UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

COMPARACIÓN Y ANÁLISIS DE LAS CURVAS DE APRENDIZAJE SOBRE LA BASE DE DATOS MNIST DE LOS ALGORITMOS DE CLASIFICACIÓN DE LA LIBRERÍA SK-LEARN

POR
ALAN ARNOLDO ALCANTAR GÓMEZ

COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE MAESTRÍA EN CIENCIAS EN INGENIERÍA DE SISTEMAS

JULIO, 2019

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

SUBDIRECCIÓN DE ESTUDIOS DE POSGRADO

COMPARACIÓN Y ANÁLISIS DE LAS CURVAS DE APRENDIZAJE SOBRE LA BASE DE DATOS MNIST DE LOS ALGORITMOS DE CLASIFICACIÓN DE LA LIBRERÍA SK-LEARN

POR

ALAN ARNOLDO ALCANTAR GÓMEZ

COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE MAESTRÍA EN CIENCIAS EN INGENIERÍA DE SISTEMAS

JULIO, 2019

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Subdirección de Estudios de Posgrado

Los miembros del Comité de Tesis recomendamos que la Tesis «Comparación y análisis de las curvas de aprendizaje sobre la base de datos MNIST de los algoritmos de clasificación de la librería SK-Learn», realizada por el alumno Alan Arnoldo Alcantar Gómez, con número de matrícula 1935040, sea aceptada para su defensa como requisito parcial para obtener el grado de Maestría en Ciencias en Ingeniería de Sistemas.

El Comité de Tesis

Dr. José Arturo Berrones Santos

Asesor

Dr. Romeo Sánchez Nigenda

Revisor

Dr. Francisco Javier Almaguer Martínez

Revisor

Vo. Bo.

Dr. Simón Martínez Martínez

Subdirección de Estudios de Posgrado

SUBDIRECCIÓN DE ESTUDIOS DE POSGRADO
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA
U.A.N.L.
FIME

San Nicolás de los Garza, Nuevo León, julio 2019

A mi familia.

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

Agradecimientos al Posgrado en Ingeniería de Sistemas y a la Universidad Autónoma de Nuevo León por aceptarme en la Maestría en Ciencias en Ingeniería de Sistemas. Al cuerpo académico de PISIS les agradezco por formar parte de mi formación académica. Agradezco a mis compañeros de generación por la gran calidad de personas que son y hacer mi estancia en este posgrado muy amena. Por ultimo agradecer a CONACyT por la beca otorgada para poder realizar la maestría.

Resumen

Alan Arnoldo Alcantar Gómez.

Candidato para obtener el grado de Maestría en Ciencias en Ingeniería de Sistemas.

Universidad Autónoma de Nuevo León.

Facultad de Ingeniería Mecánica y Eléctrica.

Título del estudio: Comparación y análisis de las curvas de aprendizaje sobre la base de datos MNIST de los algoritmos de clasificación de la librería SK-Learn.

Número de páginas: 78.

Objetivos: Realizar un análisis estadístico a las curvas de aprendizaje que se obtienen de un diseño de experimentos factorial de los algoritmos de clasificación más utilizados de la librería SK-Learn para un problema de reconocimiento óptico de caracteres, con la finalidad de identificar las interacciones entre los parámetros de cada algoritmo y como estas afectan al comportamiento del algoritmo.

Método de estudio: Realizar un análisis del estado del arte para identificar los valores habituales de los parámetros de cada algoritmo con los que se obtienen las mejores tasas de reconocimiento y a partir de estos establecer los niveles de cada parámetro para desarrollar el diseño de experimentos factorial. Con la finalidad de poder realizar un análisis estadístico se generaron varios conjuntos de entrenamiento

de forma aleatoria, balanceada y de diferentes tamaños sobre los cuales se realizó el diseño de experimentos, con el objetivo de obtener las curvas de aprendizaje y el comportamiento de los algoritmos.

CONTRIBUCIONES: Con el uso de pruebas estadísticas identificar cuales parámetros o la interacción entre estos tienen efectos en el comportamiento de los algoritmos para resolver el problema de reconocimiento de dígitos manuscritos de la base de datos MNIST. Aproximar la forma en que los parámetros de un algoritmo influyen en el rendimiento de este. Dejar los resultados como un punto de referencia para medir la generalización de los algoritmos al resolver problemas con otras características.

CONLUSIONES: Identificar que para algunos modelos sus parámetros no influyen en el rendimiento para el reconocimiento de dígitos manuscritos. En algunos algoritmos solo es necesario utilizar un porcentaje del conjunto de datos para obtener el comportamiento y limites de un algoritmo.

Firma del asesor:

Dr. José Arturo Berrones Santos

Capítulo 1

Introducción

El campo de investigación en aprendizaje máquina tomo fuerza durante el siglo pasado por su capacidad de resolver problemas complejos por medio de algoritmos que en la época eran simples y sencillos que permitían identificar patrones presentes en un conjunto de información. El aprendizaje máquina ha tenido un profundo impacto en diferentes ámbitos pero destaca su implementación en el área de ***visión computacional*** donde el objetivo es obtener información a partir de imágenes de distintos formatos, ya sean fotografías, impresiones, escaneos, video, etc [26]. El problema del ***reconocimiento óptico de caracteres*** (ROC) [7], [25] es el subdominio del procesamiento de imágenes más popular desde principios de 1930 que consiste básicamente en el reconocimiento de letras y números, es un tópico activamente estudiando en la industria y la academia por el inmenso potencial de aplicaciones como lo son el reconocimiento de pasaportes, facturas, extractos bancarios, recibos computarizados, tarjetas de crédito, correo, impresiones, etc. El objetivo principal del ROC es la digitalización de textos impresos para que puedan editarse, buscarse, almacenarse de manera más compacta, mostrarlos en línea y utilizarlos en alguna tarea en particular, por ello el ROC fue de los problemas que impulsaron el interés y desarrollo de la visión computacional debido a la complejidad del problema y sus aplicaciones.

El aprendizaje máquina consiste en formular un modelo el cual puede ser probabilístico, paramétrico o no-paramétrico donde las probabilidades, parámetros o estructura se obtienen al ajustar el modelo a un conjunto de datos llamado ***datos de entrenamiento*** en un proceso llamado ***entrenamiento***. Cuando los datos de entrenamiento están compuestos por un grupo de ***características/atributos***, conocido como ***vector/dato de entrada*** y por un grupo de ***etiquetas***, conocido como ***vector/dato de salida*** se trata de un problema de aprendizaje supervisado y en el caso que las etiquetas formen un grupo finito con elementos enteros donde el objetivo sea responder dado un vector de entrada que vector de salida le corresponde se le conoce como ***problema de clasificación***, de modo que diferentes vectores de entrada pueden tener el mismo vector de salida y a cada vector de entrada le corresponde solo un vector de salida, como es el caso del reconocimiento óptico de caracteres donde se desea identificar una letra o número.

El proceso de entrenamiento consiste en estimar las probabilidades, parámetros o estructura que permita empatar de forma correcta a cada uno de los vectores de entrada con su vector de salida del conjunto de entrenamiento con el objetivo de que el número de clasificaciones estimadas por el modelo que son erróneas sobre el número total de clasificaciones conocido como ***error de entrenamiento*** sea lo menor posible. La idea de los modelos de aprendizaje máquina es que sean entrenados con un conjunto de entrenamiento y después se ponga a prueba sobre un conjunto de datos diferente al de entrenamiento llamado ***conjunto de prueba*** para medir la ***generalización***, la cual se entiende como la capacidad de un modelo para clasificar un vector de entrada que no fue utilizado para el entrenamiento, esto porque en la practica el modelo nunca clasificara un vector de entrada que fue utilizado para él entrenamiento. Como como consecuencia el número de clasificaciones estimadas por la máquina de aprendizaje que son erróneas en comparación con el número de total de clasificaciones en un conjunto de datos de prueba se le conoce como ***error de generalización o error de prueba***.

Antes de 1998 no se tenía una base de datos de referencia para el ROC que presentara las características deseadas que deben tener los conjuntos de entrenamiento y prueba, de modo que [14] tomaron las bases de datos especiales 1 y 3 del Instituto Nacional de Estándares y Tecnología (NIST) para formar una base de datos enfocada al reconocimiento de dígitos (0-9) manuscritos, bautizada como ***MNIST***. Cuando la MNIST fue publicada tenia el objetivo de convertirse en la base de datos de referencia para medir y probar el rendimiento de cualquier sistema de reconocimiento de dígitos, tal es la importancia que hoy en día es la base de datos sobre la cual se debe probar cualquier algoritmo que se presente en la actualidad para ser reconocido por la comunidad científica.

En el estado del arte actual para el ROC se observa que la comunidad científica se ha enfocado en desarrollar modelos y algoritmos especializados como es agregar términos de regularización, términos de penalización, combinar modelos para formar uno hibrido que tenga las mejores características de cada modelo por separado de forma que se complementen, desarrollar nuevas arquitecturas en las redes neuronales [22], extracción de características y selección de características. Otras características encontradas en el estado del arte es que los investigadores expanden el conjunto de entrenamiento del MNIST al aplicar transformaciones a los datos de entrenamiento [24], [21], [8], [29] y el uso de plataformas especializadas en utilizar las unidades de procesamiento grafico (GPU) como son Pytorch, Keras, TensorFlow con el único propósito se reducir el error de generalización y los tiempos de entrenamiento.

Las críticas encontradas es que todo avance logrado por la academia y los centros de investigación no se transfiere a un nivel practico, porque tener la infraestructura para guardar y procesar grandes cantidades de datos que permitan el entrenamiento de algoritmos complejos solo está al alcance de pocos. Surge de ma-

nera natural preguntarse si los modelos y algoritmos encontrados en el estado del arte pueden ser implementados en una plataforma como ***SK-Learn*** [18] que no requiere de GPUs y es fácil de implementar por personal no especializado estando al alcance de todos.

Otra cuestión a analizar es la práctica común de incrementar el conjunto de datos de entrenamiento, porque no se explica si los algoritmos que se presentan necesitan una cantidad mayor de datos de entrenamiento para que el error de generalización se compare con los mejores errores de generalización encontrados en la literatura o simplemente su error de generalización para el conjunto de MNIST es malo y requiere de una mayor cantidad de información para que el error de generalización sea comparable con algún otro algoritmo. En base a lo anterior es necesario obtener las ***curva de aprendizaje*** [9], las cuales consisten en graficar el error de generalización contra el tamaño del conjunto de entrenamiento con el cual fue entrenado el algoritmo, de esta manera es posible modelar la generalización del error de prueba como una función del tamaño del conjunto de entrenamiento [19] y obtener el comportamiento del algoritmo bajo situaciones diferentes.

1.1 Motivación

La motivación del presente trabajo es responder si es posible transferir los avances en el estado del arte del reconocimiento de dígitos utilizando la base de datos MNIST a un nivel practico y analizar si el error de generalización tiene el mismo comportamiento al ser entrenado con un conjunto de entrenamiento cuyo tamaño es menor al presentado en el estado del arte.

1.2 Objetivo

Identificar cuales modelos del estado del arte para el MNIST son posible de implementar en la librería SK-Learn y desarrollar un diseño de experimentos factorial para cada uno de los modelos. Entre los factores de los diseños de experimentos agregar como factor el tamaño del conjunto de entrenamiento con el fin de obtener el error de generalización al entrenar con conjuntos de entrenamiento de diferentes tamaños para obtener las curvas de aprendizaje y observar el efecto de cada factor sobre el error de generalización. En base a los resultados de cada uno de los diseños de experimentos realizar un análisis estadístico para determinar diferencias significativas entre los niveles de cada factor. Todo lo anterior con el objetivo de que las curvas de aprendizajes obtenidas para el error de generalización en función del tamaño del conjunto de entrenamiento queden como referencias para promover el análisis del error de generalización a nivel practico de los futuros modelos y algoritmos con el fin de que los avances en el estado del arte tengan impacto directo a todos los niveles de la sociedad.

1.3 Hipótesis

La misma combinación de niveles de factores tiene un efecto diferente en el error de generalización de un modelo de clasificación para el reconocimiento de dígitos manuscritos cuando es entrenado sobre conjuntos de diferentes tamaños de la MNIST.

Capítulo 2

Antecedentes y análisis literario

Ha continuación se presentan los artículos sobre la base de datos MNIST encontrados en la literatura los cuales presentan curvas de aprendizaje entre sus resultados.

En 1998 se presentó el primer artículo [13] que realizo la revisión de varios modelos de aprendizaje máquina y los comparo por su rendimiento directamente sobre el MNIST, sin realizar algún tipo de extracción de características. El modelo de LeNet5 fue el único en el que se realizó una curva de aprendizaje con tamaños de conjunto de entrenamiento 15 000, 30 000 y 60 000, donde la estrategia de entrenamiento fue reducir la tasa de aprendizaje a medida que se avanzaba en las iteraciones, obteniendo un porcentaje de error de 0.95 % sobre el conjunto de prueba al utilizar los 60 000 datos para entrenar. Con el objetivo de mejorar el porcentaje de error se generaron 540 000 datos a partir de distorsiones teniendo un conjunto de 600 000 en total y consiguiendo un porcentaje de error de 0.8 % sobre el conjunto de prueba. Entre los modelos de aprendizaje máquina que se compararon se encuentran K-vecinos más cercanos, SVM y perceptrón multicapa (una capa oculta). Por otra parte la comparación de los modelos también se hizo usando una versión modificada del MNIST cuya representación matricial es de 20×20. Los problemas

encontrados es que la comparativa de los modelos se hicieron al ser entrenados con el conjunto completo de MNIST y que no se mencionan la forma en que se generaron los conjuntos de entrenamiento de 15 000 y 30 000 datos para la curva de aprendizaje.

Durante 2002 [3] presentaron una aproximación para medir la similitud entre las siluetas de dos dígitos, lo que hicieron fue definir un costo para emparejar los puntos que forman parte del contorno de un digito con los puntos de contorno de otro dígito. Una vez que se tienen emparejados todos los puntos de contorno de dos dígitos se estima una transformación de alineación para mapear los puntos de un objeto hacia el otro. Lo anterior les permitió definir la distancia entre dígitos como una función del costo de emparejamiento y a la transformación de alineación como una medida de similitud entre los dos dígitos. Para finalizar hicieron uso de un algoritmo k-medoides para obtener los k dígitos que representan mejor cada una de las clases, de modo que al tener n clases se tiene un conjunto $n \cdot k$ prototipos, teniendo este conjunto de prototipos se aplicó un algoritmo de k-vecinos más cercanos para clasificar un digito de prueba tomando como criterio la distancia entre dígitos. Entre sus resultados presentan varias curvas de aprendizaje con diferentes k para el algoritmo de vecinos más cercanos y una comparativa entre las curvas de aprendizaje al utilizar la distancia entre dígitos y la suma de diferencias cuadradas para observar la ganancia en generalización de la medida propuesta.

Para 2007 [12] se publicó un artículo enfocado en las tareas de extracción de características y clasificación, apelando que algunos modelos no procesan eficientemente los datos en su formato original, como bien se menciona en [13]. El algoritmo presentado fue bautizado como TFE-SVM el cual consiste en una red neuronal convolucional LeNet5 para obtener como salida un vector de características extraídas de tamaño 120, el cual se pasa como entrada a un algoritmo de SVM. Con el objetivo de demostrar la ganancia en la generalización del TFE-SVM se compararon las curvaturas de aprendizaje contra una LeNet5 donde los hiperparámetros del SVM

se ajustaron haciendo uso de una muestra de 15 000 elementos y los tamaños del conjunto de entrenamiento para obtener las curvas de aprendizaje fueron las mismas que en [13], es decir 15 000, 30 000 y 60 000. Los resultados obtenidos concluyeron que el algoritmo TFE-SVM generaliza mejor que una LeNet5 obteniendo un porcentaje de error del 0.83 % al utilizar los 60 000 elementos y usando el kernel rbf en la parte del SVM.

Con la premisa que una base de datos grande no solo dirige a modelos con mayor precisión, sino que además permite aprender un modelo más complejo el cual puede dirigir a una mejora sustancial en el rendimiento, en 2009 [15] Proponen un nuevo cuadro de trabajo el cual permite a un algoritmo de aprendizaje ajustar la complejidad del modelo aprendido a la cantidad de datos de entrenamiento disponibles en ese momento, desarrollando un sistema escalable y adaptativo el cual es capaz de continuar aprendiendo. Se implemento un algoritmo de SVM con Kernels polinomial y función de base radial, donde los datos de entrenamiento fueron recibidos por el algoritmo de forma aleatoria uno por uno en cada paso del tiempo y el estimador para medir el rendimiento fue el porcentaje de error sobre el conjunto de prueba. En el caso de usar función de base radial como kernel obtuvieron que al tener 1 293, 1 948 y 22 000 datos de entrenamiento los valores óptimos de γ fueron 0.008, 0.016 y 0.32 respectivamente, y en caso de usar un kernel polinomial al tener 136 y 1 845 datos de entrenamiento los valores óptimos para el grado del polinomio fueron 2 y 3 respectivamente. Al pasar todo el conjunto de entrenamiento, es decir los 60 000 datos obtuvieron un porcentaje de error sobre el conjunto de entrenamiento alrededor de 0.5 % para ambos kernels.

En los años recientes el aporte científico que se encuentra en la literatura es agregar términos de penalización a los modelos originales para mejorar su rendimiento, combinar dos modelos para formar uno hibrido que tenga las mejores características de ambos modelos por separados de forma que se complementen, desa-

rrollar nuevas arquitecturas en las redes neuronales, utilizar extracción y selección de características. Como ejemplo a lo anterior se tienen los casos [28], [23]. Existen dos problemas con los trabajos de la actualidad, el primero de ellos es que la mayoría de los artículos donde se propone un nuevo modelo, no dedican una parte del trabajo a analizar las curvas de aprendizaje y se enfocan solo en obtener el mejor rendimiento posible. El segundo problema es que las practicas mencionadas al inicio a los modelos base hacen que el modelo resultante solo funcione para la tarea que están tratando de mejorar, perdiendo generalización de poderse aplicar a problemas con otras características. El gran aporte de las curvas de aprendizaje es conocer el proceso de aprendizaje y las limitaciones que tiene el modelo.

Otro aspecto encontrado en la literatura es que aquellos artículos cuyo objetivo es comparar el rendimiento de diferentes modelos solo hacen las comparaciones tomando el conjunto de datos completo para entrenar, sin importar como es el comportamiento de la curva de aprendizaje [17], [30], [16] y [1]. Lo anterior tiene suposiciones graves, como el omitir que las variables de los modelos no tienen interacción con el tamaño del conjunto de entrenamiento o querer extrapolar los resultados. Retomando los artículos que hacen comparaciones entre modelos, también existe la problemática que las plataformas donde se desarrollaron los algoritmos para cada modelo son particulares, es decir que los experimentos no se pueden reproducir por cualquier persona.

Capítulo 3

Marco teórico

Antes de iniciar con la descripción de los modelos primero se define el problema y la terminología. La MNIST es un problema de clasificación multiclase (10 clases), es decir que la salida de un algoritmo de clasificación es solo una variable escalar la cual puede tener uno de los diez valores posibles (***etiquetas de clase***). En el caso de la MNIST se tiene un conjunto de datos de entrenamiento compuesto por el conjunto $\mathbf{X}$, el cual está formado por 60 000 vector de entrada $\vec{\mathbf{x}}$ y el conjunto $\mathbf{Y}$, el cual está formado por las 60 000 etiquetas de salida $\mathbf{y}$. Cada dígito de la MNIST se representa por un vector de 785 donde la primera componente indica el digito (etiqueta) y las 784 componentes restantes son valores entre 0 y 255, el cual al ser representado en una matriz de 28x28 en escala de grises es como se consigue la imagen del digito escaneado.

$\mathbf{X}$, $\mathbf{Y}$ conjunto de vectores de entrada de entrenamiento y conjunto de etiquetas de salida de entrenamiento respectivamente

$\vec{\mathbf{x}}_i,\ i = 1, \ldots, 60\ 000$ vectores de entrada de entrenamiento

$\mathbf{y}_i,\ i = 1, \ldots, 60\ 000$ etiquetas de salida de entrenamiento

$x_{ij},\ j = 1, \ldots, 784$ componente j del vector de entrada i

Existen dos formas de modelar el problema de clasificación múltiple y depende del algoritmo puede utilizar una o ambas formas. La primer enfoque de modelar consiste en considerar que se está trabajando con múltiples modelos de clasificación binarios, a este enfoque se le llama *uno vs resto*, es decir que se toma una clase y el resto de clases agruparla como una sola clase, para después ajustar el modelo, y esto se realiza para cada una de las clases, por lo que al resolver el problema de MNIST utilizando este enfoque se ajustan diez modelos binarios. El segundo enfoque consiste en tomar en cuenta las diez clases al mismo tiempo, llamando a este enfoque *uno vs uno*.

$\mathbf{C} = \{c_0,\ c_1\}$ conjunto de etiquetas posibles para el enfoque uno vs resto

$\mathbf{C} = \{c_1,\ \ldots,\ c_{10}\}$ conjunto de etiquetas posibles para el enfoque uno vs uno

Con esto se tiene que cada una de las etiquetas de salida de entrenamiento tiene un valor que pertenece al conjunto de etiquetas posibles, es decir $\mathbf{y}_i \in \mathbf{C}$.

3.1 Modelos probabilísticos

En este trabajo todos los modelos probabilísticos se desarrollan a partir del teorema de Bayes, el cual indica que la probabilidad de pertenecer a una clase c dado un vector de entrada de prueba $\vec{\mathbf{x}}$ se escribe de la siguiente manera:

$$\underbrace{p(c|\vec{\mathbf{x}})}_{posterior} \propto \underbrace{p(\vec{\mathbf{x}}|c)}_{verosimilitud} \cdot \underbrace{p(c)}_{priori} \tag{3.1}$$

El objetivo de estos modelos es modelar o sumir una distribución para la probabilidad posterior o la probabilidad de verosimilitud. Debido a que la probabilidad a priori es la proporción de cada clase respecto a todo el conjunto de entrenamiento y que los conjuntos de entrenamiento se formaron de forma balanceada, la probabilidad a priori es una constante con valor 1/10 que puede ser ignorada. A continuación, se presentan los modelos probabilísticos.

3.1.1 Análisis de discriminante lineal (LDA) [10]

El modelo de análisis de discriminante lineal asume que la probabilidad de verosimilitud $p(\vec{\mathbf{x}}|c)$ es una ***distribución gaussiana multivariante***:

$$p(\vec{\mathbf{x}}|c) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} exp\left(-\frac{1}{2}(\vec{\mathbf{x}} - \mu_c)^\top \Sigma^{-1}(\vec{\mathbf{x}} - \mu_c)\right) \tag{3.2}$$

En el caso de este modelo la distribución Gaussiana multivariante para cada clase comparte la misma matriz de covarianza por ello solo aparece Σ en 3.2, está suposición es lo que provoca que la superficie de decisión para separar las clases sea lineal y que a un dato de prueba $\vec{\mathbf{x}}$ se le asigne la clase que maximice la siguiente función:

$$\hat{\mathbf{y}}(\vec{\mathbf{x}}) = argmax_c\{\vec{\mathbf{x}}^\top \Sigma^{-1}\mu_c - \frac{1}{2}\mu_c^\top \Sigma^{-1}\mu_c + log\,(p(c))\} \tag{3.3}$$

3.1.2 Análisis de discriminante cuadrático (QDA)[10]

En el caso del análisis discriminante cuadrático la probabilidad de verosimilitud $p(\vec{\mathbf{x}}|c)$ se modela como una ***distribución gaussiana multivariante***, sin embargo a diferencia del análisis discriminante lineal aquí no se asume que la matriz de covarianza sea la misma para todas las clases, por lo tanto se debe estimar la matriz

de covarianza para cada clase. De forma que la probabilidad de verosimilitud para una clase está dada por:

$$p\left(\vec{\mathbf{x}}|c\right) = \frac{1}{(2\pi)^{d/2}|\Sigma_c|^{1/2}} exp\left(-\frac{1}{2}\left(\vec{\mathbf{x}} - \mu_c\right)^\top \Sigma_c^{-1}\left(\vec{\mathbf{x}} - \mu_c\right)\right) \tag{3.4}$$

Aplicando el logaritmo a la función 3.4 y algebra se tiene que la superficie de decisión para separas las clases es cuadrática de ahí el nombre del modelo, por lo tanto a un dato de prueba $\vec{\mathbf{x}}$ se le asigna la clase que maximiza la siguiente función:

$$\hat{\mathbf{y}}(\vec{\mathbf{x}}) = argmax_c\{-\frac{1}{2}\vec{\mathbf{x}}^\top\Sigma_c^{-1}\vec{\mathbf{x}} + \vec{\mathbf{x}}^\top\Sigma_c^{-1}\mu_c - \frac{1}{2}\mu_c^\top\Sigma_c^{-1}\mu_c - \frac{1}{2}log|\Sigma_c| + log(p(c))\} \tag{3.5}$$

3.1.3 Bayes ingenuo Gaussiano (GNB) [11]

Antes de describir el modelo Bayes ingenuo gaussiano hay que reescribir la probabilidad de verosimilitud, en la ecuación 3.1 la probabilidad de verosimilitud indica cual es la probabilidad de obtener un vector, de modo que es posible escribirla dejando las componentes del vector de forma explícita:

$$p(\vec{\mathbf{x}}|c) = p(x_1, ..., x_{784}|c) \tag{3.6}$$

De modo que cada una de las componentes x_j es una variable aleatoria. La idea de los modelos Bayes ingenuo es asumir independencia entre las variables aleatorias, de modo que la probabilidad de verosimilitud se puede expresar como la multiplicación de las probabilidades condicionales de que las variables aleatorias (componente) x_j aparezca en la clase c y se interpreta como una medida de evidencia de que tanto contribuye x_j para que la clase c sea la correcta.

$$p(x_1, ..., x_{784}|c) = \prod_{j=1}^{784} p(x_j|c) \tag{3.7}$$

Retomando el modelo Bayes ingenuo gaussiano, supone que la distribución $p(x_j|c)$ es una distribución gaussiana:

$$p(x_j|c) = \frac{1}{\sqrt{2\pi\sigma_{jc}^2}} exp\left(-\frac{(x_j - \mu_{jc})^2}{2\sigma_{jc}^2}\right) \tag{3.8}$$

Donde:

$$\mu_{jc} = \frac{1}{|\mathbf{X}_c|}\sum_{i\in\mathbf{X}_c} x_{ij}, \quad \sigma_{jc}^2 = \frac{1}{|\mathbf{X}_c|}\sum_{i\in\mathbf{X}_c}(x_{ij} - \mu_{jc})^2 \tag{3.9}$$

$\mathbf{X}_c$ es el subconjunto de índices del conjunto de datos que pertenecen a la clase $c \in \mathbf{C}$, μ_{jc} es el promedio en la componente j de los datos que pertenecen a la clase $c \in \mathbf{C}$ y σ_{jc}^2 es la varianza en la componente j de los datos que pertenecen a la clase $c \in \mathbf{C}$. Para evitar probabilidades con valor cero a cada una de las varianzas σ_{jc}^2 se les agrega un parámetro de suavizado $\alpha \leq 1$ de la varianza más grande correspondiente a su componente y clase.

3.1.4 Bayes ingenuo multinomial (MNB)

Este modelo supone que las variables aleatorias x_j presentan una distribución multinomial, por lo tanto, las probabilidades $p(x_j|c)$ son conteos de frecuencia relativa de la componente x_j en la clase perteneciente c:

$$p(x_j|c) = \frac{N_{jc} + \alpha}{N_c + n\alpha} \tag{3.10}$$

Donde N_{jc} es el número de ocurrencias de x_j dentro del conjunto de datos que

pertenecen a la clase c, N_c es el total de ocurrencias de todas las componentes para los datos que perteneces a la clase c y $\alpha \geq 0$ es un parámetro de suavizado que cuenta para las componentes que no presentan una muestra de conteo y previene que existan probabilidades de valor cero:

$$N_{jc} = \sum_{i \in \mathbf{X}_c} x_{ij}, \quad N_c = \sum_{j=1}^{784} N_{jc} \tag{3.11}$$

De nuevo $\mathbf{X}_c$ es el subconjunto de índices del conjunto de datos que pertenecen a la clase $c \in \mathbf{C}$.

3.1.5 Bayes ingenuo binomial (BNB)

Este modelo se declara que las variables aleatorias siguen una distribución binomial, eso quiere decir que cada una de las componentes x_j es asumida como binaria, lo que genera un indicador en cada componente, donde 1 indica la presencia de esa componente en la clase o 0 la ausencia. Para poder hacer uso de esta interpretación es necesario binarizar los vectores de entrada haciendo uso de un parámetro llamada limite o frontera. Esto lleva a que la probabilidad $p(x_j|c)$ se escriba de la siguiente manera:

$$p(x_j|c) = p(j|c)x_j + (1 - p(j|c))(1 - x_j) \tag{3.12}$$

$$p(j|c) = \sum_{i \in \mathbf{X}_c} I(x_{ij} = 1) \tag{3.13}$$

De nuevo $\mathbf{X}_c$ es el subconjunto de índices del conjunto de datos que pertenecen a la clase $c \in \mathbf{C}$.

3.1.6 Regresión logística (LR) [4]

El modelo de regresión logística utiliza el enfoque de *uno vs resto*, es decir que $c \in \{c_0,\ c_1\}$, lo que permite reescribir la probabilidad posterior del teorema de Bayes de la siguiente manera:

$$p(C_0|\vec{\mathbf{x}}) = \frac{p(\vec{\mathbf{x}}|C_0)p(C_0)}{p(\vec{\mathbf{x}}|C_0)p(C_0) + p(\vec{\mathbf{x}}|C_1)p(C_1)} = \frac{1}{1 + exp(-a)} = \sigma(a) \tag{3.14}$$

Donde $\sigma(a)$ se define como la ***función sigmoide logística*** y el argumento a es:

$$a = Ln\ \frac{p(\vec{\mathbf{x}}|C_0)p(C_0)}{p(\vec{\mathbf{x}}|C_1)p(C_1)} \tag{3.15}$$

El hecho de escribir la probabilidad de obtener una clase en términos de $\sigma(a)$ y cambiar la forma de a por una función lineal $W^\top \vec{\mathbf{x}}$, es lo que se le conoce como modelo de ***regresión logística***. El hecho de utilizar el enfoque *uno vs resto* es que este modelo construye diez funciones de probabilidad, una para cada clase de la forma:

$$p(c|\vec{\mathbf{x}}) = \frac{1}{1 + exp\left(-W^\top \vec{\mathbf{x}}\right)} \tag{3.16}$$

De modo que teniendo el conjunto de datos de entrenamiento $\mathbf{X}$, $\mathbf{Y}$ y el conjunto de etiquetas $C = \{0,\ 1\}$, la función de verosimilitud se escribe como:

$$p(\mathbf{X}, \mathbf{Y}|W) = \prod_{i=1}^{60\,000} \sigma\left(W^\top \vec{\mathbf{x}}_i\right)^{\mathbf{y}_i} \left(1 - \sigma\left(W^\top \vec{\mathbf{x}}_i\right)\right)^{1-\mathbf{y}_i} \tag{3.17}$$

Tomando el logaritmo negativo de la función 3.17 se define la ***función de error entropía-cruzada***:

$$E(W) = -Ln\, p\left(\mathbf{X}, \mathbf{Y}|W\right) = -\sum_{i=1}^{60\,000} \left\{\mathbf{y}_i Ln\, \sigma\left(W^\top \vec{\mathbf{x}}_i\right) + (1-\mathbf{y}_i) Ln\left(1-\sigma\left(W^\top \vec{\mathbf{x}}_i\right)\right)\right\} \tag{3.18}$$

Es claro que este modelo es diferente del resto, por la aparición de los parámetros $W^\top$ los cuales se optimizan para minimizar la función de error entropía-cruzada. Dado un vector de prueba $\vec{\mathbf{x}}$ se le asigna la clase c que maximice la función de probabilidad:

$$\hat{\mathbf{y}(\mathbf{x})} = argmax_c\left(p(c|\vec{\mathbf{x}})\right) \tag{3.19}$$

3.2 Vecinos más cercanos

Los modelos de vecinos más cercanos se basan en el principio de encontrar un subconjunto de los datos de entrenamiento cercanos en base a una métrica a un dato de prueba $\vec{\mathbf{x}}$, al cual se le desea asignar una clase. En base a lo anterior los modelos de vecinos más cercanos guardan el conjunto de datos $\mathbf{X}$ en una estructura de datos, que permite encontrar el subconjunto de datos de $\mathbf{X}$ que están a lo mucho a una distancia límite de un dato de prueba $\vec{\mathbf{x}}$ en base a una métrica de distancia. Estos modelos de vecinos más cercanos son conocidos como ***modelos de aprendizaje maquina no generalistas***, debido a que estos modelos simplemente memorizan todos los datos de entrenamiento y lo importante es definir la **distancia límite**. Debido al alcance de este trabajo se presentan dos formas diferentes de definir la distancia límite: centroide más cercano y **k**-vecinos más cercanos.

3.2.1 k-vecinos más cercanos (KNN)

Dado un entero positivo k y un dato de prueba $\vec{\mathbf{x}}$, el modelo k-vecinos más cercanos (KNN) identifica los k vectores de $\mathbf{X}$ más cercanos a $\vec{\mathbf{x}}$ y los representa por $N_{k,\vec{\mathbf{x}}}$, asignandole a $\vec{\mathbf{x}}$ la clase c que tenga el mayor número de elementos en $N_{k,\vec{\mathbf{x}}}$:

$$\hat{\mathbf{y}}(\vec{\mathbf{x}}) = argmax_{c \in \mathbf{C}} \left(\frac{1}{k} \sum_{i \in N_{k,\vec{\mathbf{x}}}} I(y_i = c) \right) \tag{3.20}$$

3.2.2 Centroide contraído más cercano (NC) [27]

Teniendo el conjunto de datos $\mathbf{X}$ se definen los centroides de cada clase de la siguiente forma:

$$\vec{\mu}_c = \frac{1}{|\mathbf{X}_c|} \sum_{i \in \mathbf{X}_c} \vec{\mathbf{x}}_i \tag{3.21}$$

Donde $\mathbf{X}_c$ es el subconjunto de índices del conjunto de datos $\mathbf{X}$ que pertenecen a la clase $c \in \mathbf{C}$. El entrenamiento del modelo de centroide más cercano consiste en obtener el centroide de cada clase y la regla de clasificación para un dato de prueba $\vec{\mathbf{x}}$ es asignarle la clase c cuyo centroide $\vec{\mu}_c$ es el más cercano:

$$\hat{\mathbf{y}}(\vec{\mathbf{x}}) = argmin_{c \in \mathbf{C}} ||\vec{\mu}_c - \vec{\mathbf{x}}|| \tag{3.22}$$

Una extensión al modelo de centroide más cercano es utilizar un parámetro llamado ***contracción*** denotado por α, el cual permite remover componentes de los centroides que no brindan información relevante tales como ruido. Denotando la componente j del centroide de todo el conjunto de datos $\mathbf{X}$ como μ^j y la componente j del centroide de la clase c como μ_{jc} se tienen las siguientes ecuaciones:

$$\mu^j = \frac{1}{N} \sum_{i=1}^{60\,000} x_{ij} \tag{3.23}$$

$$\mu_{jc} = \frac{1}{|\mathbf{X}_c|} \sum_{i \in \mathbf{X}_c} x_{ij} \tag{3.24}$$

La idea de α es llevar los centroides de clase hacia el centroide del conjunto de datos. Para ello se hace uso de un estadístico-t denotado por d_{jc}:

$$\mu_{jc} = \mu^j + m_c \cdot s_j \cdot d_{jc} \tag{3.25}$$

$$s_j^2 = \frac{1}{N - L} \sum_{c=1}^{10} \sum_{i \in \mathbf{X}_c} \left(x_{ij} - \mu^j \right)^2 \tag{3.26}$$

$$m_c = \sqrt{1/|\mathbf{X}_c| - 1/N} \tag{3.27}$$

Donde s_j es la desviación estándar agrupada dentro de la clase para la componente j. De modo que al restar una cantidad α a d_{jc} se puede llevar el centroide de la clase c hacia el centroide del conjunto de datos, a tal grado que si $d_{jc} - \alpha < 0$ se establece $\mu_{jc} = \mu^j$:

$$\mu_{jc} = \mu^j + m_c \cdot s_j \left(d_{jc} - \alpha \right) \tag{3.28}$$

3.3 Modelos de función discriminante

Los modelos de función discriminante son aquellos donde se asume que la regla de asignación de clase está dada por una función paramétrica, la cual mapea cada $\vec{\mathbf{x}}$ a una región del espacio que tiene asociada una etiqueta de clase.

3.3.1 Ridge [10]

El ***clasificador Ridge*** es un modelo lineal regularizado cuya función a minimizar consta de dos términos. El primer término de la función a minimizar es la ***suma de la función error cuadrada*** (RSS siglas en ingles), la cual proviene de la formulación de un modelo lineal. Por otro lado el segundo término es la suma de los cuadrados de los parámetros desconocidos $\mathbf{w}$ del modelo, llamado ***regularización***, con el objetivo de reducir los valores de $\mathbf{w}$, de modo que la función error a minimizar en el modelo Ridge es:

$$E(w) = \sum_{i=1}^{60\,000} \left(\mathbf{y}_i - w_0 + \sum_{j=1}^{784} w_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{784} w_j^2 \tag{3.29}$$

Donde $\lambda \geq 0$ es un parámetro de ***penalización*** a determinar por separado y sirve para controlar el impacto relativo de los dos términos que aparecen en la ecuación 3.29 al momento de estimar los valores de los parámetros desconocidos $\mathbf{w}$. Para terminar, se observa que la penalización por reducción es aplicado a todos los parámetros desconocidos excepto a w_0, porque solo se quiere reducir la estimación asociada a cada una de las componentes de los datos.

3.3.2 Máquinas de soporte vectorial (SVM) [4]

Las ***máquinas de soporte vectorial*** (SVM siglas en inglés) proponen utilizar un hiperplano llamado ***hiperplano de clasificación*** dado por 3.30 para dividir el espacio donde vive el conjunto de datos $\mathbf{X}$ en dos regiones, donde $\mathbf{X}_1$ es el subconjunto de índices del conjunto de datos que están en la región de la clase 1 y $\mathbf{X}_2$ es el subconjunto de índices del conjunto de datos que están en la región de la clase 2.

$$\mathbf{h}(\vec{\mathbf{x}}) = w_0 + \sum_{j=1}^{784} w_j \cdot x_j \tag{3.30}$$

Donde los parámetros desconocidos $\mathbf{w}$ del modelo representan el vector normal al hiperplano de clasificación. Suponiendo que se tienen dos clases y se asignan los valores de etiqueta $C_1 = 1$ y $C_2 = -1$ para la clase 1 y 2 respectivamente. El método de SVM hace uso de dos hiperplanos paralelos al hiperplano de clasificación llamadas ***hiperplanos de soporte*** para definir un ***margen de separación*** acotado por estos, de modo que los hiperplanos de soporte definen la región del espacio donde se encuentran los elementos de una clase:

$$h_1 = w_0 + \sum_{j=1}^{784} w_j \cdot x_{ij} \geq 1, \quad i \in \mathbf{X}_1 \tag{3.31}$$

$$h_2 = w_0 + \sum_{j=1}^{784} w_j \cdot x_{ij} \leq -1, \quad i \in \mathbf{X}_2 \tag{3.32}$$

$$\mathbf{y}_i(w_0 + \sum_{j=1}^{784} w_j \cdot x_{ij}) \geq 1, \quad i \in \mathbf{X} \tag{3.33}$$

Se observa que la ecuación 3.33 es una restricción que se debe cumplir para todo dato. Sin embargo, este modelo obliga a que todos los datos de la clase 1 estén en la región h_1 y que todos los datos de la clase 2 estén en la región h_2. Para modelar un conjunto de datos donde se permite que datos que pertenecen a la clase C_1 estén fuera de la región h_1 y viceversa para la clase 2, se introduce la distancia entre el vector de entrada y su hiperplano de soporte definida de la siguiente manera:

$$\xi_i = |\mathbf{y}_i - w_0 + \sum_{j=1}^{784} w_j \cdot x_{ij}|, \quad i \in \{\mathbf{X}_{c_1} - \mathbf{X}_1\} \cup \{\mathbf{X}_{c_2} - \mathbf{X}_2\} \tag{3.34}$$

$$\xi_i = 0, \quad i \in \{\mathbf{X}_{c_1} \cap \mathbf{X}_1\} \cup \{\mathbf{X}_{c_2} \cap \mathbf{X}_2\} \tag{3.35}$$

Donde $\mathbf{X}_{c_1}$ es el subconjunto de índices del conjunto de entrenamiento $\mathbf{X}$ que pertenecen a la clase 1 y $\mathbf{X}_{c_2}$ es el subconjunto de índices del conjunto de entrenamiento $\mathbf{X}$ que pertenecen a la clase 2. Esto lleva a que es necesario agregar la nueva

variable ξ_j a la ecuación 3.33 para que esta ecuación de restricción se cumpla para todos los datos:

$$\mathbf{y}_i(w_0 + \sum_{j=1}^{784} w_j \cdot x_{ij}) \geq 1 - \xi_i, \quad i \in \mathbf{X} \tag{3.36}$$

El modelo de SVM elige el hiperplano de clasificación que maximiza el margen de separación entre los hiperplanos de soporte y minimiza la distancia de los datos que están fuera de su región de clase, de modo que la función ha optimizar por SVM es:

$$MIN \left(C \sum_{i=1}^{60\,000} \xi_i + \frac{1}{2} \|(w_1, ..., w_{784})\|^2 \right) \tag{3.37}$$

Donde C es un parámetro de penalización. Haciendo uso de los multiplicadores de Lagrange se obtiene el problema dual, el cual se debe maximizar respecto a α_i:

$$MAX\ L(\alpha) = MAX \left(-\frac{1}{2} \sum_{i}^{60\,000} \sum_{j}^{60\,000} \alpha_i \alpha_j \mathbf{y}_i \mathbf{y}_j \vec{\mathbf{x}}_i \cdot \vec{\mathbf{x}}_j + \sum_{i}^{60\,000} \alpha_i \right) \tag{3.38}$$

Sujeto a:

$$0 \leq \alpha_i \leq C \tag{3.39}$$

$$\sum_{i=1}^{60\,000} \alpha_i \mathbf{y}_i = 0 \tag{3.40}$$

Se observa que en la ecuación 3.38 aparece el producto punto entre los vectores de entrada, por lo que es posible aplicar una transformada a los vectores de entrada $\mathbf{x}$, siempre y cuando el producto punto de la transformada este definido. De esta forma es posible separar los datos que no son separables con un simple hiperplano. Las transformaciones usadas comúnmente son las ***funciones Kernel***, por ello a

esta técnica se le conoce como truco Kernel, de modo que la ecuación 3.38 se puede escribir de la siguiente manera:

$$MAX\ L(\alpha) = MAX\left(-\frac{1}{2}\sum_{i}^{60\,000}\sum_{j}^{60\,000}\alpha_i\alpha_j\mathbf{y}_i\mathbf{y}_j K(\vec{\mathbf{x}}_i,\vec{\mathbf{x}}_j) + \sum_{i=1}^{60\,000}\alpha_i\right) \tag{3.41}$$

3.3.3 Perceptrón multicapa (MLP) [4]

La representación del modelo de red neuronal se hace utilizando un objeto llamado ***neurona***, y un conjunto de estas forman las llamadas ***capas***, las cuales dan la estructura de la red neuronal. La manera en que la red neuronal conecta una capa con la siguiente para transmitir la información es aplicando una transformación a la combinación líneas de la neuronas de la capa de entrada (vector de entrada) llamada ***función de activación***, y esta transformación se aplica para cada una de las neuronas de la capa siguiente (capa oculta), la cual está dada de la siguiente manera:

$$Z_n(\vec{\mathbf{x}}, W) = f(a_n) = f\left(\sum_{j=1}^{784} w_{nj}^{(0)} x_j + w_{n0}^{(0)}\right) \tag{3.42}$$

Donde $n = 1, ..., N$ (número de neuronas de la capa oculta) y el superíndice 0 indica que los parámetros correspondientes están en la capa de entrada de la red. Nos referiremos a los parámetros $w_{nj}^{(0)}$ como pesos y los parámetros $w_{n0}^{(0)}$ como sesgos. Se tiene entonces que cada una de estas trasformadas Z_n que son las salidas de la capa oculta, se convierten en la entrada de cada una de las neuronas de la ***capa de salida*** (siguiente capa después de la capa oculta). En esta etapa se vuelve aplicar el proceso anterior que consiste en aplicar una transformación a una combinación lineal, pero en esta ocasión de las variables Z_n, de la misma manera se aplica para cada una de las neuronas de la capa siguiente:

$$Y_m(\vec{\mathbf{x}}, W) = g(b_m) = g\left(\sum_{n=1}^{N} w_{mn}^{(1)} z_n + w_{m0}^{(1)}\right) \tag{3.43}$$

$$Y_m(\vec{\mathbf{x}}, W) = g\left(\sum_{n=1}^{N} w_{mn}^{(1)} f\left(\sum_{j=1}^{784} w_{nj}^{(0)} x_j + w_{n0}^{(0)}\right) + w_{m0}^{(1)}\right) \tag{3.44}$$

Donde $m = 1, ..., M$ (número de neuronas de la capa de salida), el superíndice 1 indica que parámetros corresponden a la primera capa oculta y nuevamente los parámetros $w_{mn}^{(1)}$ son los pesos y $w_{m0}^{(1)}$ los sesgos. Se tiene entonces que cada una de estas trasformadas Y_m son las salidas de la capa de salida, por lo tanto, las Y_m a utilizar deben ser las adecuadas al problema que se resuelve. En este caso un problema de clasificación multiclase estándar, en el que a cada entrada $\vec{\mathbf{x}}$ se le asigna solo una de las 10 clase mutuamente excluyentes, por lo tanto se tienen 10 neuronas de salida y la función de activación a utilizar es la siguiente:

$$Y_m(x, w) = \frac{exp(b_m(x, w))}{\sum_{l=1}^{10} exp(b_l(x, w))} \tag{3.45}$$

A esta ecuación se le conoce como ***softmax***, la cual satisface con $0 \leq Y_m \leq 1$ y $\sum_{m=1}^{10} Y_m = 1$. Por lo tanto, la ecuación 3.45 se interpreta como la probabilidad de que una entrada $\vec{\mathbf{x}}$ pertenezca a la clase c, lo que lleva a siguiendo la función de error:

$$E(W) = -\sum_{i=1}^{60\,000} \sum_{c=1}^{10} \mathbf{y}_c Ln\ Y_c(\vec{\mathbf{x}}_i, W) \tag{3.46}$$

3.4 Modelos no paramétricos

Los modelos no paramétricos son aquellos que utilizan un conjunto de condicionales para construir una estructura que permita separar las clases.

3.4.1 Árbol de decisión (DT) [6][20]

El clasificador con estructura de árbol es construido al particionar subconjuntos de $\mathbf{X}$ en dos subconjuntos descendientes. El objetivo de las particiones es obligar a que los datos que pertenecen a una clase sean dirigidos a uno de los subconjuntos descendientes y los datos que pertenecen a la otra clase sean dirigidos al otro subconjunto descendiente, este proceso de partición comienza con el mismo $\mathbf{X}$. Se identifica cada uno de los nodos con la variable t_m, donde el subíndice m es para diferenciarlo del resto. Se define la proporción del nodo t_m como $p(c|t_m)$, la cual indica la proporción de casos $\vec{\mathbf{x}} \in t_m$ que pertenecen a la clase c, donde $c \in \mathbf{C}$ y $p(1|t_m) + \ldots + p(L|t_m) = 1$.

Para particionar un nodo t_m se genera un conjunto de separadores $\mathbf{S}$ de la forma $s = (x_i, l_m)$ que consiste en la componente i del vector de entrada y un límite l_m, de tal modo que los dos subconjuntos descendientes de la partición del nodo t_m se definen como:

$$t_I(s(x_i, l_m)) = \{\mathbf{x} \in t_m : x_i \leq l_m\} \tag{3.47}$$

$$t_D(s(x_i, l_m)) = \{\mathbf{x} \in t_m : x_i > l_m\} \tag{3.48}$$

Para decir que partición $s \in \mathbf{S}$ se utilizara para seguir expandiendo el árbol, primero es necesario definir una función de impureza, la cual se puede interpretar como una medida de calidad para separar las clases presentes en t_m en cada uno de los subconjuntos descendientes y que dependen de las proporciones definidas anteriormente. A continuación, se presentan las dos funciones de impureza más utilizadas en la literatura, la primera de ellas llamada ***Gini***:

$$H(t_m) = \sum_{c=1}^{L} p(c|t_m)(1 - p(c|t_m)) \tag{3.49}$$

Y la segunda llamada ***Entropía***:

$$H(t_m) = -\sum_{c=1}^{L} p(c|t_m) log(p(c|t_m)) \tag{3.50}$$

Se define la impureza de la partición de la siguiente manera:

$$G(t_m, s(x_i|l_m)) = \frac{n_I}{N_m} H(t_I(s(x_i, l_m))) + \frac{n_D}{N_m} H(t_D(s(x_i, l_m))) \tag{3.51}$$

Donde N_m es número datos presentes en t_m, n_I es el número de datos que se presentes en t_I y n_D es el número de datos presentes en t_D. El objetivo de tener un conjunto de separadores $\mathbf{S}$ es para seleccionar aquel que minimiza la función de impureza de partición:

$$s^*(x_i, l_m) = argmin_s \, G(t_m, s(x_i, l_m)) \tag{3.52}$$

De modo que esta nueva partición será parte del árbol si y solo si se cumple la siguiente condición:

$$G^*(t_m, s^*(x_i, l_m)) \leq H(t_m) \tag{3.53}$$

En caso de cumplirse la condición el proceso de partición se vuelve aplicar a cada uno de los nodos t_I y t_D. En caso de no cumplirse la condición anterior para cualquier $s(x_i, l_m)$ el nodo t_m se define como ***nodo terminal*** y se le asigna la etiqueta de clase que tenga la mayor proporción, es decir se le asigna la clase $\hat{c}$ que cumple con:

$$\hat{\mathbf{y}} = argmax_c p(c|t_m) \tag{3.54}$$

De esta forma todo aquel vector de prueba un clasificador de árbol le asigna la clase que tiene asignado el nodo terminal en el cual cae.

3.4.2 Bosque aleatorio (RF) [5]

Consiste en la generación de un número D de árboles de decisión $A(\mathbf{X}_i)$, cuyo conjunto de entrenamiento $\mathbf{X}_1, \ldots, \mathbf{X}_D$ para cada árbol son formados al tomar elementos del conjunto de entrenamiento original de forma aleatoria hasta tener el mismo tamaño que el conjunto de entrenamiento original. La técnica para formar los conjuntos de entrenamiento descrita se le conoce como ***boostrap***, la cual permite que en el conjunto de datos generado se tengan elementos repetidos y que haya elementos del conjunto de entrenamiento original que no fueron seleccionados. Por otra parte, el conjunto de separadores $\mathbf{S}$ se restringe al poder utilizar solo una cantidad de componentes del vector de entrada y en cada nodo cambias las componentes que son posibles de usar. De este modo la clase que se le asigna a un vector de prueba es aquella que tenga el mayor número de votos al pasar el vector de prueba por todos los arboles generados.

$$\hat{\mathbf{y}}(\vec{\mathbf{x}}) = argmax_c \sum_{i=1}^{D} I(A(\mathbf{x}|\mathbf{X}_i) = c) \tag{3.55}$$

3.4.3 AdaBoost [32]

El algoritmo de AdaBoost es un procedimiento iterativo que en cada iteración entrena un clasificador débil sobre una versión de los datos modificada. El caso de clasificador débil mas utilizado es un clasificador de árbol con una profundidad

máxima de uno, en teoría cualquier algoritmo de clasificación cuyo rendimiento sea ligeramente superior que una elección aleatoria es considerado clasificador débil. A continuación, se presenta un pseudocódigo del algoritmo AdaBoost y después se describe cada una de sus etapas.

Capítulo 4

Experimentación

Para obtener las curvas de aprendizaje fue necesario establecer los tamaños de los conjuntos de entrenamiento y para simplificar la tarea se decidió tener los tamaños porcentuales 10, 20, 30, 40, 50, 60, 70, 80 y 90 del conjunto de datos de entrenamiento de la MNIST. Para realizar un análisis estadístico para cada uno de los tamaños establecidos anteriormente se tomaron diez muestras de forma aleatoria y balanceada para cada digito, obteniendo un total de 90 muestras las cuales fueron guardadas.

Con la finalidad de conseguir el comportamiento de los modelos se desarrolló un diseño de experimentos factorial para cada modelo, de modo que cada una de las 90 muestras se pasaron por estos diseños de experimentos y los errores de entrenamiento, errores de prueba y tiempos de entrenamiento fueron guardados en un archivo .csv para su posterior procesamiento.

Recursos de software:

- Ubuntu 14.04.5 LTS
- Conda 4.6.14

- Python 3.6.3
- Numpy 1.15.4
- Pandas 0.20.3
- Matplotlib 2.1.0

Recursos de hardware:

- Intel(R) Xeon(R) E3-1240 v3 3.40GHz
- 16 GB RAM
- 4 Núcleos, 2 hilos por núcleo

A continuación, se muestran los diseños de experimentos para cada modelo, donde los niveles de cada factor se establecieron en base a los encontrados en el estado del arte.

4.1 Centroide contraído más cercano (NC)

En este modelo el único parámetro variable es el de contracción α. Debido a que es un parámetro que resta a un estadístico-t los valores posibles que puede tomar α deben estar entre 0 y 1. Para este trabajo se eligió tomar los valores entre 0 y 0.5 con intervalos de 0.02, es decir:

Factor	Niveles
Contracción (α)	0.0, 0.02, 0.04, ..., 0.46, 0.48, 0.50

Tabla 4.1: Factores y niveles para el diseño de experimentos del modelo Centroide encogido más cercano.

4.2 Ridge

El modelo Ridge tiene como parámetros de ajuste el parámetro de penalización λ y la estrategia de solución. El parámetro λ puede tomar únicamente valores igual o mayor a 0, a continuación, se muestran los valores utilizados para ambos parámetros:

Factor	Niveles
Penalización (λ)	$10^{-4},\ 10^{-2}\ 10^{0},\ 10^{2},\ 10^{4}$
Estrategia	svd, cholesky, lsqr, sparse cg, sag, saga

Tabla 4.2: Factores y niveles para el diseño de experimentos del modelo Ridge.

4.3 Máquinas de soporte vectorial (SVM)

Para el modelo de SVM se tiene el parámetro de penalización C, la función Kernel $K(x_i, x_j)$, además de los parámetros de cada Kernel. Para este trabajo se utilizaron los Kernes lineal, polinomial, sigmoidal y radial, por ello son necesarios los parámetros gamma y grado, a continuación, se resumen los niveles de cada factor:

Factor	Niveles
Penalización (C)	1, 5, 10, 15, 20, 25
Kernel	linear, poly, rbf, sigmoid
Grado	1, 2, 3, 4, 5
Gamma	$10^{-5},\ 10^{-4},\ 10^{-3},\ 10^{-2},\ 10^{-1},\ 10^{0}$

Tabla 4.3: Factores y niveles para el diseño de experimentos del modelo SVM.

4.4 Perceptrón multicapa (MLP)

El modelo de perceptrón multicapa (MLP) tiene los factores de número de capas, número de neuronas, función de activación, penalización α, estrategia de solución y aquellos factores que conlleva utilizar cada estrategia de solución, a continuación, se presentan los niveles de cada factor:

Factor	Niveles	sgd	adam	lbfgs
Capas	1, 4, 7, 10	•	•	•
Neuronas	30, 90, 150, 210	•	•	•
Activación	tanh, sigmoid	•	•	•
Penalización α	10^{-4}, 10^{0}, 10^{4}	•	•	•

Tabla 4.4: Factores y niveles para el diseño de experimentos del modelo MLP.

4.5 Bayes ingenuo gaussiano (GNB)

En este modelo el único parámetro que se puede ajustar es el parámetro de suavizado α de la varianza más grande correspondiente a una componente y clase a cada una de las σ_{ic}^2, en la tabla 4.5 se encuentran los niveles correspondientes a este factor.

Factor	Niveles
Suavizado (α)	10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}

Tabla 4.5: Factores y niveles para el diseño de experimentos del modelo Bayes ingenuo gaussiano.

4.6 Bayes ingenuo multinomial (MNB)

Este modelo al igual que el Bayesiano ingenuo gaussiano tiene un parámetro de suavizado α que ayuda a evitar probabilidades con valor cero, en la tabla 4.6 se presentan los niveles correspondientes.

Factor	Niveles
Suavizado (α)	10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}

Tabla 4.6: Factores y niveles para el diseño de experimentos del modelo Bayes ingenuo multinomial.

4.7 Bayes ingenuo binomial (BNB)

En este modelo se presentan el par,ametro de suavizado pero además tiene el parámetro de binarización, debido a que el valor máximo que puede tener cualquier componente es 256, esto marca un límite superior para binarizar los vectores, en la tabla 4.7 se muestran los nieves propuestos para ambos parámetros.

Factor	Niveles
Suavizado (α)	10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}
Binarización	25.6, 51.2, 76.8, 102.4, 128.0, 153.6, 179.2, 204.8, 230.4

Tabla 4.7: Factores y niveles para el diseño de experimentos del modelo Bayes ingenuo binomial.

4.8 AdaBoost

Debido a que el modelo de AdaBoost se basa en el entrenamiento de un número finito de clasificadores débiles (árbol de decisión con profundidad de uno), donde

todos los factores para construir estos clasificadores débiles ya están establecidos, de modo que el único factor libre es el número de clasificadores débiles a construir, llamado número de estimadores. En la tabla se 4.8 se especifican los niveles del factor número de estimadores.

Factor	Niveles
Número de estimadores	10, 20, 30, 40, 50, 60, 70, 80, 90, 100

Tabla 4.8: Factores y niveles para el diseño de experimentos del modelo AdaBoost.

4.9 Árbol de decisión (DT)

La construcción de un árbol de decisión requiere de los factores función de impureza, selección del separador s, profundidad máxima, el número mínimo de datos para que un nodo se pueda separar, el número mínimo de datos para que un nodo sea nombrado terminal, el número de características a considerar cuando se construyen los separadores, el número máximo de nodos terminales y la reducción de impureza mínima en un nodo para que sea particionado.

Con el fin de obtener los mejores resultados se realizaron algunas pruebas preliminares para obtener los tiempos cuando se deja que el árbol de decisión crezca hasta no poder disminuir la función de impureza utilizando todo el conjunto de entrenamiento, esto quiere decir que se seleccione el separador que más reduce la impureza, que no se establezca límite para el factor de profundidad máxima, que un nodo se pueda particional a partir de tener dos datos, que un nodo que contenga un solo dato pueda ser considerado como nodo terminal, que el número máximo de nodos terminales sea ilimitado, que la reducción de impureza para poder particionar un nodo sea cero y que se consideren todas las características para construir los separadores. Teniendo en cuenta las consideraciones anteriores el único factor que se

podía variar era la función para calcular la impureza y el tiempo promedio obtenido fue de 12.08 segundos para la función gini y 16.51 segundos para la función entropía, tomando en cuenta que se utilizó todo el conjunto de entrenamiento. Debido a que el tiempo necesario para llevar acabo un diseño de experimentos con la configuración anterior no es tardado, se decidió que el único factor a tomar en cuenta fuera la función de impureza, ver tabla 4.9.

Factor	Niveles
Función de impureza	Gini y Entropía

Tabla 4.9: Factores y niveles para el diseño de experimentos del modelo árbol de decisión.

4.10 k-vecinos más cercanos (KNN)

En un modelo de k-vecinos más cercanos los factores son el número de vecinos k que se tomaran en cuenta para clasificar un dato de prueba, la métrica a utilizar para medir distancia y el tipo de influencia de los vecinos. En la tabla 4.10 se especifican los niveles de cada factor.

Factor	Niveles
k	1, 3, 5, 7
Métrica	Manhattan y Euclidiana
Peso	uniforme y distancia

Tabla 4.10: Factores y niveles para el diseño de experimentos del modelo k-vecinos más cercanos.

El factor peso indica si la aportación de cada vecino para clasificar un dato de prueba será solo por conteo (uniforme) o será proporcional a la distancia.

4.11 Análisis de discriminante lineal (LDA)

En este caso se tienen dos factores, el primero de ellos es la estrategia a utilizar para estimar los parámetros de medias y matriz de covarianza y el segundo de ellos el cual es opcional porque no es una variable propia del modelo, sin embargo es una herramienta utilizada para mejorar la estimación de la matriz de covarianza llamado contracción (shrinkage) de modo que los únicos niveles para el factor contracción es si se usa o no, ver tabla 4.11

Factor	Niveles
Estrategia	svd, lsqr y eigen
Contracción	None y auto

Tabla 4.11: Factores y niveles para el diseño de experimentos del modelo análisis de discriminante lineal.

4.12 Análisis de discriminante cuadrático (QDA)

En este caso no se tiene ningún factor debido a que las estimaciones de las medias y las matrices de covarianza para cada clase se obtienen de forma directa haciendo los cálculos sobre los datos de entrenamiento.

4.13 Regresión logística (LR)

En este modelo si se tiene explícitamente una función de error que depende de los parámetros de la función lineal con la que se aproxima el argumento de función sigmoide logística, de modo que el objetivo es encontrar la combinación de estos

parámetros que minimizan la función de error. Al igual que otros modelos donde se especifica una función de error la estrategia que más común es utilizar una aproximación del gradiente para encontrar el punto que minimiza la función error, por ello el uno de los factores de este modelo es la estrategia para aproximar el gradiente. Existe un segundo factor impuesto por la librería SK-Learn, el cual es una constante que multiplica a la función de error que se puede interpretar como un término de penalización al igual como aparece en el modelo de máquina de soporte vectorial. La tabla 4.12 muestra los niveles de cada factor.

Factor	Niveles
Estrategia	lbfgs, sgd y adam
Penalización	10^{-5}, 10^{-3}, 10^{-1}, 10^{1}, 10^{3}, 10^{5}

Tabla 4.12: Factores y niveles para el diseño de experimentos del modelo regresión logística.

4.14 Bosque aleatorio (RF)

Como su nombre lo indica este modelo construye un número fijo de árboles de decisión. En base a los resultados preliminares obtenidos con el modelo de árbol de decisión se declaro que los arboles crecieran de forma indefinida hasta no poder reducir la función de impureza. Esto deja que los factores del modelo sean la función de entropía, el numero de arboles a construir y el número de características que es posible utilizar para construir los separadores, en la tabla 4.13 se muestran los niveles de cada factor.

Factor	Niveles
Función de impureza	Gini y Entropía
Número de árboles	10, 20, 30, 40 y 50
Número de características	sqrt, log2 y None

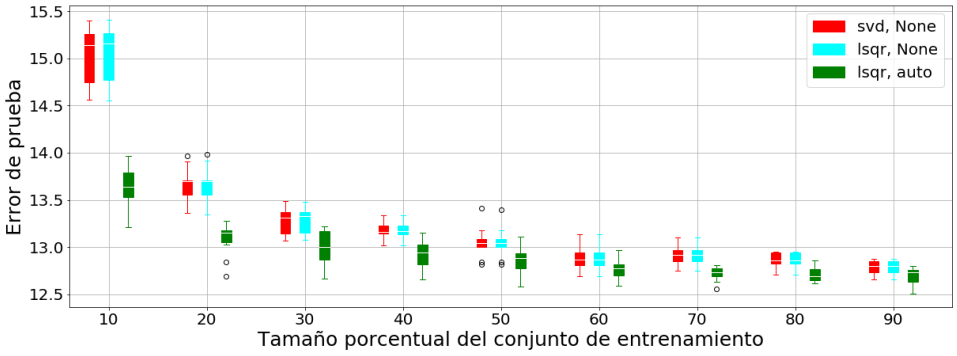
Tabla 4.13: Factores y niveles para el diseño de experimentos del modelo bosque aleatorio.

Capítulo 5

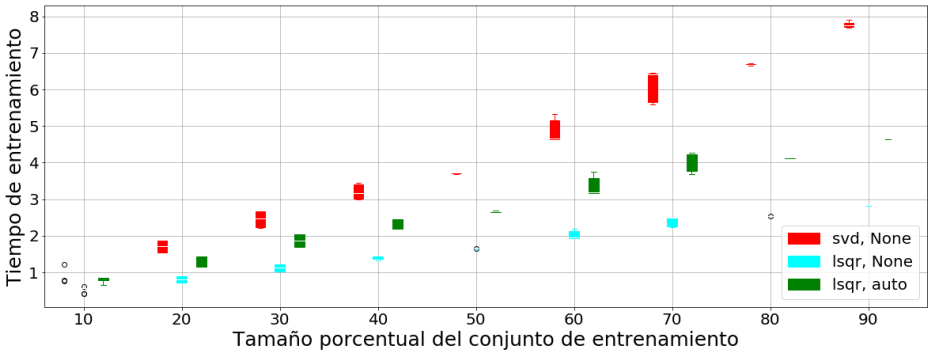
Resultados

5.1 Análisis de discriminante lineal

Las curvas de aprendizaje sobre el conjunto de prueba para el modelo de análisis discriminante lineal están en la figura 5.1 inciso (a). En este caso se tienen tres combinaciones que surgen de la combinación de los niveles de los factores *estrategia* y *contracción* (shrinkage), la estrategia svd aparece ella sola porque no es posible implementar la contracción con esta estrategia. En todas las combinaciones el error de prueba presenta un comportamiento en forma de exponencial negativa en función del tamaño del conjunto de entrenamiento. En todos los niveles del factor *tamaño del conjunto de entrenamiento* la combinación de la estrategia lsqr con contracción (shrinkage) tiene los errores de prueba más bajos, teniendo diferencia significativa con el resto de combinaciones, aunque esta diferencia disminuye con el aumento del conjunto de entrenamiento y el utilizar ambas estrategias sin contracción no presentan diferencias significativas. En la figura 5.1 inciso (b) se muestran los tiempos de entrenamiento en función del tamaño del conjunto de entrenamiento, los cuales presenta un comportamiento lineal y es claro que hay diferencias significativas las cuales aumentan a medida que se entrena con más datos. También es posible observar el efecto de utilizar la *contracción* con la estrategia *lsqr* al incrementar el tiempo de entrenamiento si no se usara, aunque no es más tardado que utilizar *svd*.



(a) Curvas de aprendizaje sobre el conjunto de prueba

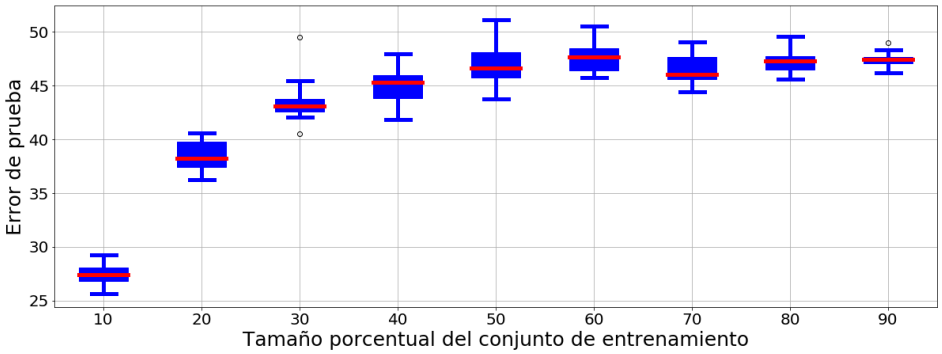


(b) Tiempos para entrenar el modelo en función del tamaño del conjunto de entrenamiento

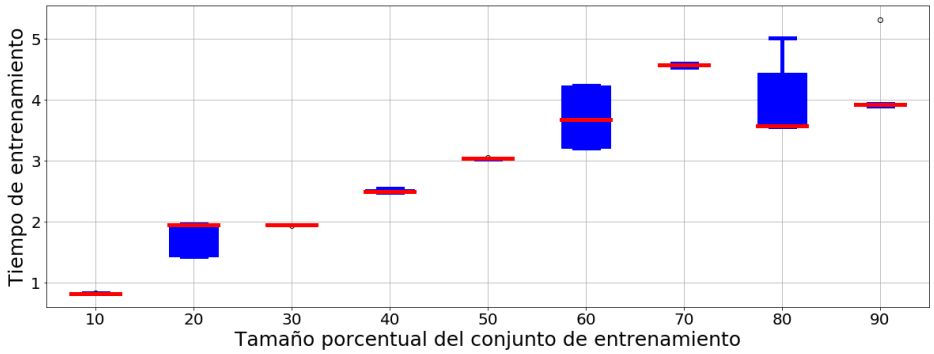
Figura 5.1: Resultados del modelo análisis discriminante lineal

5.2 Análisis de discriminante cuadrático

La curva de aprendizaje sobre el conjunto de prueba para el modelo de análisis discriminante cuadrático se encuentra en el inciso (a) de la figura 6.1, en ella se puede observar que la curva tiene un comportamiento logarítmico en función del tamaño del conjunto de entrenamiento, presentando diferencias significativas solo para los tamaños menores a 50 % donde alcanza su asíntota superior. En cuanto a los tiempos de entrenamiento, figura 6.1 inciso (b), en función del tamaño del conjunto de entrenamiento presenta un comportamiento lineal con diferencias significativas a medida que se aumenta el conjunto de entrenamiento.



(a) Curvas de aprendizaje sobre el conjunto de prueba

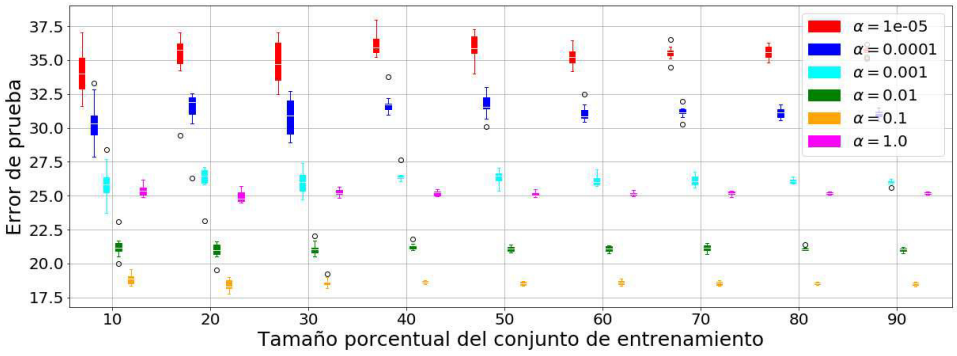


(b) Tiempos para entrenar el modelo en función del tamaño del conjunto de entrenamiento

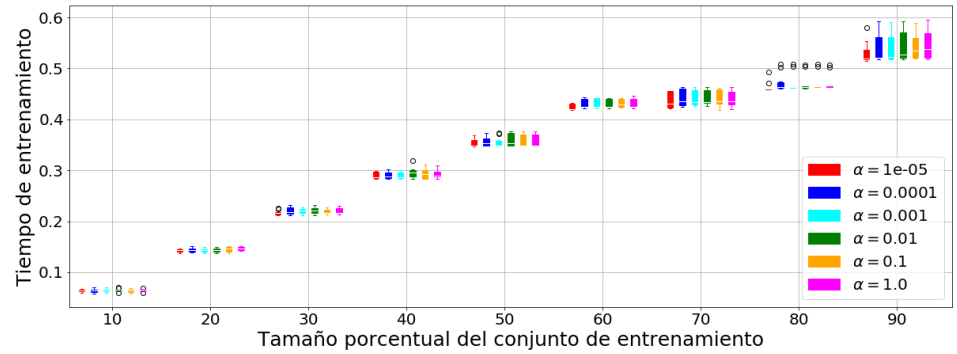
Figura 5.2: Resultados del modelo análisis discriminante cuadrático

5.3 Bayes ingenuo – Gaussiano

Las curvas de aprendizaje sobre el conjunte de prueba para el modelo de Bayes ingenuo gaussiano están en la figura 5.3 inciso (a). Cada una de las curvas corresponde con cada uno de los niveles del factor *suavizado*, en el caso de los niveles 0.00001, 0.0001, 0.001 y 0.01 no hay diferencias significativas en el error de prueba con el incremento del conjunto de datos para entrenar y cada uno de ellos genera un error de prueba diferente (entre ellos existen diferencias significativas. Para los niveles 0.1 y 1.0 del factor *suavizado* se tiene una mejora significativa en el error de prueba al pasar del nivel 10 al 20 del factor tamaño del conjunto de entrenamiento, sin embargo, no se presentan diferencias significativas para los siguientes



(a) Curvas de aprendizaje sobre el conjunto de prueba



(b) Tiempos para entrenar el modelo en función del tamaño del conjunto de entrenamiento

Figura 5.3: Resultados del modelo bayes ingenuo gaussiano

niveles del factor *tamaño del conjunto de entrenamiento*. En caso de mantener fijo el factor *tamaño del conjunto de entrenamiento* en cualquiera de sus niveles, el error de prueba disminuye a medida que se aumenta el nivel del factor *suavizado* hasta tener un mínimo con 0.1 y después empieza a crecer. Los tiempos de entrenamiento se muestran en el inciso (b) de la figura 5.3, los cuales crecen de forma lineal con el conjunto de entrenamiento y presentan diferencias significativas. Por último, los niveles del factor *suavizado* no generan diferencias significativas a los tiempos de entrenamiento.

5.4 Bayes ingenuo – Multinomial

En el inciso (a) de la figura 5.4 se muestran las curvas de aprendizaje sobre el conjunto de prueba para el modelo Bayes ingenuo multinomial. A pesar de que a simple vista parece que el factor *suavizado* afecta al error de prueba con un comportamiento en forma de parábola cuando se tiene fijo el tamaño del conjunto de entrenamiento, las pruebas estadísticas señalan que no hay diferencias significativas entre los niveles del factor *suavizado*. Al realizar pruebas para observar si existen diferencias significativas en el error de prueba entre los niveles del factor *tamaño del conjunto de entrenamiento* se tiene que los niveles 10 y 30 son diferentes de los demás, de modo que el comportamiento de las curvas es disminuir-crecer-disminuir para después mantenerse constante. En el inciso (b) de la figura 5.4 se presentan los tiempos de entrenamiento, los cuales presentan un patrón de crecimiento lineal en función del tamaño del conjunto de entrenamiento, los cuales son significativamente diferentes y el factor *suavizado* no genera diferencias significativas a los tiempos de entrenamiento.

5.5 Bayes ingenuo – Bernoulli

En la figura 5.5 inciso (a) se muestran las curvas de aprendizaje y los tiempos de entrenamiento en el inciso (b). Lo que respecta al factor *binarización* el único nivel que marca una diferencia significativa es el 102.4 y entre el resto de niveles no se presentan diferencias significativas, además que el efecto del nivel 102.4 mejora el error de prueba por lo que la figura 5.5 son con este nivel de *binarización*. Volviendo al inciso (a) de la figura 5.5 se observa que el error de prueba presenta una diferencia significativa solo cuando se entrena con el 10 % de los datos, del 20 % en adelante no se presentan diferencias significativas. En el caso del factor *suavizado* se eliminaron los niveles del 1 al 10 000 porque no aportan información relevante

al comportamiento de las curvas de aprendizaje y entre los niveles restantes solo el nivel 0.1 genera diferencias significativas al error de prueba pero solo para algunos niveles del factor *tamaño del conjunto de entrenamiento* por lo que no es generalizable. En los tiempos de entrenamiento figura 5.5 inciso (b) se observa que estos crecen de forma lineal en función del tamaño del conjunto de entrenamiento y presentan diferencias significativas, mientras que los niveles del factor *suavizado* no producen una diferencia significativa a los tiempos de entrenamiento.

5.6 Regresión logística

En este modelo se tienen los factores *estrategia* y *penalización*, en el caso del factor *penalización* se realizó un análisis preliminar para determinar si todos los niveles establecidos provocaban una diferencia significativa y se encontró que los niveles 10, 1 000 y 100 000 no presentan diferencias significativas entre ellos, es decir que no provocan un cambio al error de prueba y no tienen una interacciones con los demás factores, por lo que se quedó solo con el nivel 10 de estos tres niveles. En la figura 5.6 se encuentras las curvas de aprendizaje para cada uno de los niveles del factor *estrategia* dejando el factor de *penalización* fijo a 0.00001 y 10 en los incisos (a) y (b) respectivamente, de estas dos graficas se puede observar que hay una interacción entre los factores *penalización* y *estrategia* la cual provoca diferencias significativas en el error de prueba para las diferentes combinaciones de estos dos factores pero desaparecen con al aumento del conjunto de entrenamiento, además es posible ver que el error de prueba decrece en forma de exponencial negativa en función del tamaño del conjunto de entrenamiento.

La figura 5.7 muestra los tiempos de entrenamiento dejando el factor de *penalización* fijo a 0.00001 y 10 en los incisos (a) y (b) respectivamente, donde se puede ver claramente la interacción entre los factores *penalización* y *estrategia* y que el tamaño del conjunto de entrenamiento afecta de forma lineal a los tiempos de entrenamiento.

En el caso de tener fijo el factor *estrategia* a uno de sus niveles, el comportamiento es diferente porque el factor *penalización* provocara diferencias significativas dependiendo del nivel en que este el factor *penalización*, por ejemplo para las estrategias lbfgs y saga el factor *penalización* no provoca diferencias significativas en los tiempos de entrenamiento, en caso de tener la estrategia sag solo el nivel 0.00001 del factor *penalización* produce una diferencias significativa del resto de niveles y en caso den tener la estrategia liblinear cada uno de los niveles del factor *penalización* tienen diferencias significativas.

5.7 Árbol de decisión

Las curvas de aprendizaje sobre el conjunto de prueba para el modelo de árbol de decisión están en la figura 5.8 inciso (a). Para ambos niveles de la *función de impureza*, es decir gini y entropía tienen el mismo patrón de disminuir el error de prueba en forma de exponencial negativa en función del *tamaño del conjunto de entrenamiento* y en todo momento existe una diferencia significativa, siendo la función de impureza entropía la que da los errores de prueba más bajos. En la figura 5.8 inciso (b) se presentan los tiempos de entrenamiento, los cuales tienen un patrón lineal en función del *tamaño de entrenamiento*, presentando diferencias significativas y se observa que con la función de impureza gini es proceso de entrenamiento es más rápido.

5.8 Bosque aleatorio

Al igual que los resultados del modelo análisis discriminante lineal se hace una combinación de los niveles de los factores *estrategia* y *número de características* con el objetivo de mostrar más información en una sola figura. En la figura 5.9 se ilustran las curvas de aprendizaje sobre el conjunto de prueba, en el inciso (a) se muestran

las curvas de aprendizaje cuando se tienen 10 *estimadores* y el inciso (b) cuando se tienen 50 *estimadores*. En todos los casos las curvas de aprendizaje presentan un comportamiento de exponencial negativa en función del tamaño del conjunto de entrenamiento presentando diferencias significativas para el error de prueba. Tanto en la grafica del inciso (a) como (b) se puede ver que cuando se tiene fijo el numero de características, la función de impureza no genera diferencia significativa en el error de prueba, salvo en algunas ocasiones, pero sin ser algo generalizable. Siguiendo con la comparación de las curvas de aprendizaje, se observa que el número de estimadores provoca diferencias significativas al error de prueba, de modo que a mayor cantidad de estimadores menor es el error de prueba y aumenta las diferencias entre los niveles del número de características.

En la figura 5.10 se presentan los tiempos de entrenamiento con 10 *estimadores* inciso (a) y 50 *estimadores* inciso (b), para que se pudieran apreciar mejor fue necesario establecer el eje del tiempo en escala logarítmica y quitar el nivel None, con esto es posible ver que los tiempos de entrenamiento crecen de forma lineal con el tamaño del conjunto de entrenamiento. La figura 5.10 muestra claramente que los factores *estimadores* y *número de características* tienen efecto significativo en los tiempos de entrenamiento de forma creciente, es decir mientras más grande sea el número de estimadores o el número de características más se tardara el algoritmo en entrenar. Para finalizar también es claro que hay diferencias significativas en los tiempos de entrenamiento para las diferentes funciones de impureza, siendo entropía más tardado que gini.

5.9 Adaboost

La figura 5.11 muestra las curvas de aprendizaje en el inciso (a) y los tiempos de entrenamiento en el inciso (b). Para las curvas de aprendizaje se tiene que el *tamaño del conjunto de entrenamiento* solo tiene una diferencia para el nivel 10 y

solo para los *estimadores* 70, 100, 130 y 160. En cambio, el factor *estimadores* si tiene un efecto significativo para todos sus niveles sin importar el *tamaño del conjunto de entrenamiento*. En cuanto a los tiempos de entrenamiento se ve claramente que tanto el factor *estimadores* y *tamaño del conjunto de entrenamiento* provocan diferencias significativas para cada uno de sus niveles, en cuanto al patrón de los tiempos se tiene que es creciente de forma lineal con el factor *tamaño del conjunto de entrenamiento*.

5.10 k-vecinos más cercanos

En la figura 5.12 se muestras las curvas de aprendizaje para las combinaciones de los factores *métrica* y *número de vecinos* utilizando la *estrategia* uniforme en el inciso (a) y la *estrategia* distancia en el inciso (b). La primera observación es que sin importar el resto de factores el factor *tamaño del conjunto de entrenamiento* afecta significativamente al error de prueba para todos sus niveles, además las curvas presentan forma de exponencial negativa, de modo que se llegara algún tamaño del conjunto de entrenamiento donde ya estas diferencias significativas desaparecerán. La segunda observación es que el factor *metrica* también afecta significativamente el error de prueba para sus dos niveles manhattan y euclidiana sin importar el resto de los factores. En cuanto al factor *número de vecinos* por lo general los niveles 1, 3 y 5 no muestran diferencias significativas cuando se tiene la *estrategia* uniforme mientras que los niveles 7 y 9 si muestran diferencias significativas, en caso de tener la *estrategia* distancia los niveles 3 y 5 del factor *número de vecinos* no presentan diferencias significativas y el resto de los niveles si presentan diferencias significativas. Por último, el factor *estrategia* solamente no genera diferencias significativas cuando se tiene el *número de vecinos* en 1.

5.11 Centroide contraído más cercano

En la figura 5.13 inciso (a) se presentan las curvas de aprendizaje sobre el conjunto de prueba para el modelo centroide más cercano, antes de presentar los resultados es importante señalar que solo se presenta la curva de aprendizaje para el nivel 0 del factor *contracción* porque es el único nivel que tiene diferencia significativa del resto además que el resto de niveles no tienen diferencias significativas entre ellos y empeoran el error en prueba. Retomando la curva de aprendizaje en 5.13 inciso (a) existe una diferencia significativa entre el nivel 10 y el resto de niveles del factor *tamaño de conjunto de entrenamiento* y a partir del nivel 20 hacia adelante visualmente parece tener tendencia negativa pero al menos en el rango que se tiene no presenta diferencias significativas. En el inciso (b) de la figura 5.13 se muestran los tiempos de entrenamiento, los cuales presentan un patrón lineal y tienen diferencias significativas como función del tamaño del conjunto de entrenamiento.

5.12 Ridge

En el inciso (a) de la figura 5.14 se muestran las curvas de aprendizaje sobre el conjunto de prueba, es importante señalar que no se hace referencia al factor *penalización* porque solo para las *estrategias* svd y cholesky en el nivel 10 del factor *tamaño del conjunto de entrenamiento* es que los niveles de factor *alpha* provocan una diferencia significativa en el error de prueba, por lo que se considera un caso especial que no puede ser generalizable para el resto de situaciones. En el caso del factor *estrategias* el nivel lsqr provoca una diferencia significativa en el error de prueba solo cuando el factor *tamaño del conjunto de entrenamiento* tiene los niveles 10, 40, 70 y 80, mientras que el resto de niveles del factor *estrategias* no provocan una diferencia significativa en el error de prueba. Para finalizar todas las curvas de aprendizaje tienen un patrón decreciente con una cota inferior, incluso entre los niveles 80 y 90 del factor *tamaño del conjunto de entrenamiento* no existe una diferencia

significativa en el error de prueba para ningún nivel del factor *estrategia*.

En el inciso (b) de la figura 5.14 se presentan los tiempos de entrenamiento en los cuales se observa claramente que el factor *tamaño del conjunto de entrenamiento* provoca diferencias significativas en cada uno de los niveles del factor *estrategia*. Por otra parte, los patrones de los tiempos de entrenamiento dependen mucho del nivel que toma el factor *estrategia*, en caso de tener el nivel svd y cholesky el factor *penalización* no provoca diferencias significativas en los tiempos de entrenamiento, en el nivel lsqr solo cuando el factor *tamaño del conjunto de entrenamiento* tiene el nivel 10 es que el factor *penalización* provoca diferencias significativas, en el nivel sparce-cg cuando el factor *tamaño del conjunto de entrenamiento* esta entre los niveles 10 y 50 es que el factor *alpha* provoca diferencias significativas, y en los niveles sag y saga sin importar el nivel del factor *tamaño del conjunto de entrenamiento* todos los niveles del factor *penalización* provocan diferencias significativas en el tiempo de entrenamiento.

5.13 Perceptrón multicapa

5.13.1 LBFGS

En la figura 5.15 se presenta las curvas de aprendizaje para la combinación de los niveles de los factores *función de activación* y *alpha* dejando el factor *neuronas* fijo en 30 para el inciso (a) y 210 para el inciso (b). En la figura 5.15 se puede observar que el *tamaño del conjunto de entrenamiento* genera diferencias significativas en el error de prueba, aunque estas disminuyen o desaparecen para los niveles más altos del factor *tamaño del conjunto de entrenamiento*. En cuanto a las diferencias significativas provocadas por la combinación de los factores *función de activación* y *alpha* están presentes entre todas las combinaciones cuando se está en los niveles 10 y

20 del factor *tamaño del conjunto de entrenamiento*, para los niveles arriba del 20 solo las combinaciones logistic-10000 y tanh-1000 presentan diferencias significativas en el error de pruebas del resto de combinaciones. Por último el factor *neuronas* genera diferencias significativas para cada una de las combinaciones del resto de factores, es decir no importa en que niveles estén el resto de los factores al cambiar entre dos niveles del factor *neuronas* el error de prueba tendrá una diferencia significativa, además existe una relación directa, mientras se aumente el número de neuronas el error de prueba disminuye.

5.14 Máquinas de soporte vectorial (SVM)

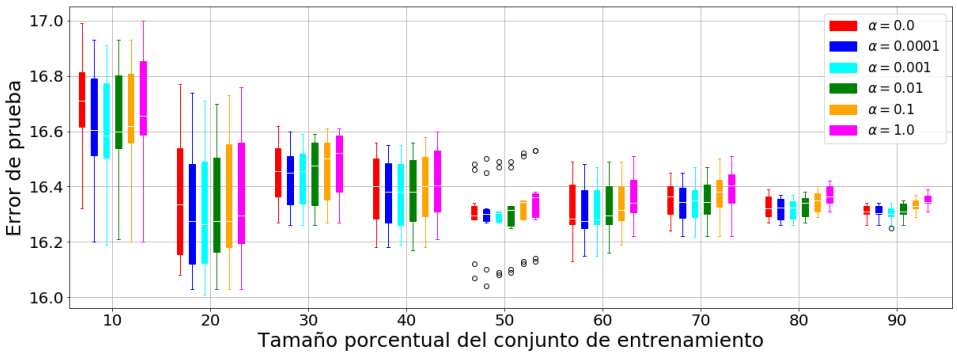
5.14.1 Kernel lineal

En la sección experimentación se mencionó que para el modelo de SVM con kernel lineal solo tiene un factor, la *penalización*, sin embargo, en la figura 5.16 inciso (a) e inciso (b) se puede apreciar que el factor *penalización* no genera diferencias significativas en los errores de prueba para las curvas de aprendizaje ni en los tiempos de entrenamiento. En las curvas de aprendizaje figura 5.16 inciso (a) se tiene un mínimo para el nivel 20 del factor *tamaño del conjunto de entrenamiento*, solo que en la parte creciente de la curva parece no tener un patón defino por el brindo entre los niveles 50 y 60, aunque entre los niveles 20-50 y 60-80 parece seguir un patrón lineal. En los tiempos de entrenamiento inciso (b) figura 5.16 sucede algo parecido a las curvas de aprendizaje, donde sigue un patrón lineal entre los niveles 10-50 y 60-80, teniendo un brinco entre los niveles 50 y 60 del factor *tamaño del conjunto de entrenamiento.*

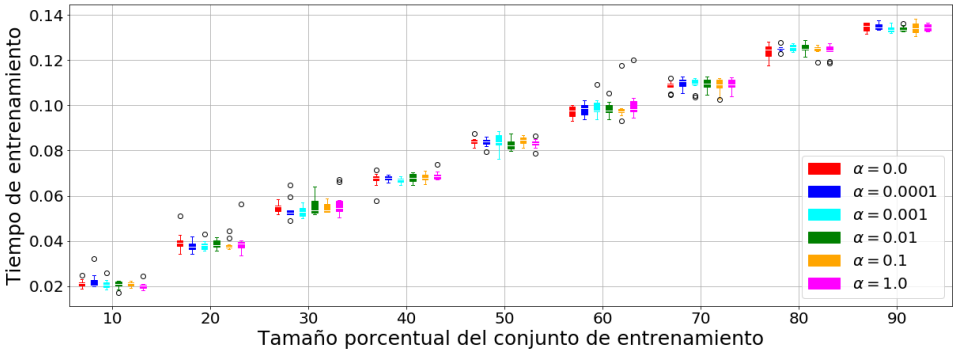
5.14.2 Kernel polinomial

Este modelo tiene tres factores *penalización*, *gamma* y *grado*, sin embargo, ninguno de los niveles de los factores *penalización* y *gamma* producen una diferencia significativa en el error de prueba ni en los tiempos de entrenamiento, por lo que se establecieron fijos *penalización* y *gamma* en 1 y 1 respectivamente. En el inciso (a) de la figura 5.17 se encuentran las curvas de aprendizaje sobre el conjunto de prueba para los niveles del factor *grado*, en ella se puede observar que presentan un patrón muy característico al decrecer significativamente el error de prueba hasta el nivel 70 del factor *tamaño del conjunto de entrenamiento*, por lo que es posible ajustar una función exponencial negativa. También en las curvas de aprendizaje se muestra el efecto que tienen los niveles del factor *grado*, donde cada uno de ellos provoca una diferencia significativa al error de prueba para todos los niveles del factor *tamaño del conjunto de entramiento*, aunque las diferencias se van haciendo más pequeñas a medida que se incrementa el *tamaño del conjunto de entrenamiento*.

En el inciso (b) de la figura 5.17 se muestras los tiempos de entrenamiento, en los cuales se puede observar diferencias significativas para ambos factores *grado* y *tamaño del conjunto de entrenamiento*, además que los tiempos de entrenamiento aumentan directamente con el aumento del *grado* del polinomio o el *tamaño del conjunto de entrenamiento*. En cuanto al patrón de los tiempos de entrenamiento parecen ajustarse a una línea recta, aunque se vuelve a presentar una especie de salto entre los niveles 50 y 60 del factor *tamaño del conjunto de entrenamiento*.

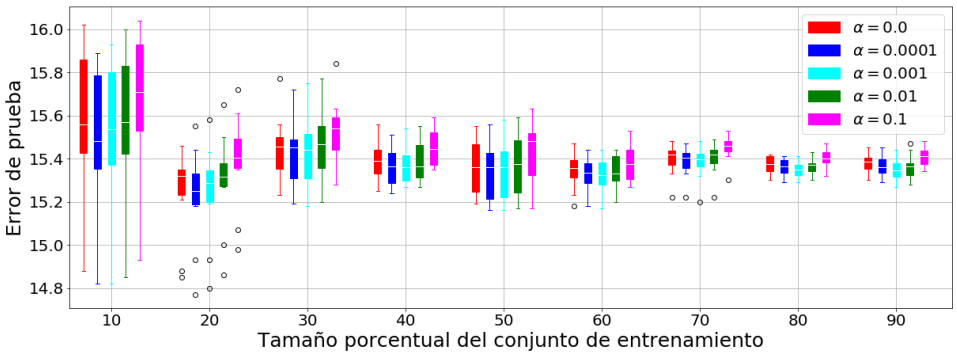


(a) Curvas de aprendizaje sobre el conjunto de prueba

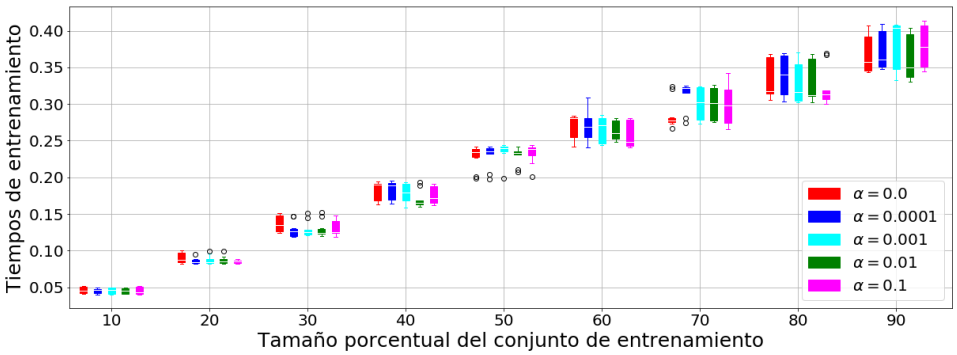


(b) Tiempos para entrenar el modelo en función del tamaño del conjunto de entrenamiento

Figura 5.4: Resultados del modelo bayes ingenuo multinomial

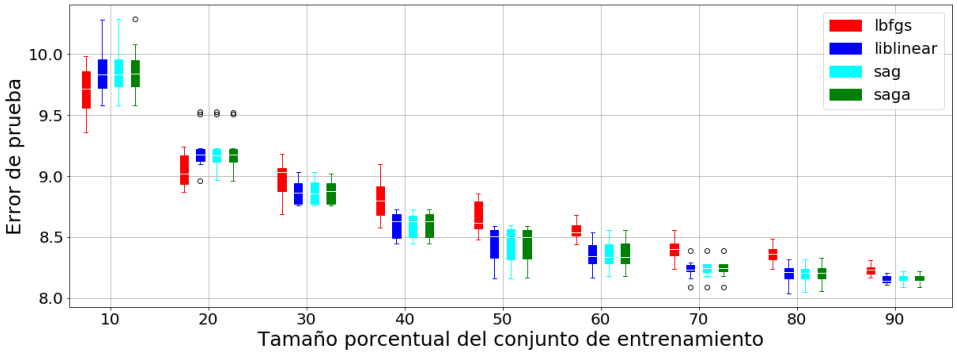


(a) Curvas de aprendizaje sobre el conjunto de prueba

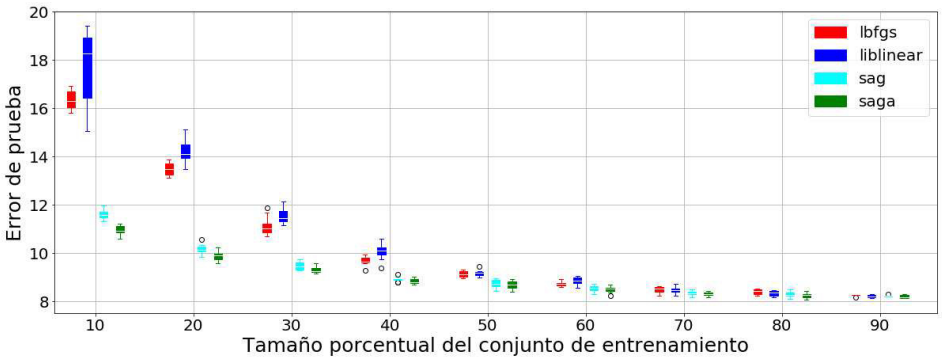


(b) Tiempos para entrenar el modelo en función del tamaño del conjunto de entrenamiento

Figura 5.5: Resultados del modelo bayes ingenuo bernoulli

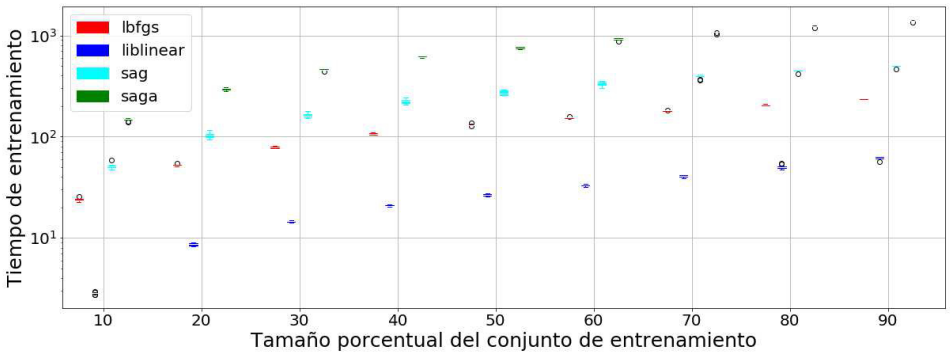


(a) *Penalización*=0.00001

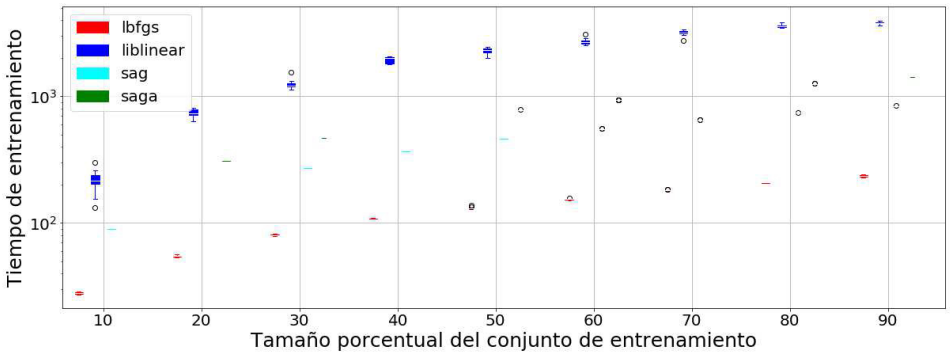


(b) *Penalización*=10.0

Figura 5.6: Curvas de aprendizaje sobre el conjunto de prueba del modelo regresión logística

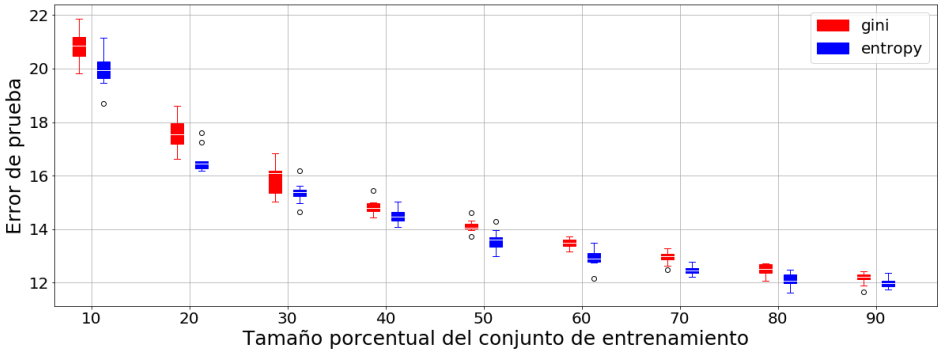


(a) *Penalización*=0.00001

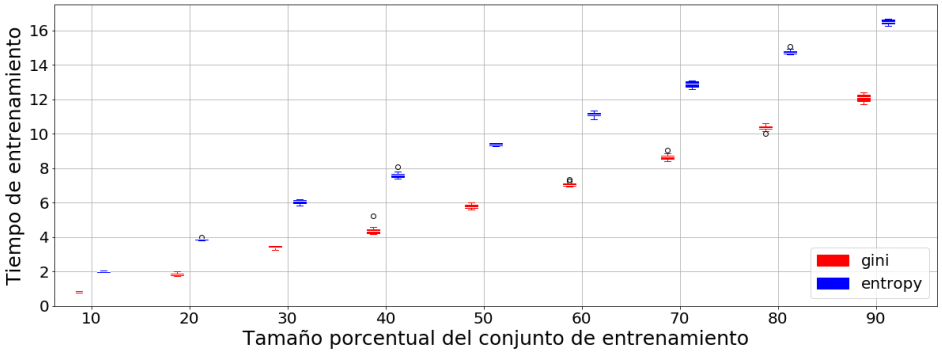


(b) *Penalización*=10.0

Figura 5.7: Tiempos de entrenamiento del modelo regresión logística

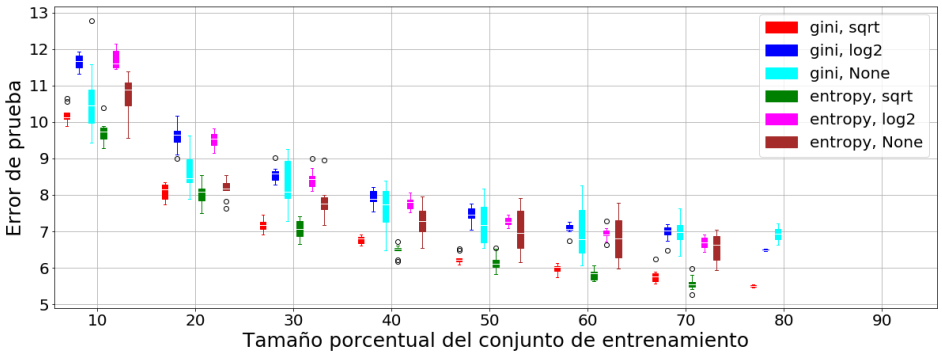


(a) Curvas de aprendizaje sobre el conjunto de prueba

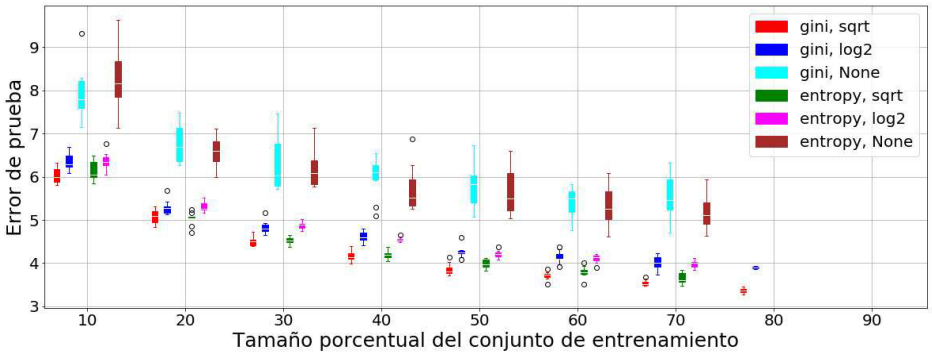


(b) Tiempos para entrenar el modelo en función del tamaño del conjunto de entrenamiento

Figura 5.8: Resultados del modelo árbol de decisión

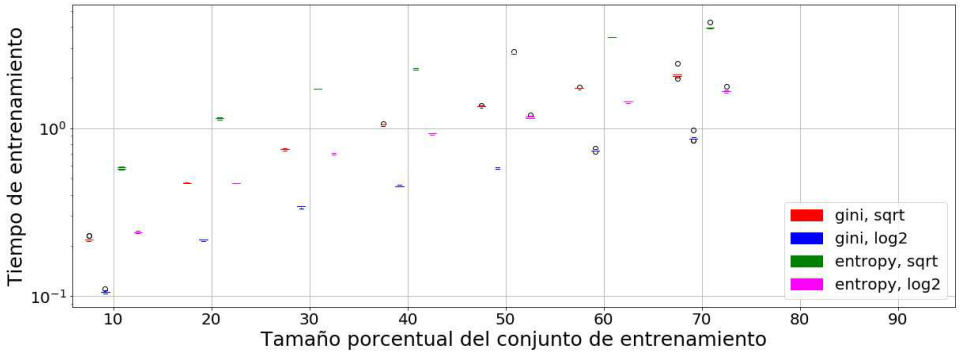


(a) 10 *estimadores*

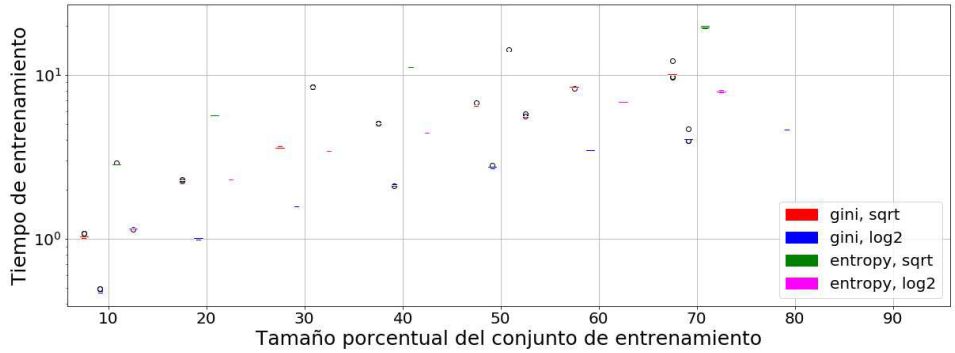


(b) 50 *estimadores*

Figura 5.9: Curvas de aprendizaje sobre el conjunto de prueba para el modelo bosque aleatorio

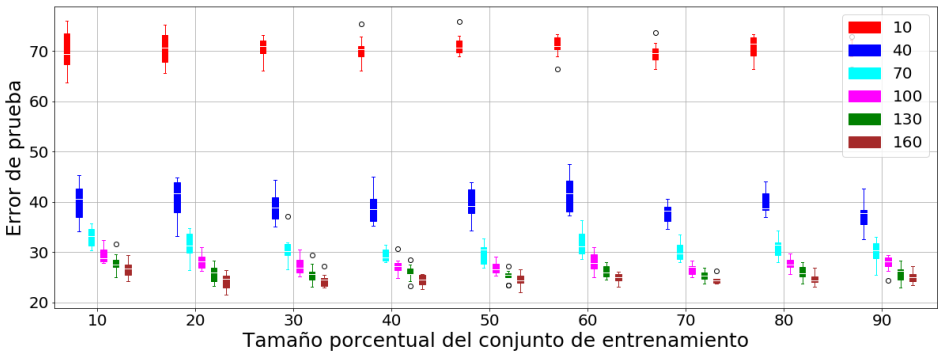


(a) 10 *estimadores*

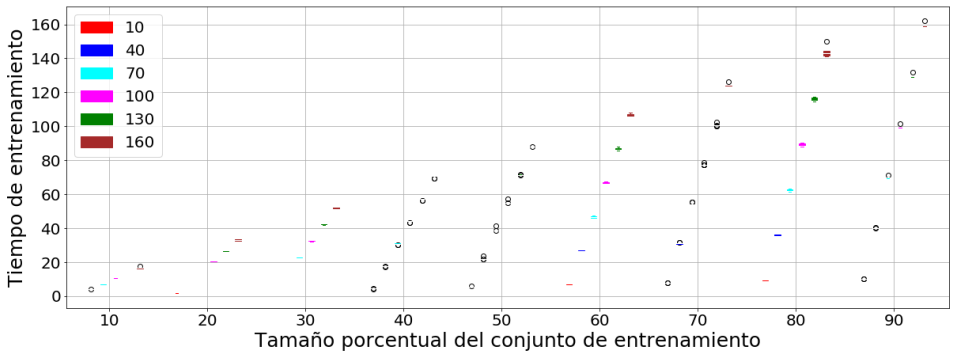


(b) 50 *estimadores*

Figura 5.10: Tiempos para entrenar el modelo bosque aleatorio en función del tamaño del conjunto de entrenamiento

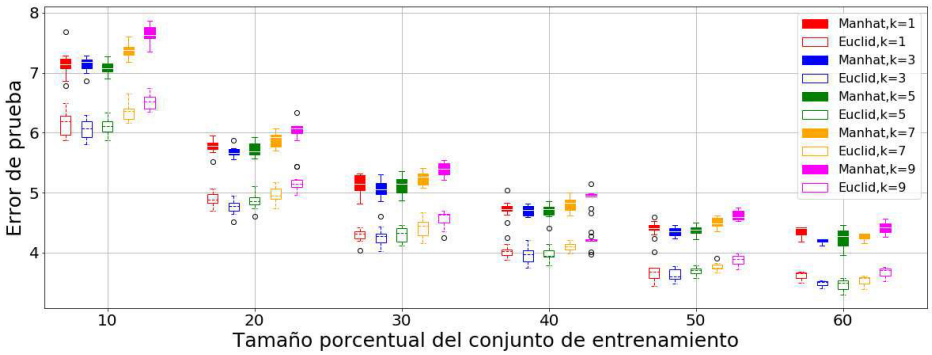


(a) Curvas de aprendizaje sobre el conjunto de prueba

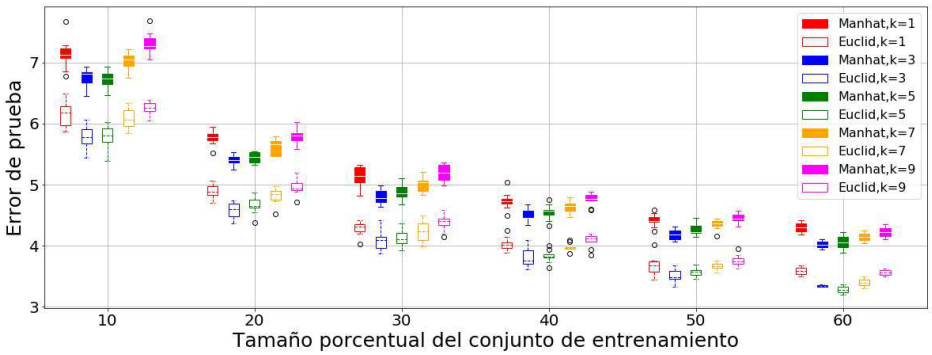


(b) Tiempos para entrenar el modelo en función del tamaño del conjunto de entrenamiento

Figura 5.11: Resultados del modelo AdaBoost

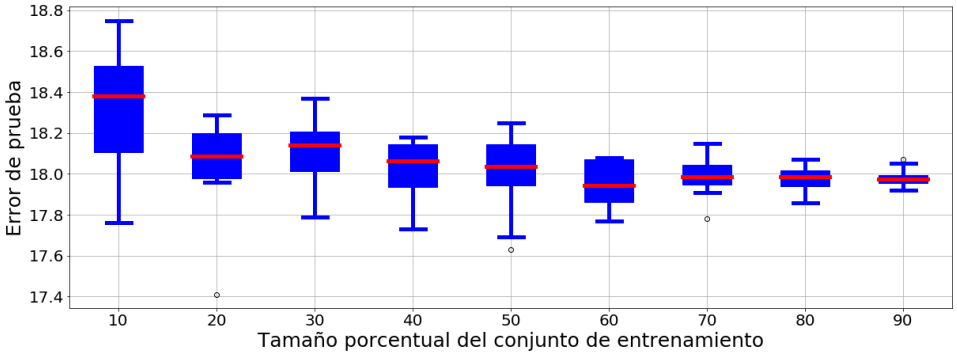


(a) Curvas de aprendizaje sobre el conjunto de prueba con estrategia uniforme

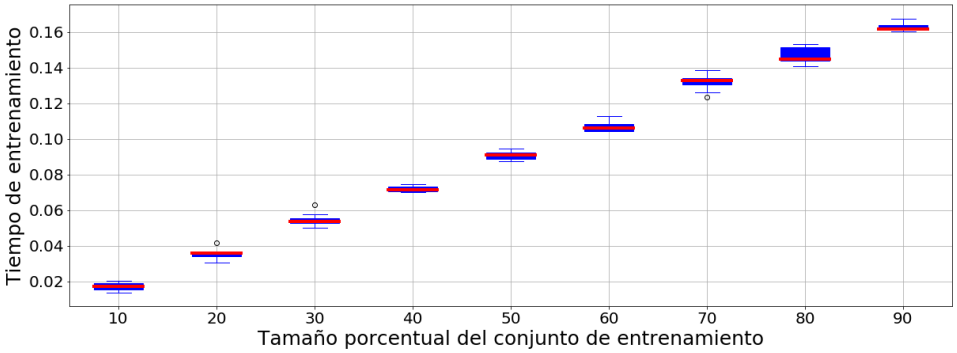


(b) Curvas de aprendizaje sobre el conjunto de prueba con estrategia distancia

Figura 5.12: Curvas de aprendizaje para el modelo k-vecinos más cercanos

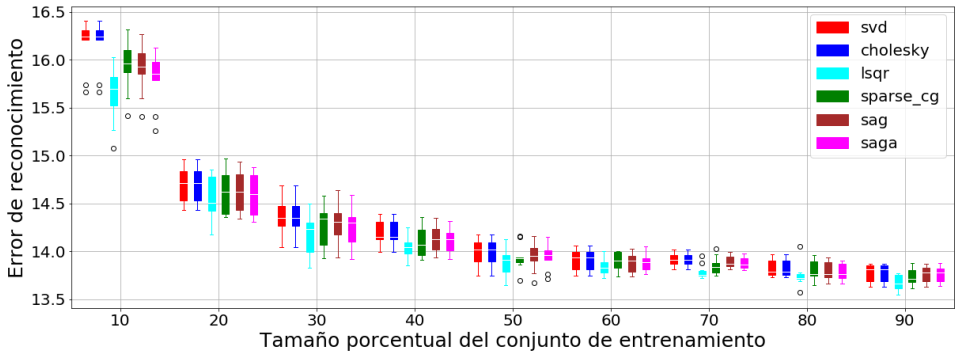


(a) Curvas de aprendizaje sobre el conjunto de prueba

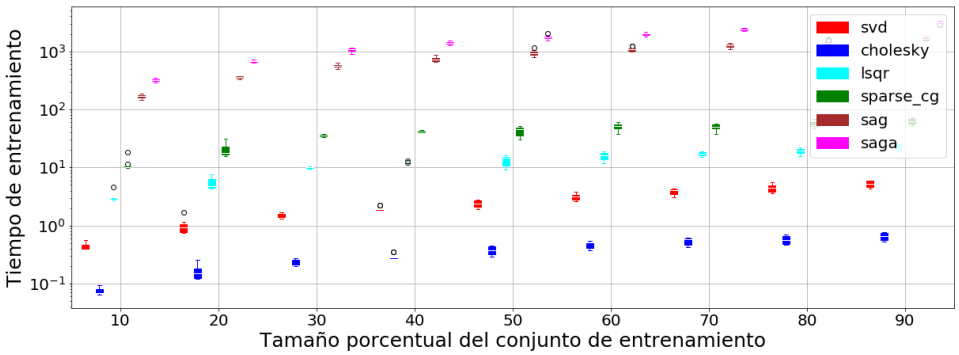


(b) Tiempos para entrenar el modelo en función del tamaño del conjunto de entrenamiento

Figura 5.13: Resultados del modelo centroide más cercano

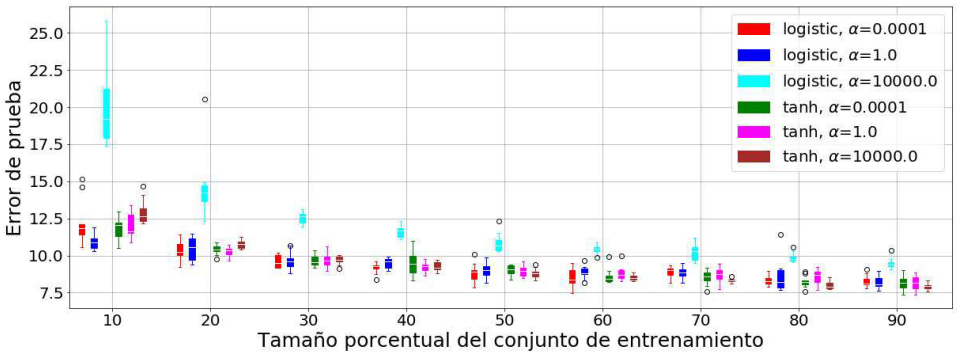


(a) Curvas de aprendizaje sobre el conjunto de prueba

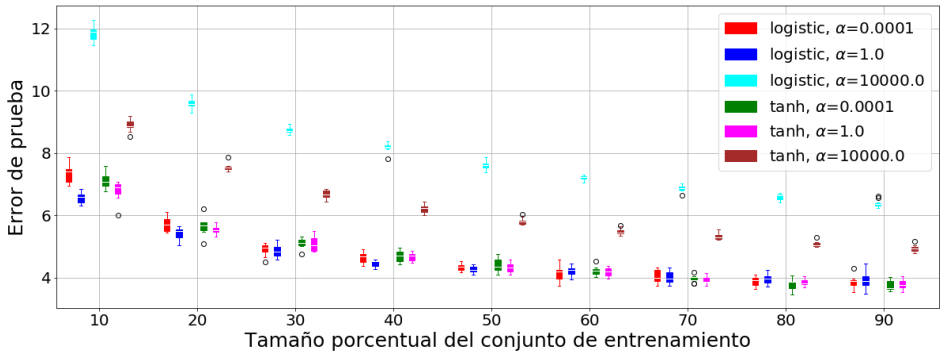


(b) Tiempos para entrenar el modelo en función del tamaño del conjunto de entrenamiento

Figura 5.14: Resultados del modelo ridge

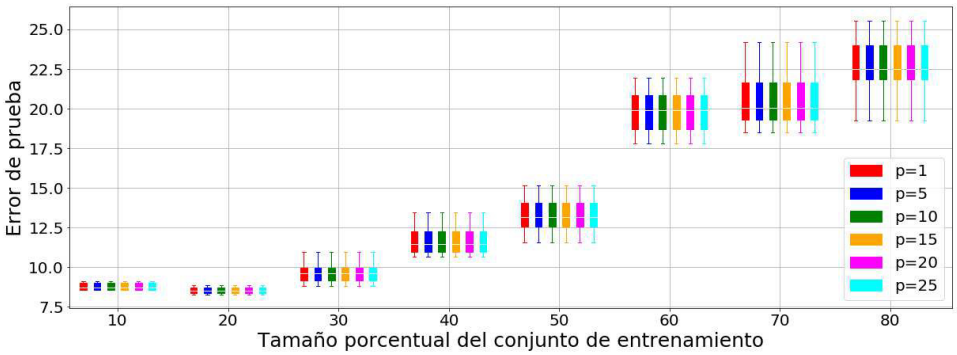


(a) 30 *neuronas*

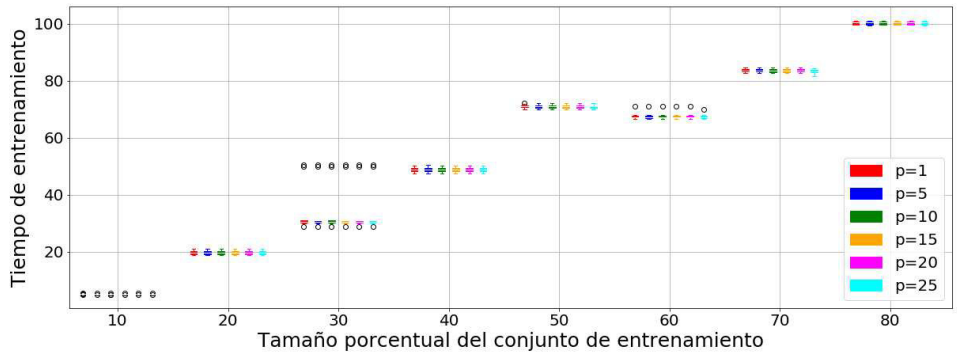


(b) 210 *neuronas*

Figura 5.15: Curvas de aprendizaje del modelo perceptrón multicada con estrategia LBFGS

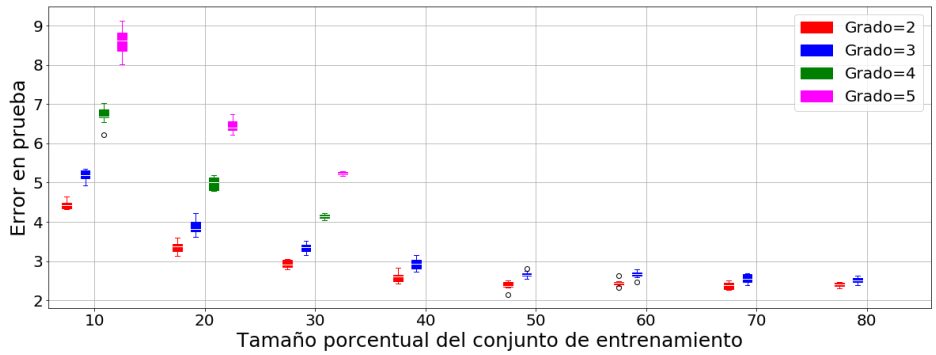


(a) Curvas de aprendizaje sobre el conjunto de prueba

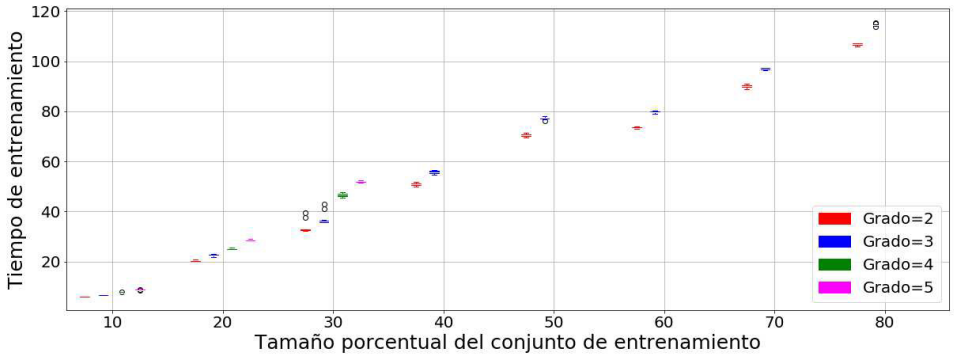


(b) Tiempos para entrenar el modelo en función del tamaño del conjunto de entrenamiento

Figura 5.16: Resultados del modelo SVM-Lineal



(a) Curvas de aprendizaje sobre el conjunto de prueba



(b) Tiempos para entrenar el modelo en función del tamaño del conjunto de entrenamiento

Figura 5.17: Resultados del modelo SVM-Polinomial

Capítulo 6

Discusión

AdaBoost

En este modelo se tiene solo el factor *estimador* el cual impacta en forma de exponencial negativa al error de prueba, para los niveles 130 y 160 existen diferencias significativas que en promedio son de una unidad para el error de prueba. También es claro que no tiene caso usar más allá del 20 % de los datos ya que no hay diferencias significativas si se aumenta el tamaño del conjunto de entrenamiento. En cuanto a los tiempos de entrenamiento en promedio son de 33.05 segundos y 26.84 segundos para 160 y 130 estimadores respectivamente cuando se tiene el 20 % de los datos. Tomando todo en cuenta lo mejor es usar 160 estimadores porque no se justifica el aumento de estimadores por la mejora que se obtendría en el error de prueba menor del orden de décimas, además de requerir más tiempo de entrenamiento. Bajo los supuestos anteriores en promedio el mejor error de prueba fue de 24.19.

k-vecinos más cercanos

Para el modelo de k-vecinos más cercanos se tienen tres factores *estrategia*, *métrica* y *vecinos*, de inicio el efecto del factor *métrica* en el error de prueba se

observa claramente, alcanzando en promedio una diferencia promedio de 0.8 a favor de la métrica euclidiana. De igual forma se observa claramente el efecto del tamaño del conjunto de entrenamiento al error de prueba en forma de exponencial negativa, aunque la diferencia entre el 50 % y 60 % no es significativa. Respecto al número de vecinos como se mencionó en resultados los niveles 3 y 5 no presentan diferencias entre ellos, pero si difieren del resto de niveles, pero en promedio la diferencia entre el mejor y peor nivel de número de vecinos se tiene una diferencia promedio de 0.3 en el error de prueba cuando se usa la estrategia distancia y 0.1 con la estrategia uniforme. En cuanto al factor *estrategia* se tiene una diferencia significativa de 0.2 en el error de prueba. En el caso de los tiempos de entrenamiento solo el tamaño del conjunto de entrenamiento tiene efecto porque solo guardan los datos con una estructura. Para fines prácticos vemos que el factor *estrategia* no tiene sentido, se recomienda usar la métrica euclidiana, que el factor *número de vecinos* tampoco tiene mucha relevancia y que solo es necesario utilizar el 50 % de los datos. El mejor error de prueba promedio fue de 3.3, el cual es comparaba con el 2.7 de [2].

Para el modelo de análisis de discriminante lineal conviene utilizar la *contracción* con la estrategia *lsqr* porque el error de prueba para esta combinación de parámetros tiene una diferencia significativa clara respecto a las demás combinaciones, manteniendo los tiempos de entrenamiento en el mismo orden de magnitud. Por otra parte, es claro que no es necesario utilizar todo el conjunto de datos de entrenamiento, ya que la diferencia en el error de prueba al entrenar con el 20 % y el 90 % de los datos de entrenamiento es menor a 0.5.

En el caso del modelo NBG el factor *suavizado* solo afecta al error de prueba mientras que el factor *tamaño del conjunto de entrenamiento* afecta tanto al error de prueba como a los tiempos de entrenamiento, de modo que uno se puede solo enfocar en que nivel del factor *suavizado* minimiza el error de prueba, el cual basado en las curvas de aprendizaje, figura 5.3, es el nivel 0.1. A pesar que hay una diferencia

significativa de mejora en el error de prueba de 0.42 al usar 20 % en lugar del 10 % de los datos de entrenamiento en la practica esta diferencia significativa es despreciable.

El modelo de NBM al igual que el modelo NBG solo tiene el factor *suavizado* solo que en este caso no genera diferencias significativas en el error de prueba ni en los tiempos de entrenamiento, por lo que este factor se puede dejar en 0 para nuestro problema. En cuanto al tamaño del conjunto de entrenamiento se tiene una diferencia significativa en el error de prueba de 0.32 al pasar de 10 % al 20 %, la cual no es significativa en la práctica por lo que solo es necesario usar el 10 % de los datos para entrenar para obtener un error de prueba promedio de 16.6.

Para el modelo de NBB el único de los factores a discutir es el de *suavizado* el cual tiene el mismo comportamiento que en el modelo NBM el factor *suavizado* de no afectar el error de prueba por lo que se pueda dejar en cero. En cuanto al tamaño del conjunto de entrenamiento pasa lo mismo que en las otras versiones de Bayes ingenuo donde a pesar de que hay una diferencia entre los usar 10 % y 20 % en la práctica no hay diferencia. En base a lo anterior el modelo de NBB presenta sus mejores resultados utilizando solo el 10 % de los datos con un valor de binarización de 102.4 para obtener un error de prueba promedio de 15.59.

Para el modelo de regresión logística se tiene los factores, *penalización* y *estrategia*. Para el caso del factor *penalización* se tiene que, al usar el nivel más bajo, es decir 0.00001, se tienen los menores errores de prueba cuando se entrena con pocos datos. En cuanto a las *estrategias* si hay diferencias significativas en el error de prueba, pero sin importar el nivel de la *penalización* al utilizar más del 50 % de los datos, la diferencia entre las *estrategias* son del orden de 0.18 en el error de prueba. Por su parte el *tamaño del conjunto de entrenamiento* afecta de manera exponencial negativa el error de prueba de modo que la disminución en el error de prueba al entrenar

con tamaños mayor de 60 % tiene mejoras menores de 0.1. Está claro que al entrenar con el 50 % de los datos no importa que niveles tomen los factores *penalización* y *estrategia* porque las diferencias entre todas las combinaciones posibles para el error de prueba no son significativas en la práctica de modo que elección de los factores se hace tomando en cuenta los tiempos de entrenamiento, que sería utilizar 0.00001 de *penalización* y la *estrategia* liblinear para obtener un error de prueba promedio de 8.6.

En el modelo árbol de decisión solo se tiene el factor *función de impureza* cuyos niveles tienen diferencias significativas en el error de prueba para todos los tamaños del conjunto de entrenamiento, pero tales diferencias son del orden de décimas. Del mismo modo la diferencia entre entrenar con el 80 % y 90 % es una mejora menor a 0.5 en el error de prueba. Por otra parte, las diferencias entre los tiempos de entrenamiento para Gini y entropía se hacen más grandes al aumentar el conjunto de datos de entrenamiento. En base a lo anterior se puede decir que en este modelo se puede utilizar todo el conjunto de entrenamiento completo porque los tiempos de entrenamiento tienen un promedio de 14 segundos sin importar que función de impureza utilizar, pero no se recomiendo agregas más datos al conjunto de entrenamiento porque las mejoras en el error de prueba son insignificantes. El mejor error de prueba fue en promedio fue de 11.97 el cual es comparable con el reportado en [2] de 12.16

Para el modelo de bosque aleatorio se tienen los factores, *estrategia*, *número de características* y *estimadores*. Una de las cosas fáciles de deducir es que las diferencias entre las funciones de impureza gini y entropía, estas apenas son el orden de 0.1 en promedio, por lo que no tiene significancia en la práctica. También está claro que el aumento de los *estimadores* tiene una diferencia significativa en el error de prueba, pero esa mejora tiene un patrón de exponencial negativa, por ejemplo, pasar de 30 a 40 estimadores solo tiene una mejora de 0.25 en promedio. Por otra parte el factor *estimadores* interacciona con el factor *número de características*, de modo que ayuda a marcar una diferencia significativa entre usar el nivel log2 y None.

En el caso del tamaño del conjunto de entrenamiento este también afecta al error de prueba en forma de exponencial negativa, de modo que las diferencias provocadas por el tamano del conjunto de entrenamiento desaparecen con al aumento de los datos de entrenamiento. Lo importante de este modelo es elegir el *número de características* sqrt y la *estrategia* Gini porque a pesar de no tener diferencia con entropía, los tiempos de entrenamiento son mejores y buenos al grado de permitir incrementar los estimadores hasta 50, en cuanto al tamaño de entrenamiento utilizar más del 50 % no genera cambios significativos en el error de prueba ya en la práctica. En promedio el mejor resultado fue de 3.35 en error de prueba, el cual está cercano al reportado en la literatura de 3.0 [31].

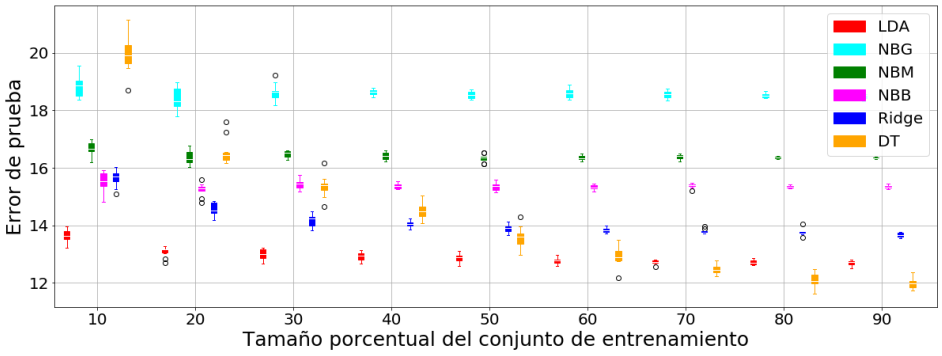
Ridge

En el modelo de Ridge se tienen dos factores, *penalización* y *estrategia*. De entrada, el factor *penalización* no afecta en ninguno de sus niveles al error de prueba, sin importar cual sea la *estrategia* que se esté utilizando, por lo que fácilmente se puede dejar en cero para nuestro problema. En cuanto al factor *estrategia* el nivel lsqr tiene diferencia significativa respecto al resto cuando se tienen solo el 10 % de los datos, al utilizar más datos las diferencias significativas en la práctica desaparecen y da igual que nivel de *estrategia* se utilice. Por su parte el *tamaño del conjunto de entrenamiento* se observa que influye significativamente en el error de prueba en forma de exponencial negativa, de modo que entre utilizar 50 % y 60 % ya no existe diferencia significativa. En base en los tiempos de entrenamiento la mejor *estrategia* a utilizar es la de cholesky porque es la que tiene los menores tiempos de entrenamiento de 0.37 segundo al entrenar con el 50 % de los datos y no presenta diferencias significativas en el error de prueba del resto de estrategias, obteniendo un error promedio de 13.9.

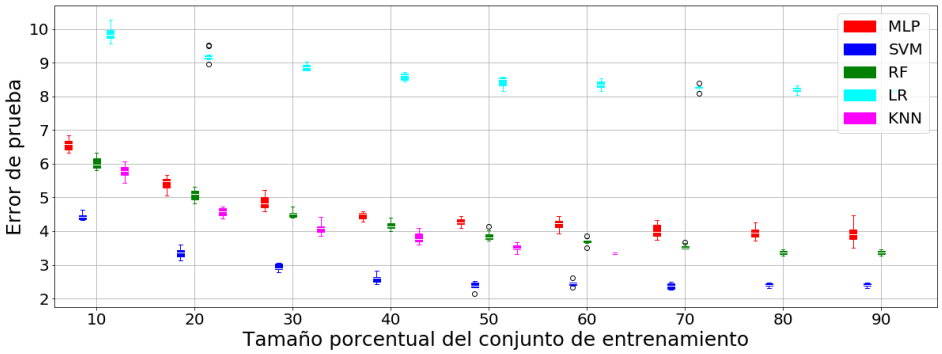
Máquinas de soporte vectorial

En el modelo de SVM con kernel polinomial tiene los factores *gamma*, *penalización* y *grado*, de los cuales *gamma* y *penalización* no influyen en el error de prueba por lo que se recomienda establecerlos en 1 para ambos factores. A partir de las curvas de aprendizaje se observa que el *tamaño del conjunto de entrenamiento* afecta al error de prueba en forma de exponencial negativa, de hecho para niveles más altos del 50 % la mejora en el error de prueba es menor a 0.1, mientras que el *grado* también afecta al error de prueba se ve que los mejores niveles son el 2 y 3, e incluso la diferencia entre estos dos niveles para *tamaños del conjunto de entrenamiento* mayores al 50 % es apenas de 0.2 en el error de prueba. En este caso se recomiendo usar el *grado* 2 porque los tiempos de entrenamiento son mejores que el *grado* 3 y para un *tamaño del conjunto de entrenamiento* del 50 % se tiene un error de prueba promedio de 2.38.

Para el kernel RBF se tienen los factores *penalización* y *gamma* pero como se mencionó en la sección de resultados se analizan los niveles 1 y 5 del factor *penalización* y los niveles 0.00001 y 0.0001 del factor *gamma* al ser los que valen la pena discutir. En la figura **??** se muestran las curvas de aprendizaje donde resalta el efecto que tiene *tamaño del conjunto de entrenamiento*, donde se observa que a partir de los tamaños 30 % y 60 % para los niveles de *gamma* 0.00001 y 0.0001 respectivamente con penalización 1 es donde se obtienen los mejores resultados para el error de prueba. Entrenar con *gamma* 0.00001 con tamaños mayores a 30 % no genera diferencias significativas y entrenar con *gamma* 0.0001 con tamaños mayores a 60 % tampoco genera diferencias significativas. Para los tiempos de entrenamiento se observan diferencias significativas solo por el *tamaño del conjunto de entrenamiento*, por lo tanto, solo es necesario entrenar con el 30 % de los datos y con un *gamma* de 0.0001 para obtener los mejores errores de prueba del modelo con un promedio de 8.6.



(a) Curvas de aprendizaje sobre el conjunto de prueba



(b) Tiempos para entrenar el modelo en función del tamaño del conjunto de entrenamiento

Figura 6.1: Resultados del modelo análisis discriminante cuadrático

Es interesante observar las características cuando se agrupan los modelos. En el caso de los modelos probabilísticos como los Bayes ingenuos y análisis de discriminantes comparten muchas características como son los tiempos rápidos de entrenamiento, solo necesitar del 10 % al 20 % de los datos para llegar a sus límites en error de prueba, exceptuando el análisis de discriminante lineal. En el caso de los modelos paramétricos como Ridge, perceptrón multicapa, SVM y regresión logística, sus curvas de aprendizaje tiene forma de exponencial negativa.

Capítulo 7

Conclusiones

En base a los resultados y discusiones es claro que no es necesario utilizar todo el conjunto de datos completo para llegar a los limites en cuanto a rendimiento de cada uno de los modelos al resolver una tarea. También es importante mencionar que no todas las variables de cada modelo provocan cambios significativos en el rendimiento. La mayoría de los modelos no presentan interacciones entre sus variables por lo que no es necesario realizar un diseño de experimentos factorial para obtener la combinación de factores que da mejores resultados, en lugar de eso se pueden tomar los factores como independiente y solo buscar el nivel que da mejores de resultados para después juntar los factores y obtener la mejor combinación de niveles.

Bibliografía

[1] Bari, M., B. Ambaw y M. Doroslovacki, «Comparison of Machine Learning Algorithms for Raw Handwritten Digits Recognition», págs. 1512–1516, 2018.

[2] Bari, M., B. Ambaw y M. Doroslovacki, «Comparison of Machine Learning Algorithms for Raw Handwritten Digits Recognition», págs. 1512–1516, 10 2018.

[3] Belongie, S., J. Malik y J. Puzicha, «Shape matching and object recognition using shape contexts», *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **24**(4), págs. 509–522, April 2002.

[4] Bishop, C. M., *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer-Verlag, Berlin, Heidelberg, 2006.

[5] Breiman, L., «Random Forests», *Machine Learning*, **45**(1), págs. 5–32, 2001, URL `https://doi.org/10.1023/A:1010933404324`.

[6] Breiman, L., J. Friedman, R. Olshen y C. Stone, *Classification and Regression Trees*, Wadsworth and Brooks, Monterey, CA, 1984.

[7] Chaudhuri, A., K. Mandaviya, P. Badelia y S. K. Ghosh, *Introduction*, Springer International Publishing, Cham, págs. 1–7, 2017.

[8] Ciregan, D., U. Meier y J. Schmidhuber, «Multi-column deep neural networks for image classification», en *2012 IEEE Conference on Computer Vision and Pattern Recognition*, págs. 3642–3649, June 2012.

[9] CORTES, C., L. D. JACKEL, S. A. SOLLA, V. VAPNIK y J. S. DENKER, «Learning Curves: Asymptotic Values and Rate of Convergence», en J. D. Cowan, G. Tesauro y J. Alspector (editores), *Advances in Neural Information Processing Systems 6*, Morgan-Kaufmann, págs. 327–334, 1994, URL http://papers.nips.cc/paper/803-learning-curves-asymptotic-values-and-rate-of-convergence.pdf.

[10] JAMES, G., D. WITTEN, T. HASTIE y R. TIBSHIRANI, *Linear Regression*, Springer New York, págs. 59–126, 2013, URL https://doi.org/10.1007/978-1-4614-7138-7_3.

[11] KOLLER, D. y N. FRIEDMAN, *Probabilistic Graphical Models: Principles and Techniques*, The MIT Press, 2009.

[12] LAUER, F., C. Y. SUEN y G. BLOCH, «A trainable feature extractor for handwritten digit recognition», *Pattern Recognition*, **40**(6), págs. 1816 – 1824, 2007, URL http://www.sciencedirect.com/science/article/pii/S0031320306004250.

[13] LECUN, Y., L. BOTTOU, Y. BENGIO y P. HAFFNER, «Gradient-based learning applied to document recognition», *Proceedings of the IEEE*, **86**(11), págs. 2278–2324, Nov 1998.

[14] LECUN, Y., C. CORTES y C. BURGES, «THE MNIST DATABASE of handwritten digits», http://yann.lecun.com/exdb/mnist/index.html, 1998.

[15] LEVI, D. y S. ULLMAN, «Learning Model Complexity in an Online Environment», en *2009 Canadian Conference on Computer and Robot Vision*, págs. 260–267, May 2009.

[16] LIU, C.-L., K. NAKASHIM, H. SAKO y H. FUJISAWA, «Handwritten digit recognition using state-of-the-art techniques», en *Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition*, págs. 320–325, 2002.

[17] LIU, C.-L., K. NAKASHIMA, H. SAKO y H. FUJISAWA, «Handwritten digit recognition: benchmarking of state-of-the-art techniques», *Pattern Recognition*, **36**(10), págs. 2271–2285, 2003, URL http://www.sciencedirect.com/science/article/pii/S0031320303000852.

[18] PEDREGOSA, F., G. VAROQUAUX, A. GRAMFORT, V. MICHEL, B. THIRION, O. GRISEL, M. BLONDEL, P. PRETTENHOFER, R. WEISS, V. DUBOURG, J. VANDERPLAS, A. PASSOS, D. COURNAPEAU, M. BRUCHER, M. PERROT y E. DUCHESNAY, «Scikit-learn: Machine Learning in Python», *J. Mach. Learn. Res.*, **12**, págs. 2825–2830, noviembre 2011, URL http://dl.acm.org/citation.cfm?id=1953048.2078195.

[19] PERLICH, C., *Learning Curves in Machine Learning*, Springer US, Boston, MA, págs. 577–580, 2010, URL https://doi.org/10.1007/978-0-387-30164-8_452.

[20] QUINLAN, J. R., *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[21] RANZATO, M., C. POULTNEY, S. CHOPRA y Y. LECUN, «Efficient Learning of Sparse Representations with an Energy-based Model», en *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, MIT Press, Cambridge, MA, USA, págs. 1137–1144, 2006, URL http://dl.acm.org/citation.cfm?id=2976456.2976599.

[22] SABOUR, S., N. FROSST y G. E. HINTON, «Dynamic Routing Between Capsules», en I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan y R. Garnett (editores), *Advances in Neural Information Processing Systems 30*, Curran Associates, Inc., págs. 3856–3866, 2017, URL http://papers.nips.cc/paper/6975-dynamic-routing-between-capsules.pdf.

[23] SCHAETTI, N., M. SALOMON y R. COUTURIER, «Echo State Networks-Based Reservoir Computing for MNIST Handwritten Digits Recognition», en *2016*

IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES), págs. 484–491, 2016.

[24] SIMARD, P. Y., D. STEINKRAUS y J. C. PLATT, «Best practices for convolutional neural networks applied to visual document analysis», en *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, págs. 958–963, Aug 2003.

[25] SINGH, R., R. KUMAR MISHRA, S. BEDI, S. KUMAR y A. KUMAR SHUKLA, «A Literature Review on Handwritten Character Recognition based on Artificial Neural Network», *International Journal of Computer Sciences and Engineering*, **6**, págs. 753–758, 11 2018.

[26] SOLEM, J. E., *Programming computer vision with Python*, O'Reilly, Beijing; Cambridge; Sebastopol [etc.], 2012, URL `http://www.amazon.com/dp/1449316549/ref=pe_259560_30100590_pd_re_dt_dt7`.

[27] TIBSHIRANI, R., T. HASTIE, B. NARASIMHAN y G. CHU, «Class Prediction by Nearest Shrunken Centroids, with Applications to DNA Microarrays», *Statistical Science.*, **18**, 2003.

[28] TONG, Z., K. AIHARA y G. TANAKA, «A Hybrid Pooling Method for Convolutional Neural Networks», en *Neural Information Processing*, Springer International Publishing, págs. 454–461, 2016.

[29] WAN, L., M. ZEILER, S. ZHANG, Y. LECUN y R. FERGUS, «Regularization of Neural Networks Using Dropconnect», en *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, JMLR.org, págs. III–1058–III–1066, 2013, URL `http://dl.acm.org/citation.cfm?id=3042817.3043055`.

[30] Wu, S., W. Wei y L. Zhang, «Comparison of Machine Learning Algorithms for Handwritten Digit Recognition», en *Computational Intelligence and Intelligent Systems*, Springer Singapore, págs. 532–542, 2018.

[31] Wu, S., W. Wei y L. Zhang, «Comparison of Machine Learning Algorithms for Handwritten Digit Recognition», en K. Li, W. Li, Z. Chen y Y. Liu (editores), *Computational Intelligence and Intelligent Systems*, Springer Singapore, págs. 532–542, 2018.

[32] Zhu, J., S. Rosset, H. Zou y T. Hastie, «Multi-class AdaBoost», *Statistics and its interface*, **2**, págs. 349–360, 02 2009.

Resumen autobiográfico

Alan Arnoldo Alcantar Gómez

Candidato para obtener el grado de
Maestría en Ciencias
en Ingeniería de Sistemas

Universidad Autónoma de Nuevo León
Facultad de Ingeniería Mecánica y Eléctrica

Tesis:
Comparación y análisis de las curvas de aprendizaje sobre la base de datos MNIST de los algoritmos de clasificación de la librería SK-Learn

Egresado de la Licenciatura en Física del Departamento de Física en él 2016 y de la Especialidad de Desarrollo Sustentable en él 2017, ambos de la Universidad de Sonora.