígitos manuscritos.

Para concluir se pudo observar que las métricas de distancia D. Euclidiana y D. Manhattan se obtuvieron los mejores valores siendo estos muy similares, cabe también destacar que con todas las métricas de distancias se obtuvieron los mejores resultados de cada una de estas con el valor de K entre 2 y 4.

8.1.2 Medidas de Desempeño

A continuación se presentaran gráficos comparativos para cada medida de desempeño:

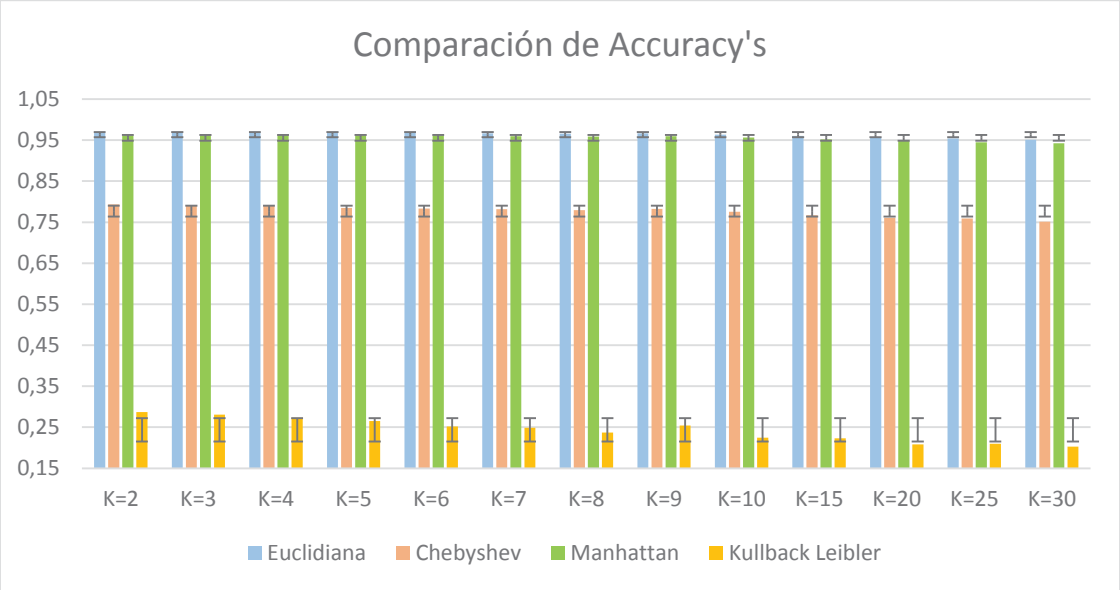

Figura 6: Gráfico Comparación de Accuracy.

Como se puede observar las mejores Accuracy's las obtienen las distancias Euclidiana y Manhattan, siendo la distancia Euclidiana un poco mejor esto queda reflejado en el aumento del K.

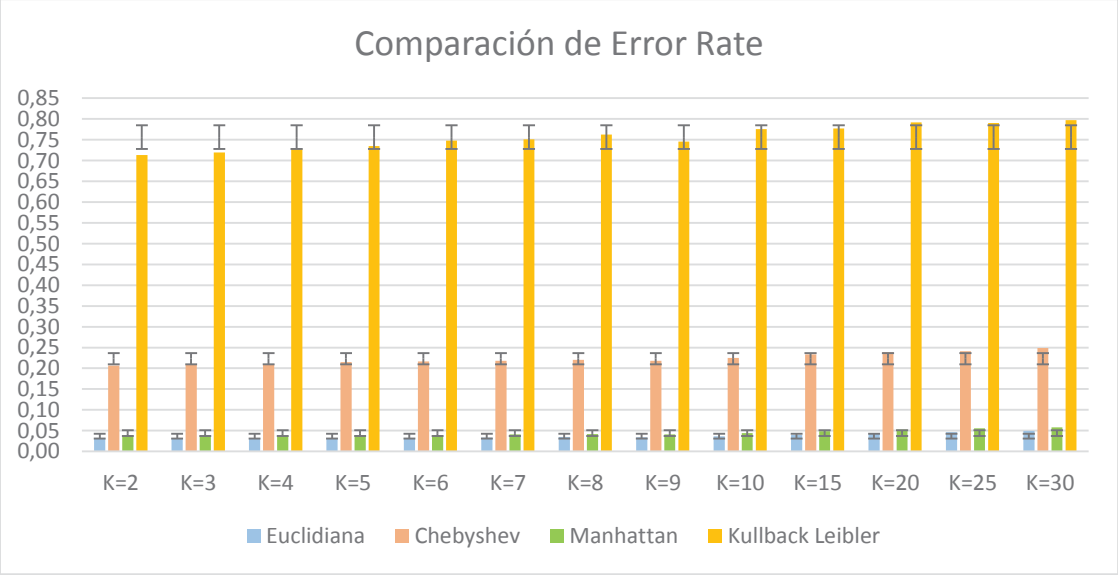

Figura 7: Gráfico de Comparación de Error Rate.

Como se muestra en la gráfica la distancia Kullback Leibler es la que obtiene la mayor tasa de Error Rate, siguiéndola la distancia Chebyshev convirtiéndolas en malas opciones a la hora de evaluar modelos de reconocimiento de dígitos manuscritos.

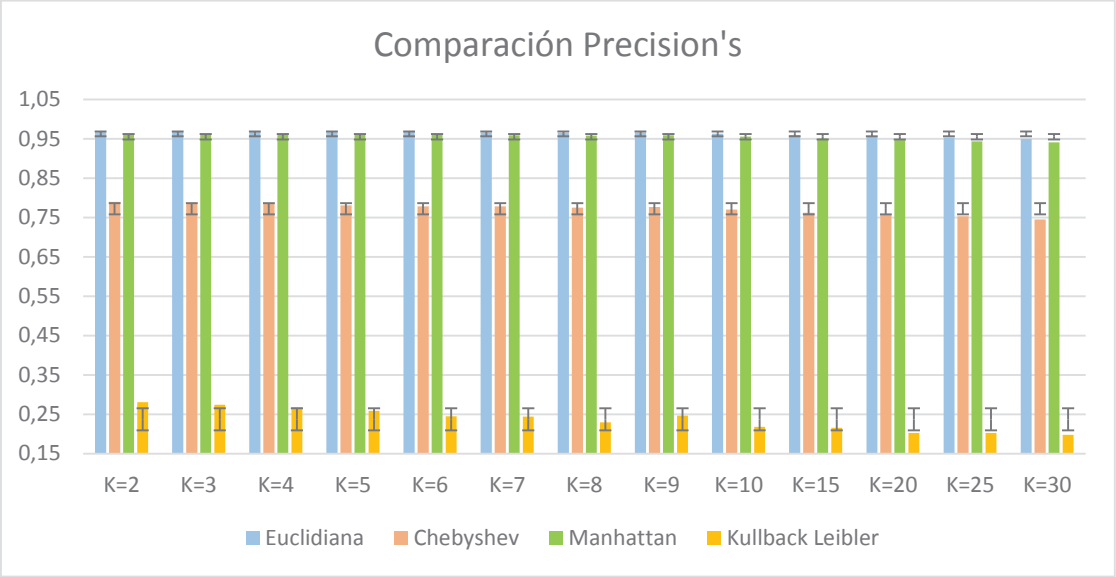

Figura 8: Gráfico de Comparación de Precision.

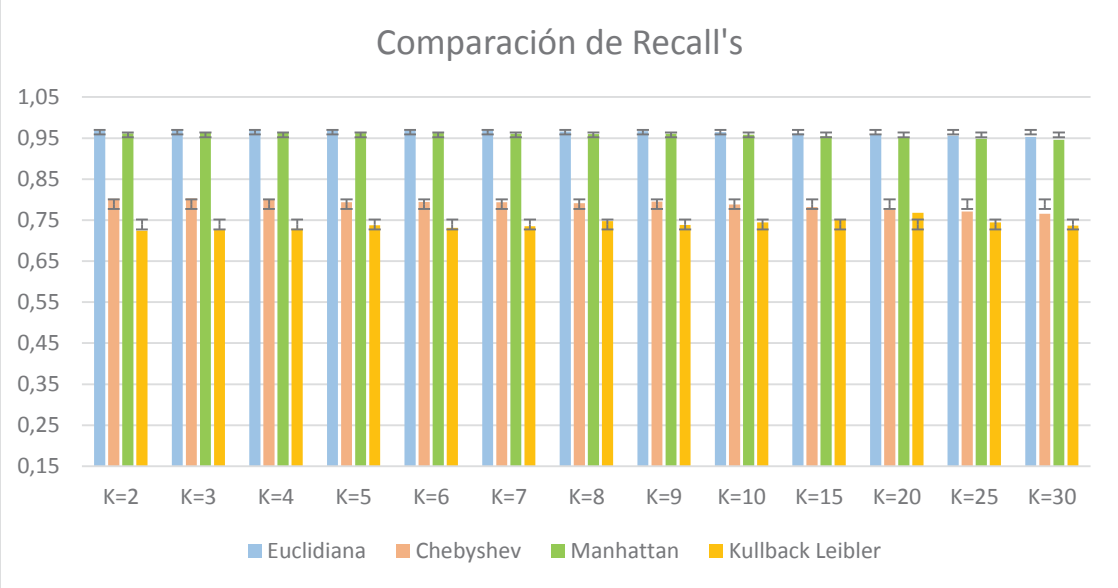

Figura 9: Gráfico de Comparación de Recall.

Como se puede observar las mejores valores de Precision's y Recall's se obtienen en las distancias Euclidiana y Manhattan, siendo las más bajas en la distancia Kullback Leibler.

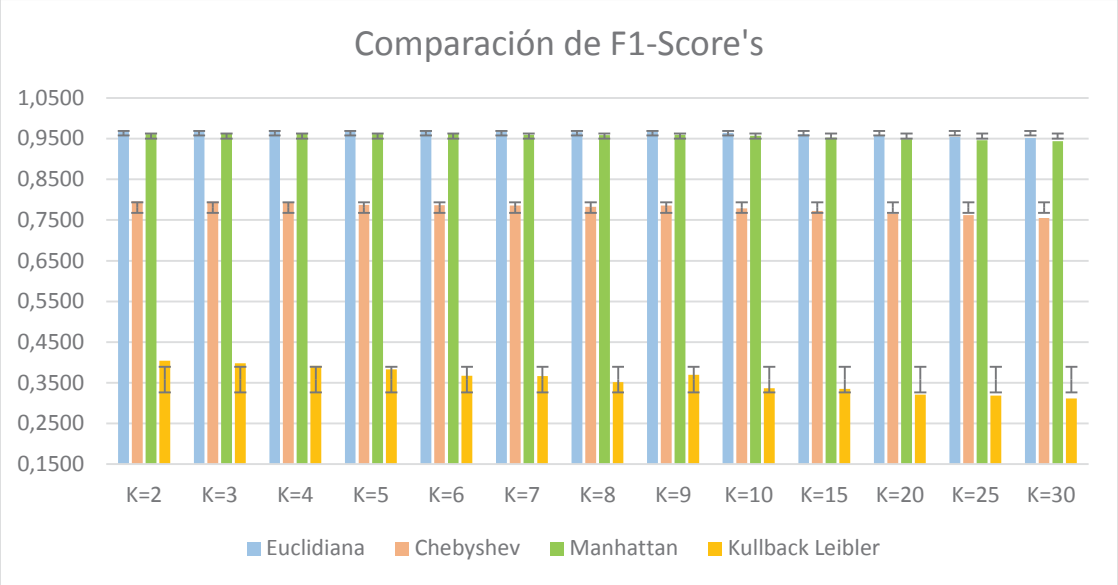

Figura 10: Gráfico comparación de F1-Score.

Como se puede observar las distancias con mejor F1-Score son la Euclidiana y la Manhattan obteniendo valores sobre el 95% en algunos valores de K, y la distancia con menor valor de su F1-Score es Kullback Leibler no superando el 40%.

Cabe destacar que como podemos observar en todas las gráficas anteriores los valores van decreciendo a medida que aumenta el valor de K, esto se debe a que mayor valor de K aumenta también su variabilidad en la asertividad del modelo por lo cual es más propenso a errores de asignación de este.

8.2 Test de Hipótesis

Se realizara un Test de Hipótesis con la medida de desempeño Accuracy y con K=3, para ello usaremos las métricas de distancia Euclidiana y Manhattan, ya que estas corresponden a los mejores resultados obtenidos, para lo cual se utilizara la siguiente formula:

$$Valor_{Est.Prueba} = \frac{(\bar{X}_1 - \bar{X}_2) - (\mu_1 - \mu_2)}{Sp\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}} \sim \mathrm{t}_{n_1+n_2-\Delta-2}$$

Donde:

$$Sp = \frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{n_1 + n_2 - 2}$$

1. **Hipótesis nula (H_0):** Sea $\mu_1 = \mu_2$
2. **Hipótesis Alternativa (H_1):** Sea $\mu_1 \neq \mu_2$
3. **Datos:** $\bar{X}_1 = 0{,}9681904761904760 \quad \bar{X}_2 = 0{,}9620535714285710$

   $S_1^2 = 0{,}0000032083184151 \quad S_2^2 = 0{,}0000042365959542$

   $n_1 = 20 \quad n_2 = 20$
4. **Región Critica:** Con $\alpha = 0.05$ y con $\mathrm{t}_{n_1+n_2-\Delta-2}$
5. **Calculo de las formulas:**

   Grados de libertad = 37

   Estadístico t = 10,0585302867799

   P(T ≤ t) una cola = $1{,}956188540511 47\mathrm{E}^{-12}$

   Valor critico de t (una cola) = 1,68709361959626

   P(T ≤ t) dos colas = $3{,}912377081022 95\mathrm{E}^{-12}$

   Valor critico de t (dos colas) = 2,02619246302911

6. **Regla de Decisión:**

   Como el Estadístico t = 10,0585302867799 es mayor al Valor critico de t (dos colas) = 2,02619246302911, se rechaza la Hipótesis Nula (H_0) de que son iguales, por lo que la diferencia es estadísticamente significativa.

9 Conclusiones

En este informe se ha presentado el modelo K-NN para dar solución a la problemática de un sistema reconocedor de dígitos manuscritos usando cuatro métricas de distancia distintas las cuales son: D. Euclidiana, D. Chebyshev, D. Manhattan, D. Kullback Leibler, utilizando la base de datos MNIST que contiene miles de imágenes de dígitos manuscritos. Con la finalidad de determinar cuál de estas arroja la menor tasa de errores.

Como hemos podido observar la problemática de un sistema reconocedor de dígitos manuscritos es relevante debido a sus diversas aplicaciones como el reconocimiento automático de códigos postales, el reconocimiento de importes en cheques bancarios y procesamiento automático de formularios, entre muchos más. Por lo cual este proyecto constituirá un aporte a dar solución a esta problemática que hasta el día de hoy esta inconclusa.

Además la metodología fue implementada en un lenguaje de programación nuevo llamado JULIA, el cual posee una vasta librería matemática y buenos tiempos de respuesta los que fue de gran ayuda a la hora de implementar y testear el modelo K-NN con cada una de sus métricas de distancias.

Dando como resultado las mejores métricas de distancias a Euclidiana y Manhattan con una Accuracy del 96% aprox. y una mal métrica de distancia a Kullback Leibler con una Accuracy del 20% aprox.

Para concluir el modelo K-NN demostró ser un buen modelo obteniendo buenos resultados con 2 métricas de distancias, además el lenguaje de programación Julia demostró ser una gran herramienta con la cual se pudo trabajar en la problemática de un sistema reconocedor de dígitos manuscritos, pudiéndose usar esta herramienta para la experimentación de otros modelos u otras problemáticas de la mismo índole.

10 Referencias

[1] S. Watanabe, *Pattern Recognition: Human and Mechanical*. New York: Wiley, 1985.

[2] A. Jain, R. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 4–37, 2000.

[3] S. Mori, C. Suen, and K. Yamamoto, "Historical review of ocr research and development," *Proceedings of the IEEE*, vol. 80, no. 7, pp. 1029–1058, 1992.

[4] C. Liu and H. Fujisawa, "Classification and learning methods for character recognition: Advances and remaining problems," in *Machine Learning in Document Analysis and Recognition* (H. F. S. Marinai, ed.), pp. 139–161, Springer, 2008.

[5] F. Bortolozzi, A. BrittoJr, L. Oliveira, and M. Morita, "Recent advances in handwritten recognition," in *Document Analysis* (U. P. et al., ed.), pp. 1–31, 2005.

[6] F. Bortolozzi, A. de Souza BrittoJr, L. Oliveira, and M. Morita, "Recent advances in handwritten recognition," *Document Analysis, Editors: Umapada Pal, Swapan Parui, Bidyut Chaudhuri*, pp. 1– 30, 2005.

[7] D. Guillevic and C. Y. Suen, "Cursive script recognition applied to the processing of bank cheques," (Montreal, Canada), pp. 11–14, Proc. of 3rd International Conference on Document Analysis and Recognition, 1995.

[8] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification - Second Edition*. John Wiley and Sons, 2001.

[9] J. Schurmann, *Pattern Classification - A unified view of statistical and neural approaches*. Wiley Interscience, 1996.

[10] A. BrittoJr, R. Sabourin, F. Bortolozzi, and C. Y. Suen, "A two-stage hmm based system for recognizing handwritten numeral srings," (Seattle, USA), pp. 396–400, Proc. of 6th International Conference on Document Analysis and Recognition, 2001.

[11] P.Nicholl, A. Amira, D. Bouchaffra, and R. Perrott, "A statistical multirresolution approach for face recognition using structural hidden markov models," *EURASIP Journal on Advances in Signal Processing*, vol. 2008, pp. 1–13, 2008.

[12] V. Vapnik, *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.

[13] L. Oliveira and R. Sabourin, "Support vector machines for handwritten numerical string recognition," (Washington DC), pp. 39–44, 9th IEEE InternationalWorkshop on Frontiers in Handwritten Recognition, IEEE Computer Society, 2004.

[14] T. Pavlidis, *Structural Pattern Recognition*. New York: Springer-Verlag, 1977.

[15] L. Perlovsky, "Conundrum of combinatorial complexity," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 6, pp. 666–670, 1998.

[16] C. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[17] R. Lopez and E. O˜nate, "A software model for the multilayer perceptron," (Salamanca, Spain), pp. 464–468, IADIS International Conference: Applied Computing, IADIS Press, 2007.

[18] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[19] D. Zhang, X. Bai, and K. Cai, "Extended neuro-fuzzy models of multilayer perceptrons," *Fuzzy Sets and Systems*, vol. 142, no. 2, pp. 221–242, 2004.

[20] D. Keysers, "Comparison and combination of state-of-the-art techniques for handwritten character recognition: Topping the mnist benchmark," Technical Report, IUPR Research Group, DFKI and Technical University of Kaiserslautern, 2006.

[21] T. Kohonen, *Self-Organizing Maps*. Springer-Verlag, 2001.

[22] S. Haykin, *Neural Networks A Comprehensive Foundation*. Prentice Hall, 1999.

[23] S. Behnke, M. Pfister, and R. Rojas, "A study on the combination of classifiers for handwritten digit recognition," (Magdeburg/Germany), 3rd International Workshop on Neural Networks in Applications NN '98, 1998.

[24] L. Xu, A. Krzyzak, and C. Suen, "Methods of combining multiple classifiers and their applications to handwritten recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 3, pp. 418–435, 1992.

[25] J. Kittler, "Combining classifiers: A theoretical framework," *Pattern Analysis & Applications*, vol. 1, no. 1, pp. 18–27, 1998.

[26] L. Oliveira, R. Sabourin, F. Bortolozzi, and C. Y. Suen, "Automatic recognition of handwritten numerical strings: A recognition and verification strategy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 11, pp. 1438–1454, 2002.

[27] http://julialang.org/.

[28] Y. LeCun, "The MNIST database of handwritten digits," http://yann.lecun.com/exdb/mnist/.

[29] J.L. Hodges (1989). « (1951): An Important Contribution to Nonparametric Discriminant Analysis and Density Estimation: Commentary on Fix and Hodges (1951) ». *International Statistical Review / Revue Internationale de Statistique.*

[30] Duda, R. O., & Hart, P. E. (1973). Pattern classification and scene analysis. John Wiley & Sons.

[31] Lachiche, N., & Flach, P. A. (2003). Improving accuracy and cost of two-class and multi-class probabilistic classifiers using ROC curves. In Proceedings of ICML'2003 (pp. 416–423).

[32] Devijver, P. A., and J. Kittler, Pattern Recognition: A Statistical Approach, Prentice-Hall, Londres, 1982.

[33] Jean-Philippe Lang, Predictors tutorial, *Bioinformatic Department Projects.*

[34] Information Processing & Management. http://www.journals.elsevier.com/information-processing-and-management

[35] Evaluating a classification model. http://www.cs.odu.edu/~mukka/cs795sum10dm/Lecturenotes/Day4/recallprecision.pdf

[36] F-Score. https://en.wikipedia.org/wiki/F1_score

[37] Test de Hipótesis. Material: Clases Estadística Computacional UTFSM, 2009.

11 Anexos

11.1 Código Precisión en Julia

```
@everywhere function Cal_Precision(MC)
   i = 1
   PR = 0
   Prec = 0
   for i in 1:10
        Precision = MC[i, i] / (MC[i, 1] + MC[i ,2] + MC[i, 3] + MC[i, 4] + MC[i,5] + MC[i,
                          6] + MC[i, 7] + MC[i, 8] + MC[i, 9] + MC[i, 10])
    PR = PR + Precision
    Prec = PR / 10
   end
   return Prec
 end
```

11.2 Código Recall en Julia

```
@everywhere function Cal_Recall(MC)
   i = 0
   RE = 0
   REC = 0
   for i in 1:10
     Recall = MC[i, i] / (MC[1, i] + MC[2, i] + MC[3, i] + MC[4, i] + MC[5, i] + MC[6, i]
              + MC[7, i] + MC[8, i] + MC[9, i] + MC[10, i])
    RE = RE + Recall
    REC = RE / 10
   end
   return REC
 end
```

11.3 Código función generar etiquetas

```
yPredictionseuc = @parallel (vcat) for i in 1:size(xTest, 2)

  nRows = size(xTrain, 1)

  imageIeuc = Array(Float32, nRows)

  for index in 1:nRows

   imageIeuc[index] = xTest[index, i]

  end

  assign_label(xTrain, yTrain, k, imageIeuc, Euc)

 end
```

11.4 Código función asignar etiqueta

```
@everywhere function assign_label(xTrain, yTrain, k, imageI, dist)
  kNearestNeighbors = get_k_nearest_neighbors(xTrain, imageI, k, dist)
  counts = Dict{Int, Int}()
  highestCount = 0
  mostPopularLabel = 0
  for n in kNearestNeighbors
   labelOfN = yTrain[n]
   if !haskey(counts, labelOfN)
    counts[labelOfN] = 0
   end
   counts[labelOfN] += 1 #add one to the count
   if counts[labelOfN] > highestCount
    highestCount = counts[labelOfN]
    mostPopularLabel = labelOfN
   end
  end
  return mostPopularLabel
 end
```

11.5 Código Obtener el vecino mas cercano

```
@everywhere function get_k_nearest_neighbors(xTrain, imageI, k, dist)
  nRows, nCols = size(xTrain)
  imageJ = Array(Float32, nRows)
  distances = Array(Float32, nCols)
  for j in 1:nCols
   for index in 1:nRows
    imageJ[index] = xTrain[index, j]
   end
   if dist == 1
     distances[j] = euclidean_distance(imageI, imageJ)
     #println("Usando distancia Euclideana")
   end
   if dist == 2
     distances[j] = Chebyshev(imageI, imageJ)
     #println("Usando distancia Chebyshev")
   end
   if dist == 3
     distances[j] = Manhattan(imageI, imageJ)
     #println("Usando distancia Manhattan")
   end
   if dist == 4
     distances[j] = Kullback_Leibler(imageI, imageJ)
     #println("Usando distancia Kullback Leibler")
```

```
 end
end
sortedNeighbors = sortperm(distances)
kNearestNeighbors = sortedNeighbors[1:k]
return kNearestNeighbors
end
```