UNIVERSIDAD JUÁREZ AUTÓNOMA DE TABASCO

DIVISIÓN ACADÉMICA DE CIENCIAS BÁSICAS

REDES NEURONALES Y SU APLICACIÓN EN LA CLASIFICACIÓN DE PATRONES

TESIS

QUE PARA OBTENER EL TÍTULO DE

MAESTRO EN CIENCIAS EN MATEMÁTICAS APLICADAS

PRESENTA

EDGAR ALAMILLA JIMÉNEZ

DIRECTORES

DRA. ADDY MARGARITA BOLÍVAR CIMÉ

DR. EDILBERTO NÁJERA RANGEL

CUNDUACÁN, TAB, MEX. DICIEMBRE 2021

DIRECCIÓN

30 de noviembre de 2021

LIC. EDGAR ALAMILLA JIMÉNEZ
PASANTE DE LA MAESTRÍA EN CIENCIAS EN MATEMÁTICAS APLICADAS
P R E S E N T E.

Por medio de la presente y de la manera más atenta, me dirijo a Usted para hacer de su conocimiento que proceda a la impresión del trabajo titulado **"REDES NEURONALES Y SU APLICACIÓN EN LA CLASIFICACIÓN DE PATRONES"** en virtud de que reúne los requisitos para presentar el EXAMEN PROFESIONAL para obtener el grado de Maestro en Ciencias en Matemáticas Aplicadas.

Sin otro particular, reciba un cordial saludo.

ATENTAMENTE

DIVISIÓN ACADÉMICA DE CIENCIAS BÁSICAS

DR. GERARDO DELGADILLO PIÑÓN
DIRECTOR

DR'GDP/M´NLBA
C.c.p.- Archivo

CARTA DE AUTORIZACIÓN

El que suscribe, autoriza por medio del presente escrito a la Universidad Juárez Autónoma de Tabasco para que utilice tanto física como digitalmente la tesis de grado denominada ***"Redes neuronales y su aplicación en la clasificación de patrones"***, de la cual soy autor y titular de los Derechos de Autor.

La finalidad del uso por parte de la Universidad Juárez Autónoma de Tabasco de la tesis antes mencionada, será única y exclusivamente para difusión, educación y sin fines de lucro; autorización que se hace de manera enunciativa mas no limitativa para subirla a la Red Abierta de Bibliotecas Digitales (RABID) y a cualquier otra red académica con las que la Universidad tenga relación institucional.

Por lo antes manifestado, libero a la Universidad Juárez Autónoma de Tabasco de cualquier reclamación legal que pudiera ejercer respecto al uso y manipulación de la tesis mencionada y para los fines estipulados en este documento.

Se firma la presente autorización en la ciudad de Villahermosa, Tabasco a los 02 días del mes de Diciembre del año 2021.

AUTORIZÓ

L.M. EDGAR ALAMILLA JIMÉNEZ

Índice general

Agradecimientos.

Primeramente a Dios por darme la oportunidad de terminar mis estudios y haberme dado salud para lograr mis objetivos, además de su infinita bondad y amor.

Le doy gracias a mis padres Enrique y María Edith por apoyarme en todo momento, por los valores que me han inculcado, y por haberme dado la facilidad de tener una excelente educación en el transcurso de mi vida. A mi hermano Elí por ser parte importante de mi vida.

A la Universidad Juárez Autónoma de Tabasco, por favorecerme con su apoyo en cumplir mis metas propuestas.

A mis maestros les agradezco todo el apoyo brindado a lo largo de la maestría, por su tiempo, amistad y por los conocimientos que me trasmitieron.

Y un agradecimiento especial a la Dra. Addy Margarita Bolívar Cimé y al Dr. Edilberto Nájera Rangel por ser mis directores de tesis y además apoyarme incondicionalmente en el desarrollo de este trabajo de tesis, por su tiempo, y la paciencia otorgada GRACIAS.

Introducción.

La ciencia de datos es hoy en día una herramienta fundamental para la explotación de datos y la generación de conocimiento. Entre los objetivos que persigue se encuentra la búsqueda de modelos que describan patrones y comportamientos a partir de los datos con el fin de tomar decisiones o hacer predicciones. Es un área que ha experimentado un enorme crecimiento al extenderse el acceso a grandes volúmenes de datos e incluso su tratamiento en tiempo real. Abarca a numerosos grupos de investigación de diferentes áreas (computación, estadística, matemáticas, ingeniería, etc.) que trabajan en la propuesta de nuevos algoritmos, técnicas de computación e infraestructuras para la captura, almacenamiento y procesado de datos, etc., ver [4].

Una parte fundamental de la ciencia de datos son las redes neuronales artificiales, las cuales son modelos computacionales inspirados en los trabajos de investigación, iniciados en 1930, cuyo propósito era modelar computacionalmente el aprendizaje humano llevado a cabo a través de las neuronas en el cerebro. Las redes neuronales son una nueva forma de analizar la información de tal modo que son capaces de detectar y aprender patrones complejos y características de los datos. Como nuestro cerebro, aprenden de la experiencia y del pasado, y aplican el conocimiento adquirido para resolver problemas nuevos. Este aprendizaje se obtiene como resultado del entrenamiento (training), el cual permite la sencillez y la potencia de adaptación y evolución ante una realidad cambiante y muy dinámica. Una vez adiestradas, las redes neuronales pueden hacer previsiones, clasificaciones y segmentación. Además, en la mayoría de los casos presentan una eficiencia y confiabilidad por lo menos similares a los métodos estadísticos y sistemas expertos. En los casos de muy alta complejidad, las redes neuronales son especialmente útiles dada la dificultad para modelar que tienen otras técnicas. En 1958, Rosenblatt [14], propuso el primer modelo precursor de redes neuronales, el perceptrón. Sin embargo éste tenía capacidades muy limitadas, lo que trajo como consecuencia que en la década de 1970 esta área de investigación fuera casi abandonada; no fue sino hasta la década de 1980, con el uso de hardware computacional, que se dio un auge en la investigación de redes neuronales, el cual persiste

hasta el día de hoy, ver [13].

En esta tesis se estudió la forma en que se utilizan las redes neuronales artificiales en la clasificación de patrones, mostrando su implementación, y proporcionando algunos ejemplos de su aplicación en diversos campos.

A continuación se describe como está divida esta tesis. En el Capítulo 1 se definen conceptos básicos fundamentales en redes neuronales artificiales. En el Capítulo 2 se presenta el perceptrón multicapa, el gradiente descendente y el algoritmo de Backpropagation (propagación hacia atrás), que son piezas claves para la implementación de las redes neuronales. En el Capítulo 3 se muestran ejemplos de aplicación con datos reales encontrados en la literatura para ilustrar el uso de las redes neuronales artificiales como una herramienta para clasificar patrones. Por último se presenta un Apéndice donde se muestran los códigos del software Rstudio utilizados en este trabajo, empleando como paquetería primordial el "nnet"(ver [12]).

Capítulo 1

Conceptos básicos.

En este capítulo se proporcionan las herramientas básicas de redes neuronales artificiales que serán necesarias en esta tesis.

1.1. El problema de clasificación.

Se tienen datos $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, donde $x_i \in \mathbb{R}^d$, $y_i \in \{1, 2, \ldots, k\}$ y k representa el número de clases. Estos datos son llamados datos de entrenamiento.

Se supone que los datos provienen de una distribución desconocida. El problema que se quiere resolver es el de clasificar un nuevo dato x en alguna de estas clases. Para ello se desea construir una regla de clasificación, la cual está dada por una función

$$g : \mathbb{R}^d \to \{1, 2, \ldots, k\}$$

$$x \mapsto g(x).$$

1.2. Probabilidad de error de Bayes.

Sea (X, Y) un par de variables aleatorias que toman valores en $\mathbb{R}^d$ y $\{0, 1\}$, respectivamente. El par (X, Y) puede ser descrito por el par (μ, η), donde μ es la medida de probabilidad de X y η es la regresión de Y en X. Es decir, para un conjunto Borel-medible $A \subseteq \mathbb{R}^d$,

$$\mu(A) = P(X \in A),$$

y para cualquier $x \in \mathbb{R}^d$,

$$\eta(x) = P(Y = 1|X = x) = E(Y|X = x).$$

Así, $\eta(x)$ es la probabilidad condicional de $Y = 1$ dado $X = x$. Para ver que (μ, η) son suficientes para describir la distribución de (X, Y), observar que para cualquier $C \subseteq \mathbb{R}^d \times \{0, 1\}$, tenemos que

$$C = (C \cap (\mathbb{R}^d \times \{0\})) \cup (C \cap (\mathbb{R}^d \times \{1\})) \equiv (C_0 \times \{0\}) \cup (C_1 \times \{1\}),$$

para algunos C_0 y C_1 subconjuntos de $\mathbb{R}^d$,

$$\begin{aligned} P((X, Y) \in C) &= P(X \in C_0, Y = 0) + P(X \in C_1, Y = 1) \\ &= \int_{C_0} (1 - \eta(x))\mu(dx) + \int_{C_1} \eta(x)\mu(dx). \end{aligned}$$

Como esto es válido para cualquier conjunto Borel-medible C, la distribución de (X, Y) está determinada por (μ, η). La función η, a veces, es llamada la **probabilidad a posteriori**.

Cualquier función $g : \mathbb{R}^d \to \{0, 1\}$ define un **clasificador** o **una regla de clasificación** o **función de decisión**. La probabilidad de error de g es $L(g) = P(g(X) \neq Y)$. Es de particular interés la **regla de clasificación de Bayes** dada por

$$g^*(x) = \begin{cases} 1 & \text{si } \eta(x) > \dfrac{1}{2}, \\ 0 & \text{en otro caso.} \end{cases}$$

Esta función de clasificación minimiza la probabilidad de error, como se verá a continuación.

Teorema 1. *Para cualquier regla de clasificación $g : \mathbb{R}^d \to \{0, 1\}$,*

$$P(g^*(X) \neq Y) \leq P(g(X) \neq Y),$$

es decir, g^* es la regla de clasificación óptima.

Demostración. Dado $X = x$, la probabilidad de error condicional de cualquier regla de clasificación g puede ser expresada como

$$\begin{aligned} P(g(X) \neq Y | X = x) &= 1 - P(Y = g(X) | X = x) \\ &= 1 - [P(Y = 1, g(X) = 1 | X = x) + P(Y = 0, g(X) = 0 | X = x)] \\ &= 1 - \left[I_{\{g(x)=1\}} P(Y = 1 | X = x) + I_{\{g(x)=0\}} P(Y = 0 | X = x) \right] \\ &= 1 - \left[I_{\{g(x)=1\}} \eta(x) + I_{\{g(x)=0\}} (1 - \eta(x)) \right], \end{aligned}$$

donde I_A denota la función indicadora del conjunto A. Así, para todo $x \in \mathbb{R}^d$,

$$
\begin{aligned}
&P(g(X) \neq Y|X = x) - P(g^*(X) \neq Y|X = x) \\
&= 1 - \left[I_{\{g(x)=1\}}\eta(x) + I_{\{g(x)=0\}}(1 - \eta(x))\right] \\
&\qquad -\{1 - \left[I_{\{g^*(x)=1\}}\eta(x) + I_{\{g^*(x)=0\}}(1 - \eta(x))\right]\} \\
&= I_{\{g^*(x)=1\}}\eta(x) + I_{\{g^*(x)=0\}}(1 - \eta(x)) - I_{\{g(x)=1\}}\eta(x) - I_{\{g(x)=0\}}(1 - \eta(x)) \\
&= \eta(x)\left[I_{\{g^*(x)=1\}} - I_{\{g(x)=1\}}\right] + (1 - \eta(x))\left[I_{\{g^*(x)=0\}} - I_{\{g(x)=0\}}\right] \\
&= \eta(x)\left[I_{\{g^*(x)=1\}} - I_{\{g(x)=1\}}\right] + (1 - \eta(x))\left[1 - I_{\{g^*(x)=1\}} - 1 + I_{\{g(x)=1\}}\right] \\
&= (2\eta(x) - 1)\left[I_{\{g^*(x)=1\}} - I_{\{g(x)=1\}}\right] \geq 0 \text{ (por la definición de } g^*(x)).
\end{aligned}
$$

Por lo tanto,

$$P(g^*(X) \neq Y|X = x) \leq P(g(X) \neq Y|X = x).$$

Integrando ambos lados de la desigualdad con respecto de $\mu(dx)$ tenemos que

$$P(g^*(X) \neq Y) \leq P(g(X) \neq Y).$$

□

Notar que si X toma valores en $\mathbb{R}$, $\eta(x)$ tiene la forma de la función en la Figura 1.1.

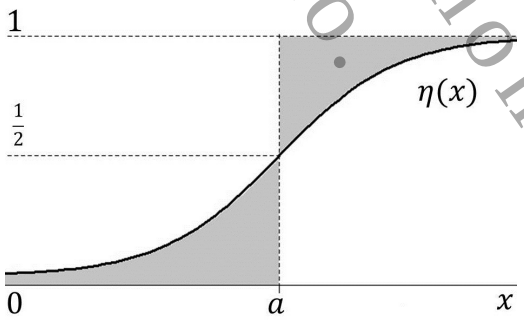


Figura 1.1: La regla de decisión de Bayes en la cual se asigna 1 si $x > a$ y 0 en otro caso, para algún $a \in \mathbb{R}$.

Por lo que en este caso tenemos que la regla de decisión está dada como

$$g^*(x) = \begin{cases} 1, & \text{si } \eta(x) > \dfrac{1}{2}, \\ 0, & \text{en otro caso.} \end{cases} = \begin{cases} 1, & \text{si } x > a, \\ 0, & \text{en otro caso.} \end{cases}$$

La cantidad $L^* = P(g^*(X) \neq Y)$ es llamada la **probabilidad de error de Bayes**, **error de Bayes** o **el riesgo de Bayes**. Por la demostración anterior

$$\begin{aligned} L(g) &= P(g(X) \neq Y) = \int_{\mathbb{R}^d} P(g(X) \neq Y | X = x)\mu(dx) \\ &= \int_{\mathbb{R}^d} \left[1 - \{I_{\{g(x)=1\}}\eta(x) + I_{\{g(x)=0\}}(1-\eta(x))\}\right] \mu(dx) \\ &= 1 - \int_{\mathbb{R}^d} \{I_{\{g(x)=1\}}\eta(x) + I_{\{g(x)=0\}}(1-\eta(x))\}\mu(dx) \\ &= 1 - E\{I_{\{g(X)=1\}}\eta(X) + I_{\{g(X)=0\}}(1-\eta(X))\}, \end{aligned}$$

en particular

$$L^* = 1 - E\{I_{\{\eta(X)>\frac{1}{2}\}}\eta(X) + I_{\{\eta(X)\leq\frac{1}{2}\}}(1-\eta(X))\}.$$

Se tiene que la probabilidad a posteriori

$$\eta(x) = P(Y = 1|X = x) = E(Y|X = x),$$

minimiza el error cuadrático cuando Y es predicha por $f(X)$, para alguna función $f : \mathbb{R}^d \to \mathbb{R}$:

$$E\{(\eta(X) - Y)^2\} \leq E\{(f(X) - Y)^2\}.$$

Para ver lo anterior, observe que para cada $x \in \mathbb{R}^d$,

$$\begin{aligned} &E\{(f(X) - Y)^2 | X = x\} = E\{(f(x) - \eta(x) + \eta(x) - Y)^2 | X = x\} \\ &= (f(x) - \eta(x))^2 + 2(f(x) - \eta(x))E\{\eta(x) - Y | X = x\} + E\{(\eta(x) - Y)^2 | X = x\}; \end{aligned} \tag{1.1}$$

puesto que

$$E\{\eta(x) - Y | X = x\} = \eta(x) - E(Y|X = x) = \eta(x) - \eta(x) = 0, \tag{1.2}$$

al sustituir (1.2) en (1.1) se obtiene

$$\begin{aligned} E\{(f(X) - Y)^2 | X = x\} &= (f(x) - \eta(x))^2 + \cancel{2(f(x) - \eta(x))(0)} \\ &\quad + E\{(\eta(x) - Y)^2 | X = x\} \\ &= (f(x) - \eta(x))^2 + E\{(\eta(X) - Y)^2 | X = x\}. \end{aligned}$$

Integrando respecto de $\mu(dx)$ tenemos

$$E\{(f(X) - Y)^2\} = E\{(f(X) - \eta(X))^2\} + E\{(\eta(X) - Y)^2\}.$$

Así,

$$E\{(\eta(X) - Y)^2\} \leq E\{(f(X) - Y)^2\}.$$

Ejemplo 1 (Desempeño de un estudiante en un curso)**.** *Considérese el desempeño de un estudiante en un curso. Sea $Y = 1$ si aprueba y $Y = 0$ si reprueba. Supongamos que X es el número de horas de estudio por semana del estudiante, que nos servirá como un factor para clasificar al estudiante en una de las categorías. La función de regresión $\eta(x) = P(Y = 1|X = x)$ es probablemente una función creciente en x.*

Supongamos que $\eta(x) = \dfrac{x}{c+x}$, con $c > 0$. La regla de clasificación de Bayes es

$$g^*(x) = \begin{cases} 1, & si\ \eta(x) > \dfrac{1}{2}, \\ 0, & en\ otro\ caso. \end{cases} = \begin{cases} 1, & si\ x > c, \\ 0, & en\ otro\ caso. \end{cases}$$

El correspondiente error de Bayes es

$$\begin{aligned} L^* &= L(g^*) = 1 - E\left[I_{\{\eta(X)>\frac{1}{2}\}}\eta(X) + I_{\{\eta(X)\leq\frac{1}{2}\}}(1-\eta(X))\right] \\ &= 1 - E\left[I_{\{X>c\}}\eta(X) + I_{\{X\leq c\}}(1-\eta(X))\right] \\ &= E\left[I_{\{X>c\}}(1-\eta(X)) + I_{\{X\leq c\}}\eta(X)\right] \\ &= E\left[I_{\{X>c\}}\frac{c}{c+X} + I_{\{X\leq c\}}\frac{X}{c+X}\right] \\ &= E\left(\frac{\text{mín}(c,X)}{c+X}\right) \\ &= E\left(\text{mín}\left(\frac{c}{c+X}, \frac{X}{c+X}\right)\right) \\ &= E(\text{mín}(1-\eta(X), \eta(X))). \end{aligned}$$

Mientras que la regla de clasificación de Bayes puede ser calculada únicamente con $\eta(x)$, para el error de Bayes L^* se requiere conocer también la distribución de X.

Supongamos que $X = c$ con probabilidad uno (como ejemplo, cuando se consideran las escuelas militares, donde todos los estudiantes son forzados a estudiar c horas por semana), entonces

$$L^* = E\left\{\frac{\text{mín}(c,X)}{c+X}\right\} = E\left(\frac{c}{c+c}\right) = E\left(\frac{1}{2}\right) = \frac{1}{2}.$$

Si en cambio X tiene distribución uniforme en $[0, 4c]$, entonces

$$\begin{aligned} L^* &= \frac{1}{4c}\int_0^{4c} \frac{\text{mín}(c,x)}{c+x}dx \\ &= \frac{1}{4c}\int_0^{c} \frac{x}{c+x}dx + \frac{1}{4c}\int_c^{4c} \frac{c}{c+x}dx \\ &= \frac{1}{4}\log\frac{5e}{4} \approx 0.305785. \end{aligned}$$

Es decir, el error de Bayes es menor cuando X tiene distribución uniforme en $[0, 4c]$ y por lo tanto funciona mejor la regla de clasificación de Bayes.

1.2.1. Decisiones plug-in.

La mejor predicción de Y para la observación X es la regla de decisión de Bayes

$$g^*(x) = \begin{cases} 0, & \text{si } \eta(x) \leq \dfrac{1}{2}, \\ 1, & \text{en otro caso.} \end{cases} = \begin{cases} 0, & \text{si } \eta(x) \leq 1 - \eta(x), \\ 1, & \text{en otro caso.} \end{cases}$$

La función η es usualmente desconocida. Asumir que tenemos acceso a funciones no negativas $\widetilde{\eta}(x)$, $1 - \widetilde{\eta}(x)$ que aproximan a $\eta(x)$ y $1 - \eta(x)$, respectivamente. En este caso parece natural utilizar la función de decisión plug-in,

$$g(x) = \begin{cases} 0, & \text{si } \widetilde{\eta}(x) \leq \dfrac{1}{2}, \\ 1, & \text{en otro caso,} \end{cases}$$

para aproximar la regla de decisión de Bayes. El siguiente teorema establece que si $\widetilde{\eta}(x)$ está cerca de la probabilidad real a posteriori en el sentido de L_1 (ver [15]), entonces la probabilidad de error de decisión g está cerca de la decisión óptima g^*.

Teorema 2. *Para la probabilidad de error de la decisión plug-in g definida anteriormente, tenemos*

$$P\{g(X) \neq Y\} - L^* = 2\int_{\mathbb{R}^d} \left|\eta(x) - \frac{1}{2}\right| I_{\{g(x) \neq g^*(x)\}} \mu(dx),$$

y

$$\begin{aligned} P\{g(X) \neq Y\} - L^* &\leq 2\int_{\mathbb{R}^d} |\eta(x) - \widetilde{\eta}(x)| \mu(dx) \\ &= 2E|\eta(x) - \widetilde{\eta}(x)|. \end{aligned}$$

Demostración. Si para alguna $x \in \mathbb{R}^d$, $g(x) = g^*(x)$, entonces la diferencia entre las probabilidades de error condicional de g y g^* es cero:

$$P\{g(X) \neq Y | X = x\} - P\{g^*(X) \neq Y | X = x\} = 0.$$

En otro caso, si $g(x) \neq g^*(x)$, entonces como se vio en la demostración del Teorema 1, la diferencia puede escribirse como

$$\begin{aligned} &P\{g(X) \neq Y | X = x\} - P\{g^*(X) \neq Y | X = x\} \\ &\qquad = (2\eta(x) - 1)\left(I_{\{g^*(x)=1\}} - I_{\{g(x)=1\}}\right) \\ &\qquad = |2\eta(x) - 1| I_{\{g(x) \neq g^*(x)\}}. \end{aligned}$$

Así,

$$\begin{aligned} P\{g(X) \neq Y\} - L^* &= \int_{\mathbb{R}^d} 2\left|\eta(x) - \frac{1}{2}\right| I_{\{g(x)\neq g^*(x)\}}\mu(dx) \\ &\leq 2\int_{\mathbb{R}^d} |\eta(x) - \widetilde{\eta}(x)|\mu(dx) \\ &= 2E|\eta(x) - \widetilde{\eta}(x)|. \end{aligned}$$

porque $g(x) \neq g^*(x)$ implica $|\eta(x) - \widetilde{\eta}(x)| \geq |\eta(x) - 1/2|$. □

1.3. Discriminación lineal.

Una **regla de discriminación lineal** o **regla de clasificación lineal**, con vector normal $a = (a_1, \ldots, a_d) \in \mathbb{R}^d$ y parámetro de localización $a_0 \in \mathbb{R}$, está dada por

$$g(x) = \begin{cases} 1, & \text{si } a^T x + a_0 > 0, \\ 0, & \text{en otro caso,} \end{cases}$$

donde $x = (x^{(1)}, \ldots, x^{(d)}) \in \mathbb{R}^d$.

La probabilidad de error del clasificador lineal dado anteriormente es denotada por $L(a, a_0)$. Sea

$$L = \inf_{a\in\mathbb{R}^d, a_0\in\mathbb{R}} L(a, a_0),$$

la probabilidad de error más pequeña dentro de los clasificadores lineales. Sean $F_{0,a}$ y $F_{1,a}$ las funciones de distribución condicionales de $a_1X^{(1)} + \cdots + a_dX^{(d)} = a^TX$ dados $Y = 0$ y $Y = 1$, respectivamente. Si $P(Y = 1) = p$ y $P(Y = 0) = 1 - p$, puede verse que

$$L = \frac{1}{2} - \sup_a \sup_x \left| pF_{1,a}(x) - (1-p)F_{0,a}(x) - p + \frac{1}{2} \right|,$$

en el caso en que $p = \dfrac{1}{2}$ se reduce a

$$L = \frac{1}{2} - \frac{1}{2}\sup_a \sup_x |F_{1,a}(x) - F_{0,a}(x)|.$$

Por lo tanto, $L = \dfrac{1}{2}$ si y sólo si $p = \dfrac{1}{2}$ y para toda a, $F_{1,a} \equiv F_{0,a}$.

Teorema 3. *$L \leq \dfrac{1}{2}$ con igualdad si y solo si $L^* = \dfrac{1}{2}$, donde L^* es el error de Bayes.*

Notar que si $L^* < \dfrac{1}{2}$ entonces $L < \dfrac{1}{2}$, por lo que una clasificación significativa por un hiperplano es posible.

Teorema 4. *Sean X_0 y X_1 variables aleatorias distribuidas como X dado $Y = 0$ y $Y = 1$, respectivamente. Sean $m_0 = E(X_0)$ y $m_1 = E(X_1)$. Definir también las matrices de covarianza*

$$\Sigma_1 = E[(X_1 - m_1)(X_1 - m_1)^T] \quad y \quad \Sigma_0 = E[(X_0 - m_0)(X_0 - m_0)^T].$$

Entonces

$$L^* \leq L \leq \inf_{a \in \mathbb{R}^d} \frac{1}{1 + \dfrac{(a^T(m_1 - m_0))^2}{((a^T\Sigma_0 a)^{1/2} + (a^T\Sigma_1 a)^{1/2})^2}}.$$

Puede verse que

$$L \leq \frac{4p(1-p)}{1 + p(1-p)\Delta^2},$$

donde $\Delta = \sqrt{(m_1 - m_0)^T\Sigma^{-1}(m_1 - m_0)}$ es la distancia de Mahalanobis entre m_0 y m_1, con $\Sigma = p\Sigma_1 + (1-p)\Sigma_0$ (ver [11]). Esto significa que una distancia de Mahalanobis grande implica una probabilidad de error pequeña para clasificadores lineales, y por lo tanto también para la regla de clasificación de Bayes.

1.3.1. Discriminate lineal de Fisher.

Existen varios métodos para obtener el vector normal a de una regla de clasificación. Uno de los primeros métodos fue propuesto por Fisher (1936). Sean $\widehat{m}_1$ y $\widehat{m}_0$ las medias muestrales de las dos clases, es decir,

$$\widehat{m}_1 = \sum_{i:Y_i=1} \frac{X_i}{n_1} \quad \text{y} \quad \widehat{m}_0 = \sum_{i:Y_i=0} \frac{X_i}{n_0}, \text{ con } n_j = \#\{i : Y_i = j\},\ j = 0, 1.$$

Sean $\widehat{\sigma}_1^2$ y $\widehat{\sigma}_0^2$ dispersiones muestrales de las proyecciones en la dirección de a para las clases 1 y 0, respectivamente, es decir

$$\widehat{\sigma}_j^2 = \sum_{i:Y_i=j} (a^T X_i - a^T \widehat{m}_j)^2 = a^T S_j a,$$

donde

$$S_j = \sum_{i:Y_i=j} (X_i - \widehat{m}_j)(X_i - \widehat{m}_j)^T,$$

para $j = 0, 1$. Es decir, S_0 y S_1 son las matrices de dispersión muestrales de la clase 0 y 1, respectivamente.

El **discriminante lineal de Fisher** es la función lineal $a^T x$ para el cual

$$J(a) = \frac{(a^T\widehat{m}_1 - a^T\widehat{m}_0)^2}{\widehat{\sigma}_1^2 + \widehat{\sigma}_0^2} = \frac{(a^T(\widehat{m}_1 - \widehat{m}_0))^2}{a^T(S_1 + S_0)a},$$

es máxima. Esto corresponde a encontrar la dirección que mejor separa a $a^T\widehat{m}_1$ de $a^T\widehat{m}_0$ relativo a las dispersiones muestrales. La solución está dada por

$$a = (S_1 + S_0)^{-1}(\widehat{m}_1 - \widehat{m}_0).$$

La regla de discriminación de Fisher está dada por

$$g_{a_0}(x) = \begin{cases} 1, & \text{si } a^T x + a_0 > 0, \\ 0, & \text{en otro caso,} \end{cases}$$

para alguna constante $a_0 \in \mathbb{R}$.

Desafortunadamente, el discriminante lineal de Fisher puede se arbitrariamente malo: existen distribuciones tales que aún cuando las clases son linealmente separables (es decir $L = 0$), el discriminante lineal de Fisher tiene una probabilidad de error cercana a 1.

Caso de la distribución normal.

Hay algunas situaciones en las cuales la regla de Bayes es un clasificador lineal. Esto sucede si la distribución de las clases es normal multivariada.

La distribución normal multivariada tiene densidad dada por

$$f(x) = \frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}} \exp\left(-\frac{1}{2}(x-m)^T\Sigma^{-1}(x-m)\right),$$

donde m es la media d-dimensional, Σ es la matriz de covarianza de tamaño $d \times d$, Σ^{-1} es la inversa de Σ, y $\det(\Sigma)$ es su determinante. Si X tiene densidad f, entonces $m = E(X)$ y $\Sigma = E[(X-m)(X-m)^T]$, simbólicamente $X \sim N_d(m, \Sigma)$. La densidad normal multivariada está completamente especificada por sus parámetros m y Σ.

En el caso de dos clases en el cual X tiene densidad $f(x) = pf_1(x)+(1-p)f_0(x)$ y f_0, f_1 ambas son normales multivariadas con parámetros m_i, Σ_i, para $i = 0, 1$, la regla de Bayes puede ser escrita como

$$g^*(x) = \begin{cases} 1, & \text{si } pf_1(x) > (1-p)f_0(x), \\ 0, & \text{en otro caso.} \end{cases}$$

Notamos que $p = P(Y = 1)$, $1 - p = P(Y = 0)$ y que f_0 y f_1 son las funciones de densidad de X dado $Y = 0$ y $Y = 1$, respectivamente. Para ver que g^* tiene la expresión anterior, recordemos que

$$g^*(x) = \begin{cases} 1, & \text{si } \eta(x) > \dfrac{1}{2}, \\ \\ 0, & \text{en otro caso,} \end{cases}$$

donde $\eta(x) = P(Y = 1|X = x) = E(Y|X = x)$. Además notar que

$$\begin{aligned} \eta(x) &= P(Y = 1|X = x) = E(Y|X = x) \\ &= \frac{pf_1(x)}{pf_1(x) + (1-p)f_0(x)} > \frac{1}{2} \\ &\Leftrightarrow 2pf_1(x) > pf_1(x) + (1-p)f_0(x) \\ &\Leftrightarrow pf_1(x) > (1-p)f_0(x). \end{aligned}$$

Ahora $g^*(x) = 1$ si y sólo si

$$\begin{aligned} \log(pf_1(x)) > \log((1-p)f_0(x)) &\Leftrightarrow \log p - \frac{1}{2}\log(\det(\Sigma_1)) - \frac{1}{2}(x-m_1)^T\Sigma_1^{-1}(x-m_1) \\ &> \log(1-p) - \frac{1}{2}\log(\det(\Sigma_0)) - \frac{1}{2}(x-m_0)^T\Sigma_0^{-1}(x-m_0) \\ &\Leftrightarrow (x-m_1)^T\Sigma_1^{-1}(x-m_1) - 2\log p + \log(\det(\Sigma_1)) \\ &< (x-m_0)^T\Sigma_0^{-1}(x-m_0) - 2\log(1-p) + \log(\det(\Sigma_0)). \end{aligned}$$

Sea $r_i^2 = (x-m_i)^T\Sigma_i^{-1}(x-m_i)$ la distancia de Mahalanobis al cuadrado de x a m_i, para $i = 0, 1$. La última desigualdad es equivalente a

$$r_1^2 < r_0^2 - 2\log\left(\frac{1-p}{p}\right) + \log\left(\frac{\det(\Sigma_0)}{\det(\Sigma_1)}\right).$$

Así la regla de Bayes es simplemente

$$g^*(x) = \begin{cases} 1, & \text{si } r_1^2 < r_0^2 - 2\log\left(\dfrac{1-p}{p}\right) + \log\left(\dfrac{\det(\Sigma_0)}{\det(\Sigma_1)}\right), \\ 0, & \text{en otro caso.} \end{cases}$$

En particular, cuando $p = \dfrac{1}{2}$, $\Sigma_0 = \Sigma_1 = \Sigma$ tenemos que

$$g^*(x) = \begin{cases} 1, & \text{si } r_1^2 < r_0^2, \\ 0, & \text{en otro caso.} \end{cases}$$

Obsérvese que si $\Sigma_0 = \Sigma_1 = \Sigma$, entonces haciendo que $a^T = 2(m_1 - m_0)^T\Sigma^{-1}$ y $a_0 = 2\log\left(\dfrac{p}{1-p}\right) + m_0^T\Sigma^{-1}m_0 - m_1^T\Sigma^{-1}m_1$, la regla de decisión de Bayes es el discriminante lineal

$$g^*(x) = \begin{cases} 1, & \text{si } a^Tx + a_0 > 0, \\ 0, & \text{en otro caso.} \end{cases}$$

1.4. Consistencia Universal.

Si tenemos una sucesión $D_n = ((X_1, Y_1), \ldots, (X_n, Y_n))$ de datos de entrenamiento, lo mejor que se puede esperar de una función de clasificación es lograr la probabilidad de error de Bayes L^*. Generalmente, no podemos obtener una función que logre exactamente la probabilidad de error de Bayes, pero es posible construir una sucesión de funciones de clasificación $\{g_n\}$, es decir una regla de clasificación, tal que el error de probabilidad

$$L_n = L(g_n) = P[g_n(X, D_n) \neq Y | D_n]$$

se acerque arbitrariamente a L^* con probabilidad cercana a 1.

Definición 1 (Consistencia débil y fuerte)**.** *Una regla de clasificación es* ***consistente*** *(o asintóticamente eficiente con la probabilidad de error de Bayes) para una cierta distribución de* (X, Y) *si*

$$E(L_n) = P[g_n(X, D_n) \neq Y] \to L^* \text{ cuando } n \to \infty,$$

y ***fuertemente consistente*** *si*

$$\lim_{n\to\infty} L_n \to L^* \text{ con probabilidad } 1.$$

Observación. *La consistencia esta definida como la convergencia del valor esperado de* L_n *a* L^**. Como* L_n *es una variable aleatoria con valores entre* L^* *y* 1*, esta convergencia es equivalente a la convergencia de* L_n *a* L^* *en probabilidad, lo que significa que para cada* $\epsilon > 0$

$$\lim_{n\to\infty} P[L_n - L^* > \epsilon] = 0.$$

Dado que la convergencia casi segura implica convergencia en probabilidad, consistencia fuerte implica consistencia.

Capítulo 2

Redes Neuronales.

2.1. Perceptrón multicapa.

El discriminante lineal o perceptrón tiene la regla de decisión

$$\phi(x) = \begin{cases} 0, & \text{si } \psi(x) \leq \frac{1}{2}, \\ 1, & \text{en otro caso,} \end{cases}$$

basada en una combinación lineal $\psi(x)$

$$\psi(x) = c_0 + \sum_{i=1}^{d} c_i x^{(i)} = c_0 + c^T x, \tag{2.1}$$

donde las c_i's son los pesos, $x = (x^{(1)}, \ldots, x^{(d)})^T$ y $c = (c_1, \ldots, c_d)^T$. Esto es llamado una **red neuronal sin capas ocultas**, ver Figura 2.1.

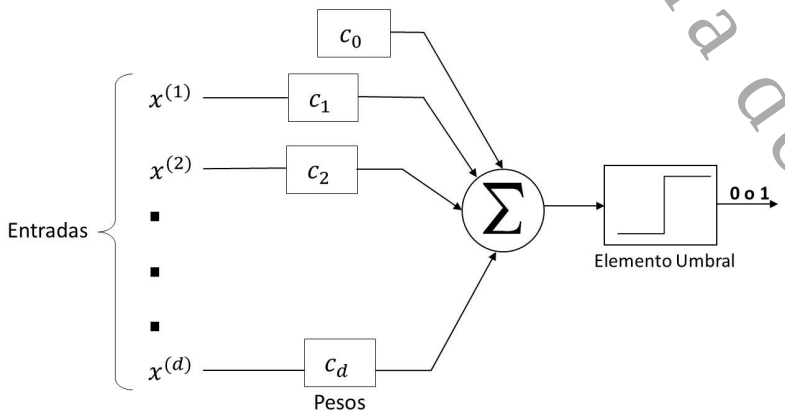


Figura 2.1: El perceptrón de Rosenblatt. La decisión se basa en una combinación lineal de los componentes del vector de entrada.

En una **red neuronal (hacia adelante)** o **perceptrón multicapa** con una capa oculta se tiene

$$\psi(x) = c_0 + \sum_{i=1}^{k} c_i \sigma(\psi_i(x)), \tag{2.2}$$

donde las c_i's son como antes y cada ψ_i es de la forma dada en (2.1): $\psi_i(x) = b_i + \sum_{j=1}^{d} a_{ij} x^{(j)}$ para algunas constantes b_i y a_{ij}, ver Figura 2.2. La función σ es llamada sigmoide, la cual se define como una función no decreciente con $\sigma \to -1$ cuando $x \downarrow -\infty$ y $\sigma \to 1$ cuando $x \uparrow \infty$. A continuación se muestran algunos ejemplos:

1. Sigmoide umbral

$$\sigma(x) = \begin{cases} -1, & \text{x} \leq 0, \\ 1, & \text{si } x > 0; \end{cases}$$

2. Sigmoide estándar o logístico

$$\sigma(x) = \frac{1 - e^{-x}}{1 + e^{-x}};$$

3. Sigmoide arcotangente

$$\sigma(x) = \frac{2}{\pi} \arctan(x);$$

4. Sigmoide gaussiano

$$\sigma(x) = 2 \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}} du - 1.$$

En la Figura 2.3 se presentan las gráficas de estos sigmoides.

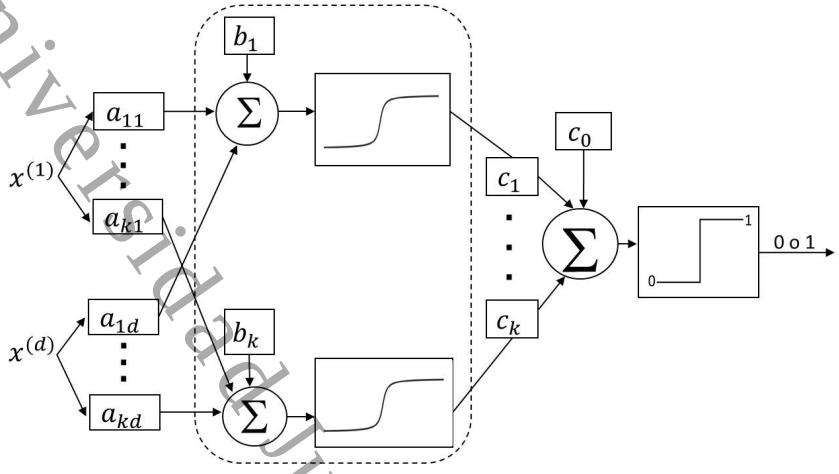


Figura 2.2: Red neuronal con una capa oculta. Las neuronas ocultas se encuentran dentro del marco.

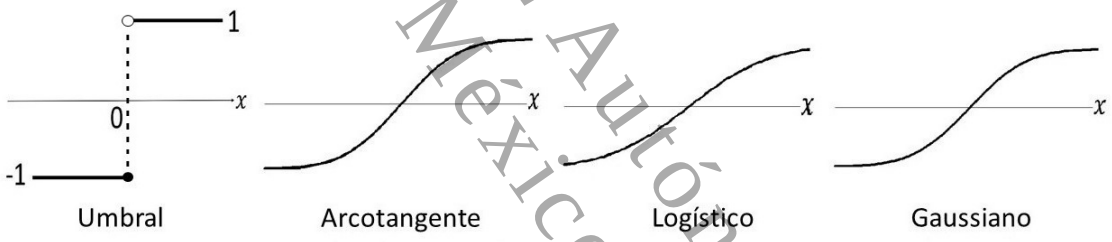


Figura 2.3: Gráficas de sigmoides.

En el perceptrón con una capa oculta, decimos que hay k neuronas ocultas (la salida de la i-ésima neurona oculta es $u_i = \sigma(\psi_i(x))$). Así, (2.2) puede reescribirse como

$$\psi(x) = c_0 + \sum_{i=1}^{k} c_i u_i,$$

que es similar a (2.1). Podemos continuar este proceso y crear redes neuronales multicapa hacia adelante. Por ejemplo, un perceptrón de dos capas ocultas es de la forma

$$\psi(x) = c_0 + \sum_{i=1}^{\ell} c_i z_i,$$

donde

$$z_i = \sigma\left(d_{i0} + \sum_{j=1}^{k} d_{ij} u_j\right) \ , \ 1 \leq i \leq \ell,$$

y

$$u_j = \sigma\left(b_j + \sum_{i=1}^{d} a_{ji}x^{(i)}\right) \ , 1 \leq j \leq k,$$

y las d_{ij}'s, b_j's y a_{ji}'s son constantes. La primera capa oculta tiene k neuronas ocultas, mientras que la segunda capa oculta tiene ℓ neuronas ocultas, ver Figura 2.4.

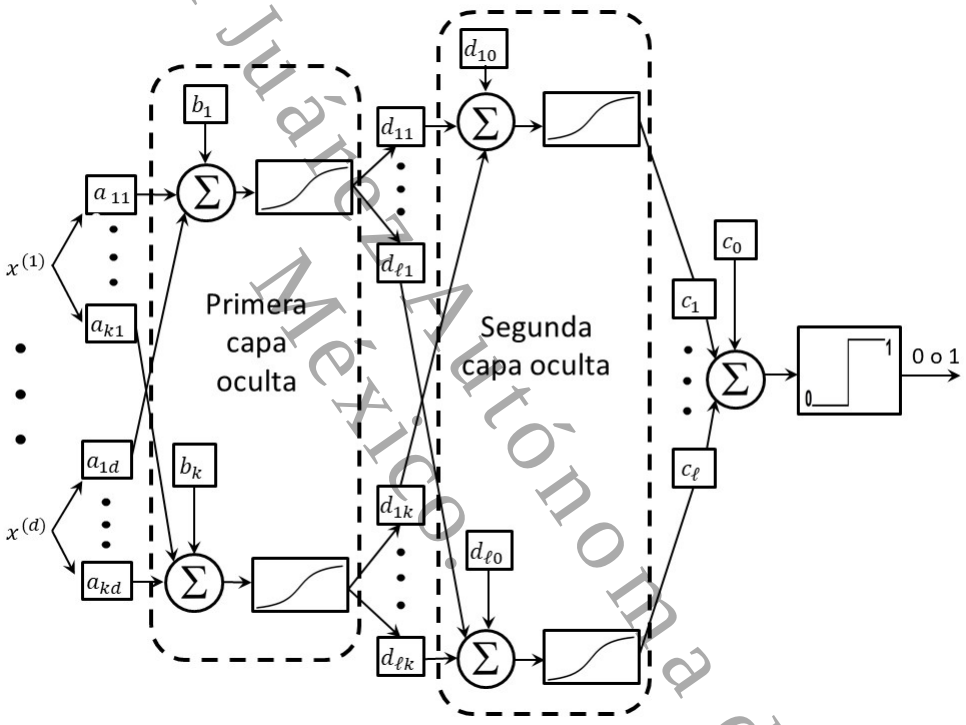


Figura 2.4: Red neuronal hacia adelante con dos capa ocultas.

2.2. Arreglos.

Un conjunto finito $\mathcal{A}$ de hiperplanos en $\mathbb{R}^d$ particiona el espacio en piezas poliédricas convexas conectadas. Tal partición $\mathcal{P} = \mathcal{P}(\mathcal{A})$ es llamada un arreglo. Un arreglo es llamado **simple** si cualesquiera d hiperplanos de $\mathcal{A}$ tienen un único punto en común y si $d+1$ hiperplanos no tienen puntos en común, ver Figura 2.5.

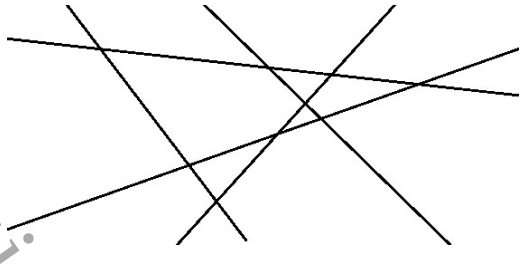


Figura 2.5: Un arreglo simple de 5 rectas en el plano.

Un arreglo simple crea celdas poliédricas. Curiosamente el número de celdas es independiente de la configuración de los hiperplanos. En particular, el número de celdas es exactamente 2^k si $d \geq k$, y

$$\sum_{i=0}^{d} \binom{k}{i} \text{ si } d < k,$$

donde $|\mathcal{A}| = k$, ver [3]. Para arreglos más generales los valores anteriores son simplemente cotas superiores.

Podemos usar arreglos para diseñar clasificadores. Definamos $g_{\mathcal{A}}$ como el **clasificador natural** obtenido por la mayoría de votos sobre todas las Y_i's para las cuales las X_i's están en la misma celda del arreglo $\mathcal{P} = \mathcal{P}(\mathcal{A})$ que un nuevo punto x, ver Figura 2.6.

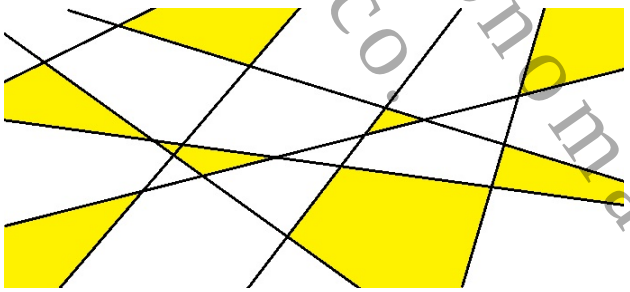


Figura 2.6: Un clasificador basado en arreglos. Las áreas sombreadas son donde se obtuvo la mayoría de votos para la clase 1 y las áreas sin sombrear son donde se obtuvo la mayoría de votos para la clase 0.

Si fijamos k y encontramos un conjunto de hiperplanos $\mathcal{A}$ con $|\mathcal{A}| = k$, para el cual el error empírico

$$L_n^{(R)} = \frac{1}{n} \sum_{i=1}^{n} I_{\{g_{\mathcal{A}}(X_i) \neq Y_i\}}$$

es mínimo, obtenemos el **clasificador de riesgo empírico óptimo**.

Teorema 5. *El clasificador de riesgo empírico óptimo basado en arreglos $\mathcal{P}(\mathcal{A})$ con $|\mathcal{A}| \leq k$ cumple que $E\{L_n\} \to L^*$ para todas las distribuciones si $k \to \infty$ y $k = o(n/\log n)$, donde L^* es el error de Bayes, es decir, la probabilidad de error de clasificación de la regla de decisión de Bayes.*

Recordemos que la regla de clasificación de Bayes es la regla de decisión óptima, es decir, la que tiene probabilidad de error de clasificación mínima. También se pueden hacer arreglos a partir de los datos disponibles de una manera más simple. Fijar k puntos $X_1, X_2, \ldots, X_k$ en posición general y considerar todos los posibles $\binom{k}{d}$ hiperplanos que se pueden formar con estos puntos, ver Figura 2.7. Estos hiperplanos forman una colección $\mathcal{A}$, que define un arreglo. No se realiza ningún tipo de optimización, se toma el clasificador natural obtenido por mayoría de votos dentro de las celdas de la partición sobre $(X_{k+1}, Y_{k+1}), \ldots, (X_n, Y_n)$.

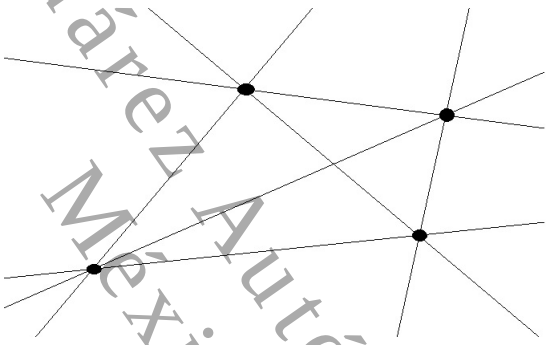


Figura 2.7: Arreglo determinado por $k = 4$ puntos en el plano

Aquí no podemos aplicar el Teorema 5 de consistencia. También el arreglo ya no es simple, porque no cumple que $d + 1$ hiperplanos no tienen puntos en común. La regla así obtenida es consistente, en términos de la Definición 1, si $\text{diam}(A(X)) \to 0$ en probabilidad y el número de celdas es $o(n)$, donde $A(X)$ es la celda a la que pertenece X en el arreglo y $\text{diam}(B)$ es el diámetro de un conjunto B, es decir $\text{diam}(B) = \sup_{x,y \in B} \| x - y \|$. Como el número de celdas es a lo más

$$\sum_{i=0}^{d} \binom{k'}{i}$$

donde $k' = \binom{k}{d}$, vemos que el número de celdas dividido por n tiende a cero si

$$\frac{k^{d^2}}{n} \to 0.$$

Esto pone una restricción severa en el crecimiento de k. El siguiente resultado permitirá proporcionar condiciones para la consistencia del clasificador por arreglos definido anteriormente.

Lema 1. *Si $k \to \infty$, entonces* $\text{diam}(A(X)) \to 0$ *en probabilidad cuando X tiene una densidad.*

Demostración. Puede mostrarse que si μ es una medida de probabilidad con densidad f en $\mathbb{R}^d$, el conjunto de todas las x para las cuales para todo $\epsilon > 0$ se tiene $\mu(x + \epsilon Q_i) > 0$, para todos los cuadrantes $Q_1, Q_2, \ldots, Q_{2^d}$ teniendo un vértice en $(0, 0, \ldots, 0)$ y lados de longitud uno, tiene μ-medida uno. Aquí, por ejemplo, $x + \epsilon Q_1 = [x_1, x_1 + \epsilon] \times \cdots \times [x_d, x_d + \epsilon]$, para $x = (x_1, x_2, \ldots, x_d)$. Para tales x's, si al menos una de las X_i's ($i \leq k$) cae en cada uno de los 2^d cuadrantes $x + \epsilon Q_i$, entonces $\text{diam}(A(x)) \leq 2d\epsilon$, ver Figura 2.8. Esto es debido a que la celda $A(x)$ a la cual pertenece x está contenida en el conjunto convexo formado por esos 2^d puntos, y ese conjunto convexo al estar contenido en un d-cubo C de 2^d vértices y de aristas de longitud 2ϵ, tiene diámetro menor o igual a $\text{diam}(C) \leq 2d\epsilon$. Por lo que $A(x) \leq 2d\epsilon$.

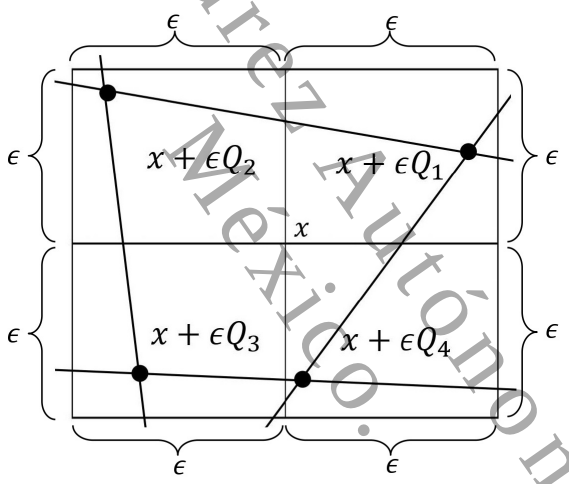


Figura 2.8: El diámetro de la celda que contiene x es menor que 4ϵ si hay un dato en cada uno de los cuatro cuadrantes de lados de longitud ϵ alrededor de x.

Por lo tanto, para $\epsilon > 0$ arbitrario, tenemos que

$$P(dim(A(x)) > 2d\epsilon) \leq P(\text{alguno de los conjuntos } x + \epsilon Q_j \text{ no contiene a ninguno de los } k \text{ datos}).$$

Notar que la probabilidad de que $x + \epsilon Q_j$ no contiene a ninguno de los k datos es igual a $(1 - \mu(x + \epsilon Q_j))^k$, para $j = 1, 2, \ldots, 2^d$. Además

$$(1 - \mu(x + \epsilon Q_j))^k \leq \left(1 - \min_{1 \leq i \leq 2^d} \mu(x + \epsilon Q_i)\right)^k, \quad j = 1, 2, \ldots, 2^d.$$

Como la probabilidad de la unión de eventos es menor o igual a la suma de las probabilidades de los eventos (desigualdad de Boole, ver [6]), tenemos que

$$P\left\{\text{diam}(A(x)) > 2d\epsilon\right\} \leq 2^d \left(1 - \min_{1\leq i\leq 2^d} \mu(x + \epsilon Q_i)\right)^k \to 0.$$

La convergencia a cero del lado derecho de la desigualdad anterior es por lo mencionado al inicio de la demostración, ya que el conjunto de las x para las cuales $1 - \min_{1\leq i\leq 2^d} \mu(x + \epsilon Q_i)$ es estrictamente menor que 1 tiene μ-medida uno. Así, por el Teorema de convergencia dominada de Lebesgue

$$P(diam(A(X)) > 2d\epsilon) \to 0.$$

□

Como una consecuencia del lema anterior y bajo ciertas condiciones adicionales, se tiene el siguiente resultado.

Teorema 6. *El clasificador por arreglos definido anteriormente es consistente siempre que X tenga una densidad y*

$$k \to \infty,\ k^{d^2} = o(n).$$

El teorema anterior señala que la minimización del error empírico sobre un conjunto finito de arreglos también puede ser consistente. Tal conjunto puede tomarse como la colección de arreglos que consisten en hiperplanos que pasan a través de d puntos de $X_1, \cdots, X_k$. El clasificador de riesgo empírico óptimo de este conjunto será consistente, ver Problema 30.1 de [3].

En una computadora se sigue la siguiente estrategia con los arreglos. Claramente, para hallar la celda a la que pertenece un dato x encontramos para cada hiperplano $A \in \mathcal{A}$ el lado al que pertenece x. Si $f(x) = a^T x + a_0$, entonces $f(x) > 0$ si x está de un lado del hiperplano, $f(x) = 0$ si x está sobre el hiperplano, y $f(x) < 0$ si x está en el otro lado del hiperplano. Si $\mathcal{A} = \{A_1, \ldots, A_k\}$, el vector $\left(I_{\{H_1(x)>0\}}, \ldots, I_{\{H_k(x)>0\}}\right)$ describe la celda a la que pertenece x, donde $H_i(x)$ es una función lineal que es positiva si x esta en un lado del hiperplano A_i, negativo si x está del otro lado de A_i, y 0 si $x \in A_i$. Es fácil ver que el clasificador representado en la figura 2.9 es idéntico al clasificador por arreglos. En la terminología de redes neuronales, la primera capa oculta de neuronas corresponde justamente a k perceptrones (y tiene $k(d+1)$ pesos o parámetros). La primera capa genera un vector de k entradas binarias que señala la ubicación precisa de x en las celdas del arreglo. La segunda capa asigna una clase (decisión -1 o $+1$) a cada celda del arreglo y activa una sola neurona (asignándole 0 a las neuronas que no activa y 1 a la neurona que activa). Se tienen 2^k neuronas (para la asignación de clase), pero como se vio anteriormente, en los clasificadores naturales estas neuronas no requieren entrenamiento o aprendizaje; la decisión se toma por mayoría de votos.

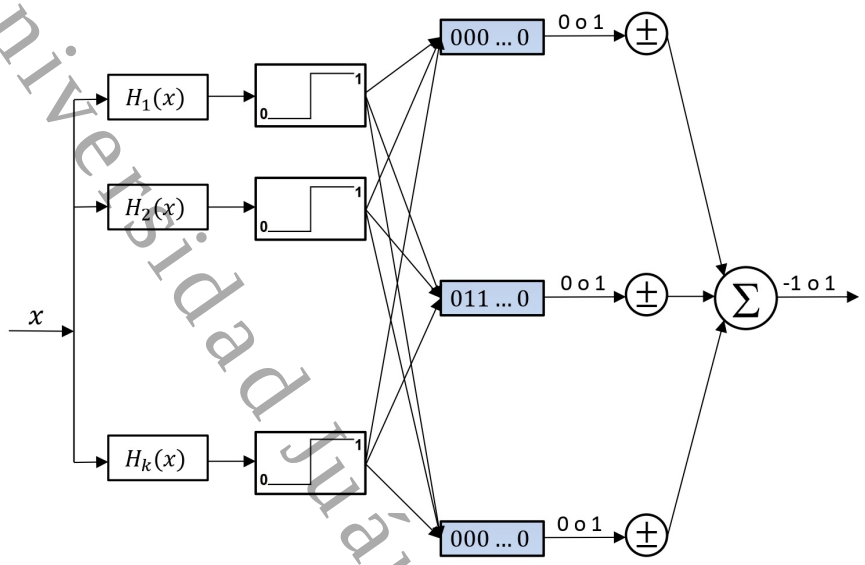


Figura 2.9: Clasificador por arreglos realizado por una red neuronal de dos capas ocultas. Cada una de las 2^k celdas de la segunda capa oculta realiza una operación "y": la salida del nodo "101" es 1 si las tres entradas son 1, 0 y 1, respectivamente. De lo contrario, su salida es 0. Por lo tanto, una y sola una de las 2^k salidas es 1.

Sea x un nuevo vector de entradas. Sea $H_i(x) = a_i^T x + a_{i0}$, la función lineal que es positiva si x está de un lado del hiperplano A_i, y 0 si $x \in A_i$. Sea

$$b_j = \sigma(H_j(x)) = \sigma(a_j^T x + a_{j0}) = \sigma\left(\sum_{i=1}^{d} a_{ji} x^{(i)} + a_{j0}\right),$$

donde σ es el sigmoide umbral, por lo que $b = (b_1, \ldots, b_k)$ es un vector de dimensión k cuyas entradas son -1 o 1, y que describe la celda del arreglo a la que pertenece x.

Cada vector de dimensión k cuyas entradas son -1 o 1 corresponde a una celda del arreglo. Sea $c_\ell = (c_{\ell 1}, \ldots, \ldots c_{\ell k})$ el vector cuyas entradas son -1 o 1 correspondiente a la celda ℓ, para $\ell = 1, 2, \ldots, 2^k$. Sea

$$z_\ell = \sigma\left(\sum_{j=1}^{k} c_{\ell j} b_j - k + \frac{1}{2}\right) = \sigma\left(\sum_{j=1}^{k} c_{\ell j} b_j + c_{\ell 0}\right), \tag{2.3}$$

donde $c_{\ell 0} = -k + \frac{1}{2}$ y σ es el sigmoide umbral. El argumento de la función sigmoide es igual a $\frac{1}{2}$ si $b = c_\ell$ y es negativo de otro modo. Por lo tanto, $z_\ell = 1$ si $b = c_\ell$, y $z_\ell = -1$ si $b \neq c_\ell$.

Asumimos que tomamos una regla de decisión basada en el signo de

$$\sum_{\ell=1}^{2^k} w_\ell z_\ell + w_0,$$

donde las w_ℓ's son pesos y las z_ℓ's son las salidas de la segunda capa oculta, dadas por (2.3). Supongamos que deseamos asignar la clase 1 a s celdas del arreglo y la clase 0 a las t celdas restantes. Para una región ℓ de la clase 1, sea $w_\ell = 1$; para una región ℓ de la clase 0, sea $w_\ell = -1$. Definir $w_0 = 1 + s - t$. Si para algún $j \in \{1, 2, \ldots, 2^k\}$, $z_j = 1$ y $z_i = -1$ $\forall i \neq j$, entonces, debido a que $\sum_{r=1}^{2^k} w_r = s - t$, se tiene

$$\begin{aligned}
\sum_\ell w_\ell z_\ell + w_0 &= w_j + w_0 - \sum_{i \neq j} w_i \\
&= w_j + (1 + s - t) - (s - t - w_j) \\
&= 2w_j + 1 \\
&= \begin{cases} 3, & \text{si } w_j = 1, \\ -1, & \text{si } w_j = -1. \end{cases}
\end{aligned}$$

Por lo tanto, la regla de decisión es

$$\phi(x) = \begin{cases} 1, & \text{si } \sum_\ell w_\ell z_\ell + w_0 > 0, \\ 0, & \text{si } \sum_\ell w_\ell z_\ell + w_0 \leq 0, \end{cases}$$

donde

$$z_\ell = \sigma\left(\sum_{j=1}^{k} c_{\ell j} b_j + c_{\ell 0}\right), \ 1 \leq \ell \leq 2^k;$$

y

$$b_\ell = \sigma\left(\sum_{i=1}^{d} a_{ji} x^{(i)} + a_{j0}\right), \ 1 \leq j \leq k;$$

es decir, este clasificador por arreglos es una red neuronal con dos capas ocultas, la primer capa oculta con k neuronas y la segunda capa oculta con 2^k neuronas.

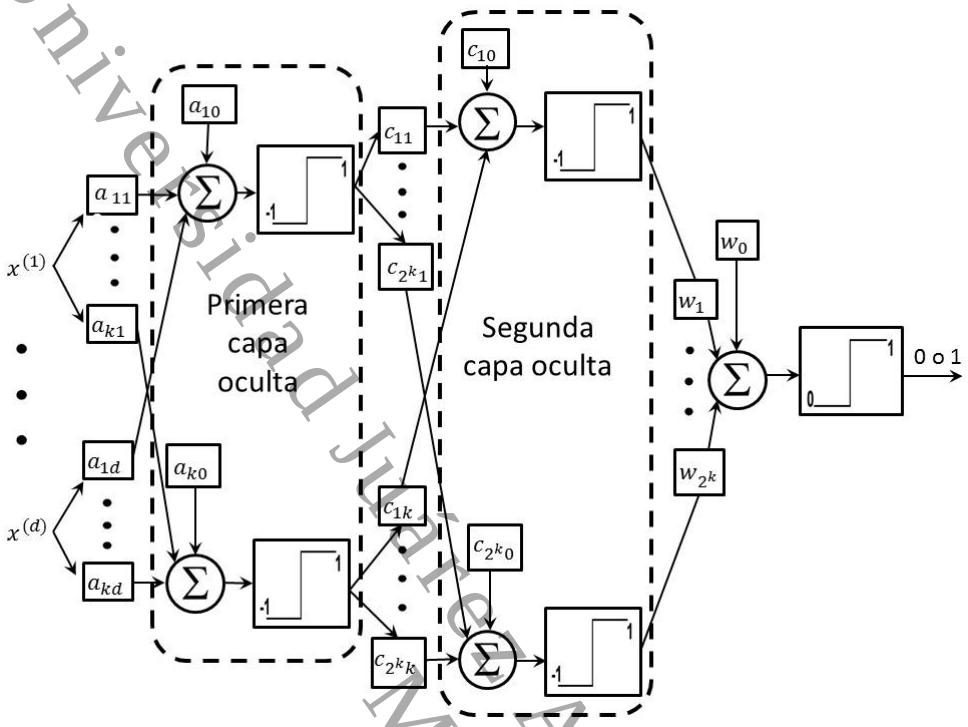


Figura 2.10: Red neuronal del clasificador por arreglos.

2.3. Aproximación por redes neuronales.

Considere primero la clase $\mathcal{C}^{(k)}$ de clasificadores (2.2) que contiene a todos los clasificadores por redes neuronales con sigmoide umbral y k neuronas ocultas en dos capas ocultas. Los datos de entrenamiento D_n son usados para seleccionar un clasificador en $\mathcal{C}^{(k)}$. Para un buen desempeño de la regla seleccionada, es necesario que la mejor regla en $\mathcal{C}^{(k)}$ tenga probabilidad de error cercana a L^*, esto es que

$$\inf_{\phi \in \mathcal{C}^{(k)}} L(\phi) - L^*$$

sea pequeña. Llamamos a esta cantidad el **error de aproximación**. Naturalmente para k fijo el error de aproximación es positivo para la mayoría de las distribuciones. Sin embargo, para k grande se espera que sea pequeño. La pregunta es si la última afirmación es verdadera para todas la distribuciones de (X, Y). Vimos en la sección anterior que la clase de clasificadores por redes neuronales de dos capas ocultas con m nodos en la primera capa y 2^m nodos en la segunda capa contiene todos los clasificadores por arreglos con m hiperplanos. Por lo tanto, para $k = m + 2^m$ la clase de todos los clasificadores por arreglos con m hiperplanos es una subclase de $\mathcal{C}^{(k)}$. De esto se obtiene fácilmente el siguiente resultado de aproximación.

Teorema 7. *Si $\mathcal{C}^{(k)}$ es la clase de todos los clasificadores por redes neuronales con sigmoide umbral y k neuronas en dos capas ocultas,*

$$\lim_{k\to\infty} \inf_{\phi\in\mathcal{C}^{(k)}} L(\phi) - L^* = 0$$

para todas las distribuciones de (X, Y).

Es interesante ver que la misma propiedad se mantiene si $\mathcal{C}^{(k)}$ es la clase de redes neuronales de una capa oculta con k neuronas ocultas y sigmoide arbitrario. Más precisamente, sea $\mathcal{C}^{(k)}$ la clase de clasificadores

$$\phi(x) = \begin{cases} 0, & \text{si } \psi(x) \leq \dfrac{1}{2}; \\ 1, & \text{en otro caso,} \end{cases}$$

donde $\psi(x)$ es como en (2.2). Por el Teorema 2 tenemos

$$L(\phi) - L^* \leq 2E\{|\psi(X) - \eta(X)|\},$$

donde $\eta(x) = P\{Y = 1|X = x\}$. Así,

$$\inf_{\phi\in\mathcal{C}^{(k)}} L(\phi) - L^* \to 0$$

cuando $k \to \infty$ si $E\{|\psi_k(x) - \eta(x)|\} \to 0$, para alguna sucesión $\{\psi_k\}$, con $\phi_k \in \mathcal{C}^{(k)}$ y $\phi_k(x) = I_{\{\psi_k(x)>1/2\}}$. Por la consistencia universal vista en la Sección 1.4, necesitamos asegurarnos de que la familia de las ψ's puede aproximarse a cualquier η en el sentido $L_1(\mu)$. En otras palabras, el error de aproximación $\inf_{\phi\in\mathcal{C}^{(k)}} L(\phi) - L^*$ converge a cero si la clase de funciones ψ es denso en $L_1(\mu)$ para cada μ. Otra condición suficiente para esto (pero demasiado severa) es que la clase $\mathcal{F}$ de funciones ψ llegue a ser denso con la norma L_∞ en el espacio de funciones continuas $C[a, b]^d$ en $[a, b]^d$, donde $[a, b]^d$ denota el hiperrectángulo de $\mathbb{R}^d$ definido por sus vértices opuestos a y b, para cualesquiera a y b.

Lema 2. *Supóngase que una sucesión de clases de funciones $\mathcal{F}_k$ se vuelve denso con la norma L_∞ en el espacio de funciones continuas en $[a, b]^d$, es decir,*

$$\lim_{k\to\infty} \inf_{f\in\mathcal{F}_k} \sup_{x\in[a,b]^d} |f(x) - g(x)| = 0.$$

Entonces para cualquier distribución de (X, Y),

$$\lim_{k\to\infty} \inf_{\phi\in\mathcal{C}^{(k)}} L(\phi) - L^* = 0,$$

donde $\mathcal{C}^{(k)}$ es la clase de clasificadores $\phi(x) = I_{\{\psi(x)>1/2\}}$ para $\psi \in \mathcal{F}_k$.

Demostración. Para $\epsilon > 0$ fijo, encontramos a, b tales que $\mu([a,b]^d) \geq 1 - \epsilon/3$, donde μ es la medida de probabilidad de X. Elegimos una función continua $\widehat{\eta}$ que se hace cero fuera de $[a,b]^d$ tal que

$$E\{|\eta(X) - \widehat{\eta}(X)|\} \leq \frac{\epsilon}{6}.$$

Encontramos k y $f \in \mathcal{F}_k$ tales que

$$\sup_{x \in [a,b]^d} |f(x) - \widehat{\eta}(x)| \leq \frac{\epsilon}{6}.$$

Para $\phi(x) = I_{\{f(x)>1/2\}}$, y por el Teorema 2, tenemos

$$\begin{aligned} L(\phi) - L^* &\leq 2E\{|f(X) - \eta(x)|I_{\{X \in [a,b]^d\}}\} + \frac{\epsilon}{3} \\ &\leq 2E\{|f(X) - \widehat{\eta}(X)|I_{\{X \in [a,b]^d\}}\} + 2E\{|\widetilde{\eta}(X) - \eta(X)|\} + \frac{\epsilon}{3} \\ &\leq 2 \sup_{x \in [a,b]^d} |f(x) - \widetilde{\eta}(x)| + 2E\{|\widehat{\eta}(X) - \eta(X)|\} + \frac{\epsilon}{3} \\ &\leq \frac{\epsilon}{3} + \frac{\epsilon}{3} + \frac{\epsilon}{3} = \epsilon. \end{aligned}$$

□

A continuación el siguiente teorema muestra una aproximación por redes neuronales con una capa oculta.

Teorema 8. *Para cada función continua $f : [a,b]^d \to \mathbb{R}$ y para cada $\epsilon > 0$, existe una red neuronal con una capa oculta y una función $\psi(x)$ como en* (2.2) *tales que*

$$\sup_{x \in [a,b]^d} |f(x) - \psi(x)| < \epsilon.$$

Demostración. La prueba se hará sólo para el sigmoide umbral

$$\sigma(x) = \begin{cases} -1, & \text{si } x \leq 0; \\ 1, & \text{si } x > 0. \end{cases}$$

Sea $\epsilon > 0$ fijo. Tomamos la aproximación por series de Fourier de $f(x)$; y por el Teorema de Stone-Wierstrass, ver [3], existen un entero positivo grande M, coeficientes reales distintos de cero $\alpha_1, \alpha_2, \ldots, \alpha_M$, $\beta_1, \beta_2, \ldots, \beta_M$, y enteros $m_{i,j}$, para $i = 1, 2, \ldots, M$, $j = 1, 2, \ldots, d$, tales que

$$\sup_{x \in [a,b]^d} \left| \sum_{i=1}^{M} \left(\alpha_i \cos\left(\frac{\pi}{a} m_i^T x\right) + \beta_i \operatorname{sen}\left(\frac{\pi}{a} m_i^T x\right)\right) - f(x) \right| < \frac{\epsilon}{2},$$

donde $m_i = (m_{i,1}, \ldots, m_{i,d})$. Toda función continua en la recta real que es cero fuera de un intervalo acotado puede ser arbitrariamente aproximada de manera

uniforme en el intervalo mediante redes neuronales unidimensionales, es decir, por funciones de la forma

$$\sum_{i=1}^{k} c_i \sigma(a_i x + b_i) + c_0.$$

Observe que la función indicadora de un intervalo $[b, c]$ puede ser escrita como $\sigma(x - b) + \sigma(-x + c)$. Esto implica que funciones acotadas como seno y coseno pueden ser aproximadas de manera arbitraria por redes neuronales. Por lo que, existen redes neuronales $u_i(x)$, $v_i(x)$, con $i = 1, 2, \ldots, M$, (i.e. mapeos de $\mathbb{R}^d$ a $\mathbb{R}$) de tal forma que

$$\sup_{x\in[a,b]^d} \left| u_i(x) - \cos\left(\frac{\pi}{a} m_i^T x\right)\right| < \frac{\epsilon}{4M|\alpha_i|}$$

y

$$\sup_{x\in[a,b]^d} \left| v_i(x) - \operatorname{sen}\left(\frac{\pi}{a} m_i^T x\right)\right| < \frac{\epsilon}{4M|\beta_i|}.$$

Por lo tanto, aplicando la desigualdad del triángulo obtenemos

$$\sup_{x\in[a,b]^d} \left| \sum_{i=1}^{M} \left(\alpha_i \cos\left(\frac{\pi}{a} m_i^T x\right) + \beta_i \operatorname{sen}\left(\frac{\pi}{a} m_i^T x\right)\right) - \sum_{i=1}^{M} (\alpha_i u_i(x) + \beta_i v_i(x))\right| < \frac{\epsilon}{2}.$$

Como las u_i's y $v_i(x)$'s son redes neuronales, su combinación lineal

$$\psi(x) = \sum_{i=1}^{M} (\alpha_i u_i(x) + \beta_i v_i(x)),$$

es una red neuronal también y de hecho

$$\begin{aligned}
&\sup_{x\in[a,b]^d} |f(x) - \psi(x)| \\
&\leq \sup_{x\in[a,b]^d} \left| f(x) - \sum_{i=1}^{M} \left(\alpha_i \cos\left(\frac{\pi}{a} m_i^T x\right) + \beta_i \operatorname{sen}\left(\frac{\pi}{a} m_i^T x\right)\right)\right| \\
&\quad + \sup_{x\in[a,b]^d} \left| \sum_{i=1}^{M} \left(\alpha_i \cos\left(\frac{\pi}{a} m_i^T x\right) + \beta_i \operatorname{sen}\left(\frac{\pi}{a} m_i^T x\right)\right) - \psi(x)\right| < \frac{2\epsilon}{2} = \epsilon. \qquad \square
\end{aligned}$$

El siguiente corolario del Teorema 8 se obtiene a través del Lema 2:

Corolario 1. *Sea $\mathcal{C}^{(k)}$ la clase que contiene a todos los clasificadores por redes neuronales definidos por redes de una capa oculta con k neuronas ocultas y un sigmoide arbitrario σ. Entonces para cualquier distribución de (X, Y),*

$$\lim_{k\to\infty} \inf_{\phi\in\mathcal{C}^{(k)}} L(\phi) - L^* = 0.$$

La convergencia anterior también se cumple si el rango de los parámetros de la red neuronal, a_{ij}, b_i, c_i, es restringido a un intervalo $[-\beta_k, \beta_k]$, donde $\lim_{k\to\infty} \beta_k = \infty$.

Observación: También es cierto que la clase de redes neuronales de una capa oculta con k neuronas ocultas se vuelve densa en $L_1(\mu)$, para cada medida de probabilidad μ en $\mathbb{R}^d$ cuando $k \to \infty$. Entonces el Teorema 2 puede usarse directamente para demostrar el Corolario 1.

En la práctica, la arquitectura de la red (es decir, k en nuestro caso) es dejada al experimentador, quien ajusta los parámetros a_{ij}, b_i y c_i, dependiendo de los datos D_n. Con respecto a los resultados anteriores, estos son de interés teórico. Es de mayor interés encontrar qué tan lejos está la probabilidad de error de la regla elegida de $\inf_{\phi \in \mathcal{C}^{(k)}} L(\phi)$.

2.4. Gradiente descendente.

En la clasificación basada en verosimilitud, los estimadores son estadísticos suficientes de la función de densidad de probabilidad dada la clase, $p(x|\mathcal{C}^{(i)})$, y de la probabilidad a priori de la clase, $P(\mathcal{C}^{(i)})$, y el método que se utiliza para estimar los parámetros es el de máxima verosimilitud. En discriminación los parámetros son de los discriminantes y están optimizados para minimizar el error de clasificación.

Cuando $w = (w^{(1)}, \ldots, w^{(d)})^T$ denota el vector de pesos o el conjunto de parámetros y $\mathbf{E}(w|X)$ es el error con parámetros w para el conjunto de entrenamiento X dado, buscamos

$$w^* = \arg\min_w \mathbf{E}(w|X).$$

En muchos casos no existe una solución analítica y debemos recurrir a métodos de optimización iterativos, el método más empleado es el **gradiente descendente**. Cuando $\mathbf{E}(w)$ es una función diferenciable de un vector de variables, tenemos el vector gradiente compuesto de las derivadas parciales

$$\nabla_w \mathbf{E} = \left[\frac{\partial \mathbf{E}}{\partial w_1}, \frac{\partial \mathbf{E}}{\partial w_2}, \cdots, \frac{\partial \mathbf{E}}{\partial w_d} \right]^T ;$$

el procedimiento del gradiente descendente para minimizar $\mathbf{E}$ comienza a partir de un w aleatorio y en cada paso actualiza w en la dirección opuesta del gradiente

$$\Delta w_i = -\eta \frac{\partial \mathbf{E}}{\partial w_i} \ , \forall i, \tag{2.4}$$

por lo que

$$w_i = w_i + \Delta w_i, \tag{2.5}$$

donde η es llamado el **tamaño de paso** o **factor de aprendizaje** y determina cuanto moverse en esa dirección. El gradiente ascendente se utiliza para maximizar una función y va en la dirección del gradiente. Cuando llegamos a un mínimo (o

máximo), la derivada es 0 y el procedimiento termina. Esto indica que el procedimiento encuentra el mínimo más cercano que puede ser un mínimo local, y no hay garantía de encontrar el mínimo global, a menos que la función tenga solo un mínimo. El uso de un buen valor para η también es crítico; si es demasiado pequeño la convergencia puede ser demasiado lenta, y un valor grande puede causar oscilaciones e incluso divergencia.

2.5. Discriminación logística.

La cantidad $\log y/(y-1)$ es conocida como la transformación **logit** o **log odds** de y. En el caso de dos clases normales que comparten una matriz de covarianza común, las log odds son lineales:

$$\begin{aligned}
\text{logit}(P(\mathcal{C}^{(1)}|x)) &= \log \frac{P(\mathcal{C}^{(1)}|x)}{1-P(\mathcal{C}^{(1)}|x)} \\
&= \log \frac{P(\mathcal{C}^{(1)}|x)}{P(\mathcal{C}^{(2)}|x)} \\
&= \log \frac{p(x|\mathcal{C}^{(1)})}{p(x|\mathcal{C}^{(2)})} + \log \frac{P(\mathcal{C}^{(1)})}{P(\mathcal{C}^{(2)})} \\
&= \log \frac{(2\pi)^{-d/2}|\Sigma|^{-1/2}\exp[-(1/2)(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)]}{(2\pi)^{-d/2}|\Sigma|^{-1/2}\exp[-(1/2)(x-\mu_2)^T\Sigma^{-1}(x-\mu_2)]} \\
&\quad + \log \frac{P(\mathcal{C}^{(1)})}{P(\mathcal{C}^{(2)})} \\
&= w^T x + w_0,
\end{aligned}$$

donde

$$w = \Sigma^{-1}(\mu_1 - \mu_2)$$

y

$$w_0 = -\frac{1}{2}(\mu_1+\mu_2)^T\Sigma^{-1}(\mu_1-\mu_2) + \log \frac{P(\mathcal{C}^{(1)})}{P(\mathcal{C}^{(2)})}.$$

La inversa del logit

$$\log \frac{P(\mathcal{C}^{(1)}|x)}{1-P(\mathcal{C}^{(1)}|x)} = w^T x + w_0,$$

es la **función logística**, también llamada **función sigmoide logística**, dada por

$$P(\mathcal{C}^{(1)}|x) = \sigma(w^T x + w_0) = \frac{1}{1+\exp[-(w^T x + w_0)]}.$$

2.5.1. Dos clases.

En discriminación logística no modelamos las densidades de clase condicionales $p(x|\mathcal{C}^{(i)})$, sino más bien su proporción. Consideremos dos clases y asumamos que la razón de logverosimilitud es lineal

$$\log\left[\frac{p(x|\mathcal{C}^{(1)})}{p(x|\mathcal{C}^{(2)})}\right] = w^T x + w_0^o.$$

Como se vio anteriormente, esto se cumple cuando las densidades condicionales de clase son normales. Sin embargo, la discriminación logística tiene un alcance más amplio de aplicación, por ejemplo, x puede estar compuesta por atributos discretos o puede ser una mezcla de atributos continuos y discretos.

Utilizando la regla de Bayes, tenemos

$$\begin{aligned}\text{logit}(P(\mathcal{C}^{(1)}|x)) &= \log\frac{P(\mathcal{C}^{(1)}|x)}{1-P(\mathcal{C}^{(1)}|x)}\\ &= \log\frac{p(x|\mathcal{C}^{(1)})}{p(x|\mathcal{C}^{(2)})} + \log\frac{P(\mathcal{C}^{(1)})}{P(\mathcal{C}^{(2)})} = w^T x + w_0,\end{aligned}$$

donde

$$w_0 = w_0^o + \log\frac{P(\mathcal{C}^{(1)})}{P(\mathcal{C}^{(2)})}.$$

Reordenando los términos, obtenemos la función logística

$$y = \widehat{P}(\mathcal{C}^{(1)}|x) = \frac{1}{1+\exp[-(w^T x + w_0)]}, \tag{2.6}$$

como nuestro estimador de $P(\mathcal{C}^{(1)}|x)$.

Veamos cómo obtener w y w_0. Se nos da una muestra de dos clases, $X = \{x^t, r^t\}$, donde $r^t = 1$ si $x \in \mathcal{C}^{(1)}$ y $r^t = 0$ si $x \in \mathcal{C}^{(2)}$. Asumimos que r^t, dado x^t, es Bernoulli con probabilidad $y^t \equiv P(\mathcal{C}^{(1)}|x^t)$, calculada con la ecuación (2.6)

$$r^t|x^t \sim Bernoulli(y^t).$$

Vemos la diferencia de los métodos basados en la verosimilitud donde modelamos $p(x|\mathcal{C}^{(i)})$, en el enfoque basado en discriminantes modelamos directamente $r|x$. La verosimilitud de la muestra es

$$\ell(w, w_0|X) = \prod_t (y^t)^{(r^t)}(1-y^t)^{(1-r^t)}.$$

Sabemos que cuando tenemos una función de verosimilitud para maximizar, podemos siempre convertirla en una función de error para minimizar, $\mathbf{E} = -\log\ell$, en nuestro caso tenemos la **entropía cruzada**

$$\mathbf{E}(w, w_0|X) = -\sum_t r^t \log(y^t) + (1-r^t)\log(1-y^t). \tag{2.7}$$

Debido a la no linealidad de la función logística, no podemos resolver directamente esta minimización, por lo que utilizamos el gradiente descendente para minimizar la entropía cruzada, lo que equivaldría a maximizar la verosimilitud o la logverosimilitud. Si $y = \sigma(a) = 1/[1 + \exp(-a)]$, su derivada está dada por

$$\frac{dy}{da} = y(1 - y)$$

y obtenemos las siguientes ecuaciones de actualización

$$\begin{aligned} \Delta w_j &= -\eta \frac{\partial \mathbf{E}}{\partial w_j} = \eta \sum_t \left(\frac{r^t}{y^t} - \frac{1 - r^t}{1 - y^t} \right) y^t (1 - y^t) x_j^t \\ &= \eta \sum_t (r^t - y^t) x_j^t, \ j = 1, \ldots, d, \end{aligned}$$

y

$$\Delta w_0 = -\eta \frac{\partial \mathbf{E}}{\partial w_0} = \eta \sum_t (r^t - y^t).$$

Una vez que se han obtenido los parámetros w y w_0, durante la clasificación de un dato de prueba x^t, calculamos $y_0^t = \sigma(w^T x^t + w_0)$ y elegimos $C^{(1)}$ si $y_0^t > 0.5$; elegimos $C^{(2)}$ de otro modo.

2.5.2. Múltiples clases.

Ahora se generaliza para $k > 2$ clases. Tomamos una de las clases, por ejemplo $\mathcal{C}^{(k)}$, como clase de referencia y asumimos que

$$\log \frac{p(x|\mathcal{C}^{(i)})}{p(x|\mathcal{C}^{(k)})} = w_i^T x + w_{i0}^o,$$

entonces tenemos

$$\frac{P(\mathcal{C}^{(i)}|x)}{P(\mathcal{C}^{(k)}|x)} = \exp[w_i^T x + w_{i0}],$$

con $w_{i0} = w_{i0}^o + \log P(\mathcal{C}^{(i)})/P(\mathcal{C}^{(k)})$.

Vemos que

$$\sum_{i=1}^{k-1} \frac{P(\mathcal{C}^{(i)}|x)}{P(\mathcal{C}^{(k)}|x)} = \frac{1 - P(\mathcal{C}^{(k)}|x)}{P(\mathcal{C}^{(k)}|x)} = \sum_{i=1}^{k-1} \exp[w_i^T x + w_{i0}],$$

de donde

$$P(\mathcal{C}^{(k)}|x) = \frac{1}{1 + \sum_{i=1}^{k-1} \exp[w_i^T x + w_{i0}]},$$

y también que

$$\frac{P(\mathcal{C}^{(i)}|x)}{P(\mathcal{C}^{(k)}|x)} = \exp[w_i^T x + w_{i0}],$$

luego

$$P(\mathcal{C}^{(i)}|x) = \frac{\exp[w_i^T x + w_{i0}]}{1 + \sum_{j=1}^{k-1} \exp[w_j^T x + w_{j0}]},\ i = 1, \ldots, k-1.$$

Para tratar todas las clases de manera uniforme, escribimos

$$y_i = \widehat{P}(\mathcal{C}^{(i)}|x) = \frac{\exp[w_i^T x + w_{i0}]}{\sum_{j=1}^{k} \exp[w_j^T x + w_{j0}]},\ i = 1, \ldots, k, \tag{2.8}$$

la cual es llamada la **función softmax**. Si la suma ponderada de una clase es suficientemente más grande que para las otras, después de la exponenciación y normalización, su correspondiente y_i estará cerca de 1 y las demás estarán cerca de 0. Esto es como tomar un máximo, excepto que es diferenciable, de ahí el nombre de softmax. La función softmax también garantiza que $\sum_i y_i = 1$.

Veamos cómo obtener los parámetros w y w_0. En este caso se tienen $k > 2$ clases, cada punto de la muestra es ensayo multinomial con una realización, es decir, $r^t|x^t \sim Mult_k(1, y^t)$, donde $y_i^t \equiv P(\mathcal{C}^{(i)}|x^t)$. La verosimilitud es

$$\ell(\{w_i, w_{i0}\}_i|X) = \prod_t \prod_i (y_i^t)^{r_i^t}$$

y la función de error es de nuevo la entropía cruzada

$$\mathbf{E}(\{w_i, w_{i0}\}_i|X) = -\sum_t \sum_i r_i^t \log y_i^t.$$

Utilizamos nuevamente el gradiente descendente. Si $y_i = \exp(a_i)/\sum_j (a_j)$, tenemos

$$\frac{\partial y_i}{\partial a_j} = y_i(\delta_{ij} - y_j),$$

donde δ_{ij} es la delta de Kronecker, que es 1 si $i = j$ y 0 si $i \neq j$. Ya que $\sum_i r_i^t = 1$, tenemos las siguientes ecuaciones de actualización, para $j = 1, \ldots, k$,

$$\begin{aligned}
\Delta w_j &= \eta \sum_t \sum_i \frac{r_i^t}{y_i^t} y_i^t (\delta_{ij} - y_j^t) x^t \\
&= \eta \sum_t \sum_i r_i^t (\delta_{ij} - y_j^t) x^t \\
&= \eta \sum_t \left[\sum_i r_i^t \delta_{ij} - y_j^t \sum_i r_i^t \right] x^t \\
&= \eta \sum_t (r_j^t - y_j^t) x^t
\end{aligned}$$

y

$$\Delta w_{j0} = \eta \sum_t (r_j^t - y_j^t).$$

Note que debido a la normalización en la función softmax, w_j y w_{j0} se ven afectados no solo por $x^t \in \mathcal{C}^{(j)}$ sino también por $x^t \in \mathcal{C}^{(i)}$, $i \neq j$. Los discriminantes se actualizan para que la clase correcta tenga la suma ponderada más alta después de la función softmax, y las otras clases tengan sus sumas ponderadas lo más bajas posible.

2.6. Discriminación por regresión.

En regresión, el modelo probabilístico es

$$r^t = y^t + \epsilon,$$

donde, $\epsilon \sim N(0, \sigma^2)$. Si $r^t \in \{0, 1\}$, y^t puede restringirse a estar en este rango utilizando la función logística

$$y^t = \sigma(w^T x^t + w_0) = \frac{1}{1 + \exp[-(w^T x^t + w_0)]}.$$

Entonces la verosimilitud muestral en regresión, asumiendo $r|x \sim N(y, \sigma^2)$, es

$$\ell(w, w_0|X) = \prod_t \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(r^t - y^t)^2}{2\sigma^2}\right].$$

Maximizar la logverosimilitud es minimizar la suma de errores cuadráticos

$$\mathbf{E}(w, w_0|X) = \frac{1}{2} \sum_t (r^t - y^t)^2.$$

Utilizando el gradiente descendente, obtenemos

$$\Delta w = \eta \sum_t (r^t - y^t) y^t (1 - y^t) x^t,$$

y

$$\Delta w_0 = \eta \sum_t (r^t - y^t) y^t (1 - y^t).$$

Este método también puede ser usado cuando hay $k > 2$ clases. El modelo probabilístico es

$$r^t = y^t + \epsilon,$$

donde $\epsilon \sim N_k(0, \sigma^2 I_k)$. Asumiendo un modelo lineal para cada clase, tenemos

$$y_i^t = \sigma(w_i^T x^t + w_{i0}) = \frac{1}{1 + \exp[-(w_i^T x^t + w_{i0})]}.$$

Entonces la verosimilitud muestral es

$$\ell(\{w_i, w_{i0}\}_i|X) = \prod_t \frac{1}{(2\pi)^{k/2}|\Sigma|^{1/2}} \exp\left[-\frac{||r^t - y^t||^2}{2\sigma^2}\right]$$

y la función de error es

$$\mathbf{E}(\{w_i, w_{i0}\}_i|X) = \frac{1}{2}\sum_t ||r^t - y^t||^2 = \frac{1}{2}\sum_t \sum_i (r_i^t - y_i^t)^2.$$

Las ecuaciones de actualización para $i = 1, \ldots, k$, son

$$\Delta w_i = \eta \sum_t (r_i^t - y_i^t) y_i^t (1 - y_i^t) x^t$$

y

$$\Delta w_{i0} = \eta \sum_t (r_i^t - y_i^t) y_i^t (1 - y_i^t).$$

Note que al hacerlo de este modo no hacemos uso de la información de que solo una de las y_i debe ser 1 y las demás deben ser 0, o que $\sum_i y_i = 1$. La función softmax de la ecuación (2.8) nos permite incorporar esta información extra que tenemos, debido al rango de valores de las estimaciones de las probabilidades a posteriori de las clases. Usando salidas logísticas en el caso $k > 2$, tratamos a las y_i's como si fueran funciones independientes.

Notar que para una clase determinada, si utilizamos el enfoque de regresión, habrá actualizaciones hasta que la salida correcta sea 1 y todas las demás sean 0. De hecho esto no es necesario, porque durante las pruebas simplemente vamos a elegir el máximo de todas las salidas, es suficiente entrenar solo hasta que la salida correcta sea mayor que las otras, que es exactamente lo que hace la función softmax.

Entonces este enfoque con múltiples salidas logísticas es más apropiado cuando las clases no son mutuamente excluyentes y exhaustivas. Esto es, para un x^t, todas las r_i^t pueden ser 0, es decir, x^t no pertenece a ninguna de las clases, o más de un r_i^t puede ser 1, cuando las clases se superponen.

2.7. Entrenamiento de un perceptrón.

El perceptrón define un hiperplano, y la red neuronal es solo una forma de implementar el hiperplano. Dada una muestra de datos, los valores de los pesos se pueden calcular **fuera de línea**, y luego cuando estos valores son insertados, el perceptrón se puede usar para calcular los valores de salida.

Al entrenar redes neuronales, generalmente usamos el aprendizaje en **línea** donde no se da la muestra completa, sino que se dan puntos (o datos) uno por uno y la red actualiza sus parámetros después de cada punto, adaptándose lentamente en el tiempo. Este enfoque es interesante por las siguientes razones:

1. Nos ahorra el costo de almacenar la muestra de entrenamiento en una memoria externa y almacenar los resultados intermedios durante la optimización. Un enfoque como el support vector machine (ver [8]) puede ser bastante costoso con muestras grandes; en algunas aplicaciones se prefiere un enfoque más simple, donde no se necesite almacenar toda la muestra, y resolver un problema de optimización complejo con ella.

2. El problema puede estar cambiando con el tiempo, lo que significa que la distribución de la muestra no es fija y un conjunto de entrenamiento no puede elegirse a priori. Por ejemplo, podemos estar implementado un sistema de reconocimiento de voz que se adapte por si mismo a su usuario.

3. Pueden haber cambios físicos en el sistema. Por ejemplo, en un sistema robótico, los componentes del sistema pueden desgastarse o los sensores pueden degradarse.

En el aprendizaje en línea no escribimos la función de error sobre toda la muestra, sino sobre puntos individuales. Partiendo de pesos iniciales aleatorios, en cada iteración ajustamos un poco los parámetros para minimizar el error, sin olvidar lo aprendido previamente. Si esta función de error es diferenciable podemos usar el gradiente descendente.

Por ejemplo, en regresión el error sobre el par individual con índice t, (x^t, r^t), donde x^t denota al vector de entradas y r^t es la etiqueta asociada a x^t, es

$$\mathbf{E}^t(w|x^t, r^t) = \frac{1}{2}(r^t - y^t)^2 = \frac{1}{2}[r^t - (w^T x^t)]^2, \tag{2.9}$$

y para $j = 0, \ldots, d$, la actualización en línea es

$$\Delta w_j^t = \eta(r^t - y^t)x_j^t,$$

donde η es el factor de aprendizaje, que disminuye gradualmente en el tiempo para la convergencia estocástica. Esto es conocido como el **gradiente descendente estocástico**.

2.8. El algoritmo de Backpropagation.

El entrenamiento de un perceptrón multicapa es análogo al de un perceptrón, la única diferencia es que ahora la salida es una función no lineal de las entradas debido a la función sigmoide en las neuronas ocultas.

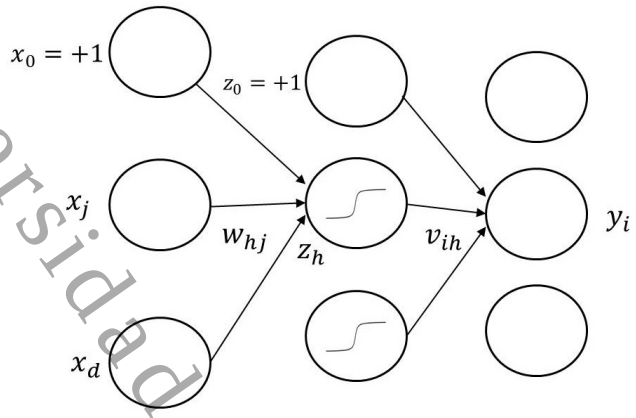


Figura 2.11: Estructura de un perceptrón multicapa en la cual las x_j, $j = 0, \ldots, d$, son las entradas y las z_h, $h = 1, \ldots, H$, son las neuronas ocultas, donde H es la dimensión de este espacio oculto, z_0 es el intercepto de la capa oculta, las y_i, $i = 1, \ldots, k$, son las unidades de salida, las w_{hj} son los pesos de la primera capa, y las v_{ih} son los pesos de la segunda capa.

Como se muestra en la Figura 2.11, se consideran las neuronas ocultas como entradas de la segunda capa la cual es un perceptrón, que se ajusta de la forma usual dadas las entradas z_h. Para los pesos de la primera capa, w_{hj}, utilizamos la regla de la cadena para calcular la derivada parcial

$$\frac{\partial \mathbf{E}}{\partial w_{hj}} = \frac{\partial \mathbf{E}}{\partial y_i} \frac{\partial y_i}{\partial z_h} \frac{\partial z_h}{\partial w_{hj}}.$$

Es como si el error se propagara desde la salida y de regreso a las entradas, y por esa razón se le denominó **Backpropagation**, ver [1]. Con la fórmula anterior se calculan las entradas del vector gradiente y se lleva a cabo el método del gradiente descendente. Existen diversos casos en los cuales se puede aplicar el algoritmo de Backpropagation, a continuación se mostrarán algunos de estos casos.

2.8.1. Regresión no lineal.

Tomemos primero el caso de la regresión no lineal (con una salida) calculada como

$$y^t = \sum_{h=1}^{H} v_h z_h^t + v_0,$$

con z_h calculado por el sigmoide estándar o logístico. La función de error sobre toda la muestra en regresión es

$$\mathbf{E}(W, v|X) = \frac{1}{2} \sum_t (r^t - y^t)^2,$$

donde W es la matriz de pesos w_{hj}, cuyas filas son los vectores de pesos. La segunda capa es un perceptrón con las neuronas ocultas como las entradas, y utilizamos la regla de mínimos cuadrados para actualizar los pesos de la segunda capa

$$\Delta v_h = \eta \sum_t (r^t - y^t) z_h^t.$$

La primera capa también consiste de perceptrones con las neuronas ocultas como unidades de salida, pero al actualizar los pesos de la primera capa, no podemos usar la regla de mínimos cuadrados directamente porque no tenemos una salida deseada que sea especificada por las neuronas ocultas. Aquí es donde entra en juego la regla de la cadena. Tenemos que

$$\begin{aligned} \Delta w_{hj} &= -\eta \frac{\partial \mathbf{E}}{\partial w_{hj}} \\ &= -\eta \sum_t \frac{\partial \mathbf{E}^t}{\partial y^t} \frac{\partial y^t}{\partial z_h^t} \frac{\partial z_h^t}{\partial w_{hj}} \\ &= \eta \sum_t (r^t - y^t) v_h z_h^t (1 - z_h^t) x_j^t. \end{aligned}$$

El producto de los dos primeros términos $(r^t - y^t)v_h$ actua como el término de error para la neurona oculta h. Este error es llamado **backpropagated**, porque se propaga hacia atrás desde el error hasta la neurona oculta. La cantidad $(r^t - y^t)$ es el error de salida, ponderado por la "responsabilidad" de la neurona oculta dada por su peso v_h. En el tercer factor, $z_h(1 - z_h)$ es la derivada del sigmoide logístico y x_j^t es la derivada de la suma ponderada con respecto al peso w_{hj}. Notar que el cambio en el peso de la primera capa, Δw_{hj}, hace uso del peso de la segunda capa, v_h. Por lo tanto, debemos calcular los cambios en ambas capas y actualizar los pesos de la primera capa, haciendo uso de los valores previos de los pesos de la segunda capa, luego actualizar los pesos de la segunda capa.

Los pesos w_{hj}, v_h parten inicialmente de pequeños valores aleatorios, por ejemplo, en el rango $[-0.01, 0.01]$, para no saturar a los sigmoides. También es una buena idea normalizar las entradas para que todos tengan media 0 y varianza unitaria y tengan la misma escala, porque usamos un único parámetro η.

Con las ecuaciones de aprendizaje que se dan aquí, para cada patrón, calculamos la dirección en la que es necesario cambiar cada parámetro y la magnitud de este cambio. En el **aprendizaje por lotes**, acumulamos estos cambios sobre todos los patrones y realizamos el cambio después de realizar una pasada completa sobre el conjunto de entrenamiento, como se mostró en las ecuaciones de actualización anteriores.

De igual manera es posible tener aprendizaje en línea, actualizando los pesos después de cada patrón, implementando así el gradiente descendente estocástico. Una pasada completa sobre todos los patrones en el conjunto de entrenamiento es llamada una **época**. El factor de aprendizaje, η, debe elegirse más pequeño en

este caso y los patrones deben escanearse en un orden aleatorio. El aprendizaje en línea converge más rápido porque pueden haber patrones similares en el conjunto de datos, y la estocasticidad tiene el efecto de agregar ruido y puede ayudar a escapar de los mínimos locales. También es posible tener múltiples unidades de salidas, en cuyo caso varios problemas de regresión están aprendiendo al mismo tiempo. Tenemos

$$y_i^t = \sum_{h=1}^{H} v_{ih} z_h^t + v_{i0},$$

y el error es

$$\mathbf{E}(W, V|X) = \frac{1}{2} \sum_t \sum_i (r_i^t - y_i^t)^2,$$

donde V es la matriz de pesos, v_{ih}. Las reglas de actualización por lotes son entonces

$$\Delta v_{ih} = \eta \sum_t (r_i^t - y_i^t) z_h^t,$$

por lo que

$$\Delta w_{hj} = \eta \sum_t \left[\sum_i (r_i^t - y_i^t) v_{ih} \right] z_h^t (1 - z_h^t) x_j^t.$$

La suma $\sum_i (r_i^t - y_i^t) v_{ih}$ es el error propagado hacia atrás (backpropagated) acumulado de la neurona oculta h de todas las unidades de salida. Notar que en este caso, las unidades de salida comparten las mismas neuronas ocultas y en consecuencia usan la misma representación oculta, por lo tanto, asumimos que tenemos problemas de predicción relacionados, correspondientes a estas salidas diferentes. Una alternativa es entrenar perceptrones multicapa separados para los problemas de regresión separados, cada uno con sus propias neuronas ocultas separadas.

2.8.2. Discriminación de dos clases.

Cuando hay dos clases, una unidad de salida es suficiente y puede ser vista de la siguiente forma

$$y^t = \sigma \left(\sum_{h=1}^{H} v_h z_h^t + v_0 \right),$$

que aproxima a $P(\mathcal{C}^{(1)}|x^t)$, y $\widehat{P}(\mathcal{C}^{(2)}|x^t) \equiv 1 - y^t$. La función de error, mejor conocida como **entropía cruzada** es

$$\mathbf{E}(W, v|X) = -\sum_t r^t \log y^t + (1 - r^t) \log(1 - y^t).$$

Las ecuaciones de actualización implementando el gradiente descendente son

$$\Delta v_h = \eta \sum_t (r^t - y^t) z_h^t$$

y

$$\Delta w_{hj} = \eta \sum_t (r^t - y^t) v_h z_h^t (1 - z_h^t) x_j^t.$$

Como en el perceptrón simple, las ecuaciones de actualización para la regresión y clasificación son idénticas (esto no significa que los valores lo sean).

2.8.3. Discriminación multiclase.

En un problema de clasificación multiclase ($k > 2$), hay k salidas

$$o_i^t = \sum_{h=1}^{H} v_{ih} z_h^t + v_{i0},$$

se utiliza la función softmax para indicar la dependencia entre clases, es decir, son mutuamente excluyentes y exhaustivas

$$y_i^t = \frac{\exp o_i^t}{\sum_k \exp o_k^t},$$

donde y_i aproxima a $P(\mathcal{C}^{(i)}|x^t)$. La función de error es

$$\mathbf{E}(W, V|X) = -\sum_t \sum_i r_i^t \log y_i^t,$$

de donde obtenemos las ecuaciones de actualización utilizando el gradiente descendente:

$$\Delta v_{ih} = \eta \sum_t (r_i^t - y_i^t) z_h^t$$

y

$$\Delta w_{hj} = \eta \sum_t \left[\sum_i (r_i^t - y_i^t) v_{ih} \right] z_h^t (1 - z_h^t) x_j^t.$$

Richard y Lippmann, 1991 (ver [1]), han demostrado que dada una red de suficiente complejidad y suficientes datos de entrenamiento, un perceptrón multicapa adecuadamente entrenado estima de buena forma las probabilidades a posteriori.

2.8.4. Múltiples capas ocultas.

Como se ha mostrado anteriormente, es posible tener múltiples capas ocultas, cada una con sus propios pesos y en la que se aplica la función sigmoide a su suma ponderada. Para regresión, digamos, si tenemos un perceptrón multicapa con dos capas ocultas, escribimos

$$z_{1h} = \sigma(w_{1h}^T x) = \sigma\left(\sum_{j=1}^{d} w_{1hj} x_j + w_{1h0} \right), \; h = 1, \ldots, H_1,$$

$$z_{2\ell} = \sigma(w_{2\ell}^T z_1) = \sigma\left(\sum_{h=0}^{H_1} w_{2\ell h} z_{1h} + w_{2\ell 0}\right),\ \ell = 1, \ldots, H_2,$$

$$y = v^T z_2 = \sum_{\ell=1}^{H_2} v_\ell z_{2\ell} + v_0,$$

donde w_{1h} y $w_{2\ell}$ son los pesos de la primera y segunda capa, z_{1h} y z_{2h} son las neuronas de la primera y segunda capa oculta, y v es el vector cuyas entradas son los pesos de la tercera capa. El entrenamiento de una red de este tipo es similar, excepto que para entrenar los pesos de la primera capa necesitamos propagar hacia atrás (backpropagate) una capa más.

A continuación se presenta un ejemplo de redes neuronales utilizando rutinas computaciones.

Ejemplo 2. *Se ajustará una red neuronal de una capa oculta (con diez neuronas ocultas) a través de Backpropagation. Se obtienen* 1000 *observaciones del siguiente modelo*

$$Y = \sigma(a_1^T X) + (a_2^T X)^2 + 0.30 * Z,$$

donde σ *es una función sigmoide logística,* Z *es normal estándar,* $X^T = (X_1, X_2)$*, siendo* X_j *,* $j = 1, 2$*, normales estándar independientes,* $a_1 = (3, 3)$*,* $a_2 = (3, -3)$*. Los datos se dividen aleatoriamente en dos subconjuntos, los datos de entrenamiento* (700 *datos*) *y los datos de prueba* (300 *datos*)*. Se analiza el comportamiento de los datos con un modelo lineal y una red neuronal. Se grafica el error de entrenamiento variando el número de neuronas ocultas en la red, de* 1 *a* 10*, y se determina el número mínimo de neuronas ocultas para desempeñar bien esta tarea. El código del programa escrito en el software Rstudio que ajusta una red neuronal de una capa oculta con diez neuronas ocultas, vía Backpropagation, se encuentra en el Apéndice de este trabajo.*

Como se define en [5]*, si* $\widehat{Y}$ *es un vector de* n *predicciones y* Y *es el vector de los valores verdaderos, entonces el error cuadrático medio de un modelo lineal (MSE.ML) es*

$$MSE.ML = \frac{1}{n}\sum_{i=1}^{n}(\widehat{Y}_i - Y_i)^2.$$

Análogamente, el error cuadrático medio de la red neuronal (MSE.RN) es

$$MSE.RN(k) = \frac{1}{n}\sum_{i=1}^{n}(\widehat{Y}.predictorRN(k)_i - Y_i)^2,$$

donde $\widehat{Y}.predictorRN(k)$ *es el vector predictor y* k *es el número de neuronas ocultas. Para los datos de prueba, se tiene que el MSE.ML es igual a* 878.185*. En la Tabla* 1 *se presentan los valores del MSE.RN de una capa oculta y con el número de neuronas ocultas variando de* 1 *a* 10*.*

Neuronas ocultas	***MSE.RN***
1	1287.8617
2	559.9929
3	58.3557
4	51.1239
5	58.5398
6	54.4330
7	48.3486
8	51.7532
9	40.7448
10	75.7933

Tabla 1. *MSE.RN de una red neuronal con una capa oculta y variando el número de neuronas ocultas de* 1 *a* 10.

Debido a lo obtenido en la tabla anterior y al resultado de MSE.ML, podemos observar que a partir de aplicar dos neuronas ocultas el MSE.RN es menor que el MSE.ML.

Por otro lado en la Tabla 2 *se muestran los errores de entrenamiento de la red neuronal variando de* 1 *a* 10 *el número de neuronas ocultas en la red. Estos errores de entrenamiento son calculados mediante la siguiente fórmula usando los datos de entrenamiento*

$$\frac{1}{2}\sum_{i=1}^{m}\left(\widehat{Y}.predictorRN(k)_i - Y_i\right)^2.$$

En el ajuste de la red neuronal de este ejemplo se utilizan los datos de entrenamiento centrados y escalados.

Neuronas ocultas	***Error de entrenamiento***
1	6.6033
2	3.8474
3	0.1963
4	0.1978
5	0.1920
6	0.2160
7	0.2160
8	0.1748
9	0.1800
10	0.2934

Tabla 2. *Errores de entrenamiento en la red neuronal.*

Observamos que a partir de tres neuronas ocultas en la red neuronal de una capa oculta, se tiene una tendencia a un error de entrenamiento bajo en la red neuronal. Del mismo modo, en la Figura 2.12 *se puede ver el comportamiento de dichos errores. Por lo tanto, para este ejemplo podríamos quedarnos con una red neuronal de* 3 *neuronas ocultas.*

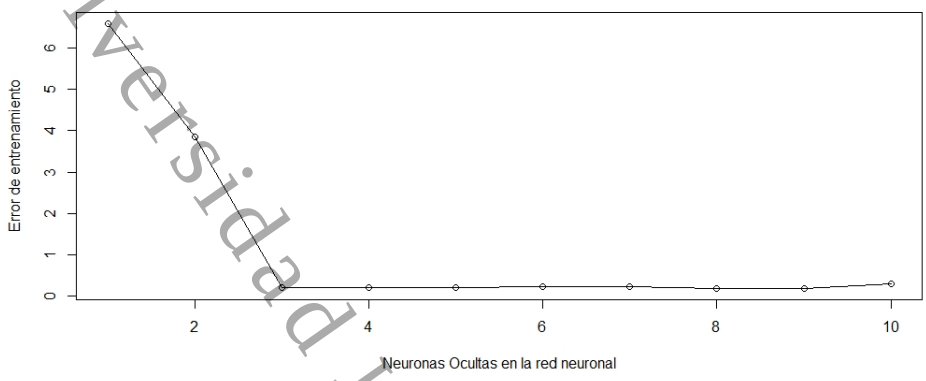


Figura 2.12: Gráfica de errores de entrenamiento en la red neuronal.

Finalmente, en la Figura 2.13 *se muestra una red neuronal de una capa oculta con tres neuronas ocultas, la cual tiene un error de* 0.1963 *y se efectuó en* 482 *pasos.*

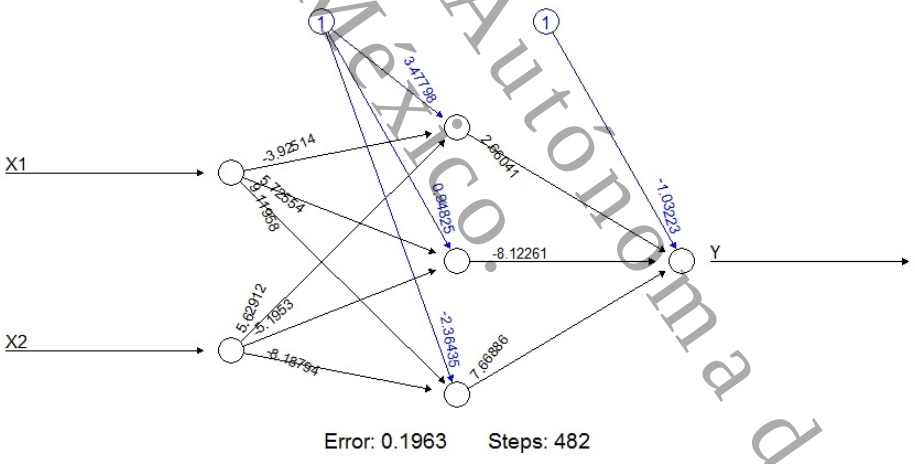


Figura 2.13: Red neuronal de una capa oculta con tres neuronas ocultas.

Capítulo 3

Clasificación de patrones con redes neuronales.

En este capítulo se mostrarán algunas aplicaciones de redes neuronales con datos encontrados en la literatura. Estos datos son divididos en datos de entrenamiento y de prueba. Con los datos de entrenamiento se ajustan redes neuronales para clasificación utilizando la función sigmoide logística y la función sigmoide arcotangente. Posteriormente, se utilizan los datos de prueba para evaluar el modelo de la red neuronal para la clasificación de patrones a través de tablas de confusión, y se comparan las proporciones de error de clasificación correspondientes a las dos funciones sigmoides en estudio.

3.1. Dígitos escritos a mano.

Se cuenta con datos de dígitos escritos manualmente tomados del conjunto de datos del Modified National Institute of Standards and Technology (MNIST), ver [2], que a su vez se construyó modificando un subconjunto de un conjunto de datos mucho más grande producido por NIST (the National Institute of Standards and Technology). La base de datos consiste de un conjunto de entrenamiento de $60,000$ muestras y un conjunto de prueba de $10,000$ muestras. Algunos datos fueron recopilados de empleados de la oficina del censo y el resto fueron recopilados de niños de secundaria, se tuvo cuidado de garantizar que las muestras del conjunto de prueba fueran escritos por personas diferentes a las de las muestras del conjunto de entrenamiento.

Los datos originales del NIST tenían píxeles binarios (blanco o negro). Para crear los datos del MNIST, se normalizó el tamaño de estas imágenes para que quepan en una imagen de 20×20 píxeles, mientras se conserva su relación de aspecto. En consecuencia, el suavizado, (anti-aliasing) utilizado para cambiar la resolución de las imágenes de los dígitos del MNIST resultaron tener escalas de grises. Luego estas imágenes se centraron en una imagen de 28×28 píxeles. Algunos

ejemplos se muestran en la Figura 3.1.

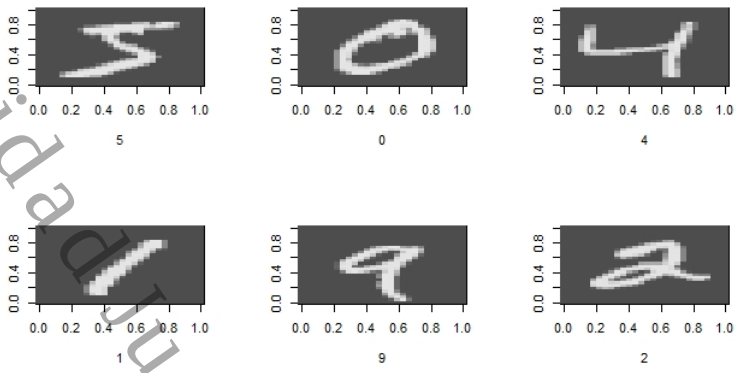
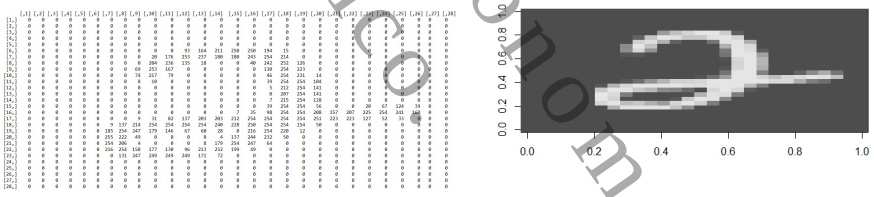


Figura 3.1: Imágenes de escritura de dígitos del MNIST de 28×28 píxeles en escalas de grises.

Cada imagen es transformada en un vector de entradas reales de dimensión $784 = 28 \times 28$, mediante la mediante la concatenación de columnas. En la Figura 3.2 se puede apreciar un ejemplo de ello.



(a) Vista concatenada. (b) Vista en escalas de grises.

Figura 3.2: Visualización del dígito 2.

La motivación de este ejemplo es mostrar que la clasificación de dígitos puede ser automatizada mediante redes neuronales artificiales de manera adecuada. El proceso de clasificar grandes cantidades de dígitos se puede efectuar en lapsos de tiempo relativamente cortos, además el procesamiento es realizado por un equipo de cómputo y el trabajo de catalogar los dígitos se realiza de manera objetiva. En cambio, un grupo de personas clasificaría una cantidad grande de dígitos en un tiempo extenso, además de que podrían haber errores debido al agotamiento o por carencia de objetividad en dicho proceso.

Como se menciona en [5], la clasificación de dígitos escritos manualmente se puede utilizar en la implementación de un proceso de ordenamiento automático de

sobres de acuerdo a su código postal. Otras aplicaciones son el reconocimiento de números de placa y en el procesamiento de cheques bancarios, ver [10].

Utilizando los datos de entrenamiento se ajustaron redes neuronales usando la función sigmoide logística y la función sigmoide arcotangente. En cada caso se consideró una capa oculta y el número de neuronas ocultas se varió en 3, 6, 10, 13, 15 y 20. A continuación se presentan las correspondientes tablas de confusión. En estas tablas "Reales" se refiere a los verdaderos valores de los dígitos; "Predichos" al valor predicho por la red neuronal; los valores dentro de la tabla proporcionan las frecuencias en la clasificación.

Reales \ **Predichos**	1	2	3	4	6	7
0	1	59	910	2	1	7
1	1077	4	39	1	13	1
2	16	699	179	28	105	5
3	7	54	928	3	4	14
4	8	13	2	822	25	112
5	6	76	787	1	5	17
6	8	95	21	36	798	0
7	19	3	55	19	5	927
8	18	64	832	11	11	38
9	5	4	43	57	2	898

Tabla 3. *Tabla de confusión de la red neuronal con función de activación logística de* 1 *capa oculta con* 3 *neuronas ocultas.*

Reales \ **Predichos**	0	1	4	7
0	956	21	2	1
1	3	1130	0	2
2	764	228	29	11
3	59	928	1	22
4	36	20	678	248
5	106	745	12	29
6	926	17	14	1
7	12	66	40	910
8	70	873	2	29
9	19	35	414	541

Tabla 4. *Tabla de confusión de la red neuronal con función de activación arcotangente de* 1 *capa oculta con* 3 *neuronas ocultas.*

Reales \ Predichos	0	1	2	3	4	5	6	7	8	9
0	900	0	3	20	35	8	7	5	1	1
1	0	1101	4	9	1	0	3	2	8	7
2	25	58	747	92	19	3	25	27	29	7
3	13	21	22	814	3	56	7	29	25	20
4	0	5	4	1	803	7	35	2	6	119
5	67	3	14	116	77	540	26	19	10	20
6	28	8	14	0	47	5	855	0	0	1
7	11	16	24	13	4	0	2	866	4	88
8	8	56	18	89	225	41	15	5	448	69
9	6	2	6	22	55	24	2	29	2	861

Tabla 5. *Tabla de confusión de la red neuronal con función de activación logística de* 1 *capa oculta con* 6 *neuronas ocultas.*

Reales \ Predichos	0	1	2	3	4	5	6	7	8	9
0	860	0	42	4	1	45	14	9	3	2
1	0	1090	2	16	0	0	3	3	21	0
2	21	6	859	47	6	5	23	20	32	13
3	1	2	80	767	0	115	4	21	17	3
4	3	6	1	1	837	0	26	10	50	48
5	21	5	14	43	14	656	31	31	68	9
6	25	2	11	1	13	5	874	0	27	0
7	2	20	25	26	11	2	1	912	4	25
8	13	26	12	45	18	102	23	18	673	44
9	15	8	3	5	49	16	0	120	25	768

Tabla 6. *Tabla de confusión de la red neuronal con función de activación arcotangente de* 1 *capa oculta con* 6 *neuronas ocultas.*

Reales \ Predichos	0	1	2	3	4	5	6	7	8	9
0	947	0	5	3	1	3	6	5	9	1
1	0	1096	4	6	0	3	3	3	17	3
2	19	16	888	30	14	2	24	11	20	8
3	12	10	31	832	5	42	2	19	47	10
4	2	1	6	1	905	2	16	4	3	42
5	20	8	5	40	17	708	15	11	57	11
6	15	1	12	1	9	12	896	3	9	0
7	5	16	34	4	14	1	0	923	12	19
8	8	13	15	23	19	25	33	7	809	22
9	6	5	4	7	56	3	7	38	28	855

Tabla 7. *Tabla de confusión de la red neuronal con función de activación logística de* 1 *capa oculta con* 10 *neuronas ocultas.*

Reales \ Predichos	0	1	2	3	4	5	6	7	8	9
0	928	0	4	3	1	17	18	2	2	5
1	0	1074	8	16	0	3	2	7	23	2
2	25	13	889	13	11	10	18	22	29	2
3	2	4	53	839	1	57	1	18	24	11
4	3	4	3	3	865	6	9	7	15	67
5	9	1	11	56	13	692	18	14	56	22
6	14	0	18	2	17	31	867	0	6	3
7	3	23	15	14	10	8	2	897	10	46
8	9	15	15	29	11	36	29	15	752	63
9	7	3	0	10	37	20	3	24	17	888

Tabla 8. *Tabla de confusión de la red neuronal con función de activación arcotangente de* 1 *capa oculta con* 10 *neuronas ocultas.*

Reales \ Predichos	0	1	2	3	4	5	6	7	8	9
0	918	0	9	9	1	15	22	3	3	0
1	0	1086	3	6	1	3	2	3	31	0
2	21	6	901	34	12	6	17	13	17	5
3	0	5	49	860	1	45	1	18	24	7
4	1	3	8	1	914	2	12	6	3	32
5	21	4	13	55	8	724	18	10	30	9
6	12	3	9	0	15	11	900	1	5	2
7	4	6	27	11	7	4	1	925	7	36
8	7	6	22	23	17	66	18	13	789	13
9	9	5	5	3	73	12	2	37	19	844

Tabla 9. *Tabla de confusión de la red neuronal con función de activación logística de* 1 *capa oculta con* 13 *neuronas ocultas.*

Reales \ Predichos	0	1	2	3	4	5	6	7	8	9
0	921	0	7	4	7	16	19	2	3	1
1	0	1096	6	5	0	2	4	3	19	0
2	10	7	915	30	10	9	14	11	22	4
3	5	4	27	851	0	75	4	8	27	9
4	0	3	12	4	896	0	17	1	6	43
5	27	2	8	52	10	736	11	3	40	3
6	16	4	8	0	17	18	885	1	9	0
7	2	10	28	15	5	3	0	925	4	36
8	15	10	15	14	13	29	14	8	831	25
9	8	3	0	4	35	19	2	27	13	898

Tabla 10. *Tabla de confusión de la red neuronal con función de activación arcotangente de* 1 *capa oculta con* 13 *neuronas ocultas.*

Reales \ Predichos	0	1	2	3	4	5	6	7	8	9
0	948	0	1	1	0	9	12	6	3	0
1	0	1105	7	3	0	3	1	4	12	0
2	7	2	923	21	12	12	11	16	25	3
3	8	2	24	892	1	28	3	13	30	9
4	3	3	2	1	914	2	21	2	6	28
5	14	6	9	31	4	772	15	6	23	12
6	14	3	8	0	14	25	889	2	3	0
7	6	10	25	8	7	0	0	924	9	39
8	8	7	15	38	15	16	8	7	849	11
9	8	6	2	11	50	8	0	23	16	885

Tabla 11. *Tabla de confusión de la red neuronal con función de activación logística de* 1 *capa oculta con* 15 *neuronas ocultas.*

Reales \ Predichos	0	1	2	3	4	5	6	7	8	9
0	938	0	10	2	0	11	16	1	1	1
1	0	1103	5	9	0	4	2	3	9	0
2	10	1	910	25	16	5	11	13	35	6
3	4	2	35	870	2	58	2	18	11	8
4	3	1	3	0	911	2	19	7	6	30
5	13	3	5	55	9	744	12	6	33	12
6	30	3	10	0	12	18	875	0	9	1
7	1	12	45	9	8	0	0	930	3	20
8	10	8	12	31	9	46	16	6	828	8
9	9	5	2	14	47	14	2	32	21	863

Tabla 12. *Tabla de confusión de la red neuronal con función de activación arcotangente de* 1 *capa oculta con* 15 *neuronas ocultas.*

Reales \ Predichos	0	1	2	3	4	5	6	7	8	9
0	941	0	6	3	7	7	8	1	4	3
1	0	1111	3	3	0	1	5	2	10	0
2	18	6	920	10	15	5	17	13	25	3
3	4	0	21	912	0	25	7	15	21	5
4	4	4	4	4	875	3	16	6	8	58
5	12	3	5	28	10	784	13	7	23	7
6	9	6	15	3	14	14	891	0	5	1
7	4	5	20	9	4	4	0	949	8	25
8	6	1	16	30	14	16	11	9	859	12
9	8	4	2	15	58	5	2	14	11	890

Tabla 13. *Tabla de confusión de la red neuronal con función de activación logística de* 1 *capa oculta con* 20 *neuronas ocultas.*

Reales \ Predichos	0	1	2	3	4	5	6	7	8	9
0	939	1	1	2	2	12	11	8	4	0
1	0	1107	5	3	0	3	5	3	8	1
2	8	9	910	31	17	5	11	14	25	2
3	3	3	25	914	1	14	5	14	24	7
4	1	0	8	0	923	1	21	2	2	24
5	6	2	3	59	3	764	20	8	20	7
6	17	6	8	1	12	22	882	1	9	0
7	0	6	16	16	10	2	0	928	3	47
8	7	8	4	24	15	16	16	12	851	21
9	9	5	1	8	43	7	2	27	17	890

Tabla 14. *Tabla de confusión de la red neuronal con función de activación arcotangente de* 1 *capa oculta con* 20 *neuronas ocultas.*

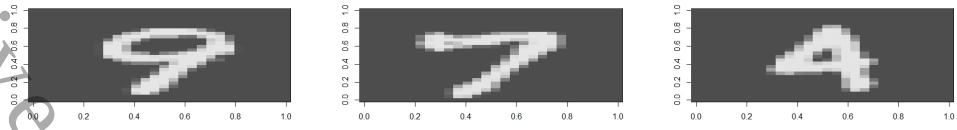
Se observa en las tablas de confusión con 3 neuronas ocultas, y cualquiera de las funciones sigmoide, que no se clasifican los dígitos en todas las clases $(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)$, únicamente en unas cuantas. A partir de 6 neuronas ocultas ya se observan todas las clases. Después de obtener las tablas de confusión respectivas, los cuales muestran la clasificación hecha de los dígitos, con ellas se obtuvieron las proporciones de error de clasificación, las cuales se proporcionan en la Tabla 15.

Neuronas ocultas \ Función de activación	Logística	Arcotangente
3	0.4749	0.6326
6	0.2065	0.1704
10	0.1141	0.1309
13	0.1139	0.1046
15	0.0899	0.1028
20	0.0868	0.0892

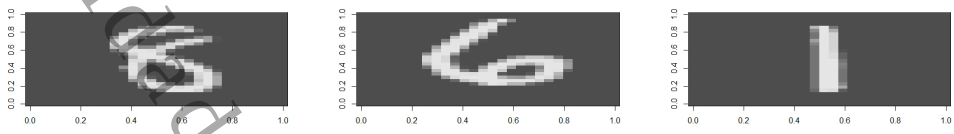
Tabla 15. *Proporciones de error de clasificación considerando* 1 *capa oculta y variando en* 3*,* 6*,* 10*,* 13*,* 15 *y* 20 *el número de neuronas ocultas.*

Por lo tanto, cuando se consideran 3 ó 6 neuronas ocultas, las proporciones de error con ambas funciones de activación sigmoide son elevadas, por lo que no se recomiendan estas cantidades de neuronas ocultas. Sin embargo, para esta clase de datos, a partir de 10 neuronas ocultas no hay mucha diferencia significativa entre las dos funciones de activación, porque resultan ser similares o existe una pequeña diferencia entre ellas, además de tener proporciones de error significativamente pequeñas, por lo que puede utilizarse cualquiera de ellas para ajustar la red neuronal. De igual forma, si se desea tener proporciones de error más pequeñas se tendrían que aplicar más neuronas ocultas dentro de la capa oculta, por lo que esto se deja a consideración del experimentador.

Finalmente, se muestran ejemplos de cómo la red neuronal clasifica dígitos, los cuales se presentan en las figuras 3.3 y 3.4.

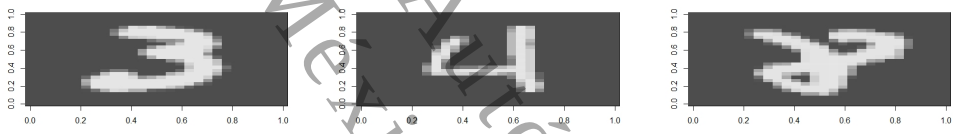


(a) Dígito 9, predicción 9. (b) Dígito 7, predicción 7. (c) Dígito 4, predicción 9.

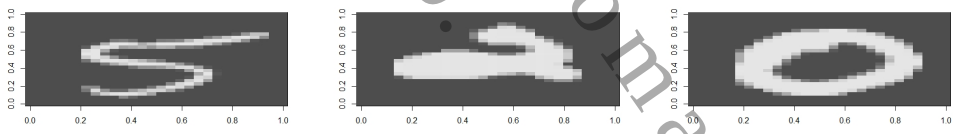


(d) Dígito 8, predicción 8. (e) Dígito 6, predicción 6. (f) Dígito 1, predicción 1.

Figura 3.3: Clasificación de dígitos correspondientes a una red neuronal de 1 capa oculta con 20 neuronas ocultas y función de activación sigmoide logística.



(a) Dígito 3, predicción 3. (b) Dígito 4, predicción 4. (c) Dígito 8, predicción 8.



(d) Dígito 5, predicción 5. (e) Dígito 2, predicción 4. (f) Dígito 0, predicción 0.

Figura 3.4: Clasificación de dígitos correspondientes a una red neuronal de 1 capa oculta con 20 neuronas ocultas y función de activación sigmoide arcotangente.

3.2. Reconocimiento de caras.

Se cuenta con datos de imágenes de rostros de varias personas en varias poses, los cuales son proporcionados por [9]. La base de datos consiste de imágenes de 20 personas diferentes, con aproximadamente 32 imágenes por persona variando su estado de ánimo (feliz, triste, enojado, neutral), la dirección en la que miran

(izquierda, derecha, de frente, arriba) y si portan o no gafas de sol, tal como puede verse en la Figura 3.5. En total se recopilaron 624 imágenes en escala de grises, cada una con una resolución de 64×60, con cada píxel de imagen descrito por un valor de la intensidad en la escala de grises entre 0 (negro) y 255 (blanco). Cada imagen fue transformada en un vector de entradas reales de dimensión $3840 = 64 \times 60$, mediante la concatenación de columnas. En cada análisis que se presenta se procedió a dividir de forma aleatoria en dos partes a la base de datos, el 70 % de los puntos como datos de entrenamiento (436 puntos) y el 30 % restante (188 puntos) conformaron los datos de prueba.

Figura 3.5: Imágenes de rostros de varias personas de 64×60 píxeles en diferentes poses, diferentes estados de ánimo y si portan o no gafas de sol.

El propósito de este ejemplo es mostrar que la clasificación de sujetos puede ser automatizada mediante redes neuronales artificiales. El proceso de clasificar grandes cantidades de caras se puede efectuar en lapsos de tiempo relativamente cortos; el procesamiento es realizado por un equipo de cómputo y el trabajo de catalogar las caras se realiza de manera objetiva. De igual forma, la misma base puede ser utilizada para clasificar a los sujetos de acuerdo a si portan o no gafas sol en la imagen. Del mismo modo, la base puede ser útil para clasificar la dirección en el que miran los sujetos, es decir, izquierda, derecha, de frente o arriba. Así mismo, la base de datos puede ser empleada para clasificar el estado de ánimo de los sujetos, es decir, enojado, feliz, neutral o triste.

Cabe mencionar que la clasificación de caras es útil en procedimientos de seguridad, videovigilancia, interacción persona-computadora con fines de entretenimiento, robótica y tarjetas “inteligentes” (pasaportes, licencias de conducir, registro de votantes), ver [7].

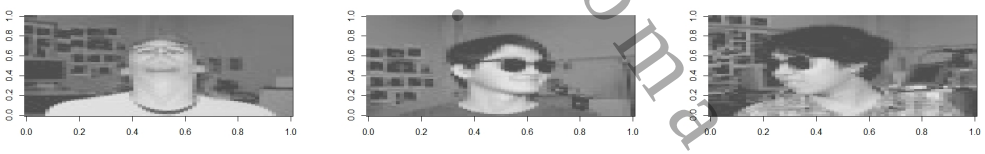
3.2.1. Clasificación de sujetos.

En este ejemplo se procedió a reconocer a un sujeto en particular de entre los demás, para ello se etiquetó al sujeto de interés, en este caso al sujeto "an2i", con la etiqueta 1 y a los demás sujetos que conforman la base de datos con la etiqueta 2. Se ajustaron redes neuronales utilizando la función sigmoide logística y la función sigmoide arcotangente, en cada caso se consideró una capa oculta con una neurona oculta. Las tablas de confusión con ambas funciones sigmoides fueron idénticas. A continuación se presenta la tabla de confusión.

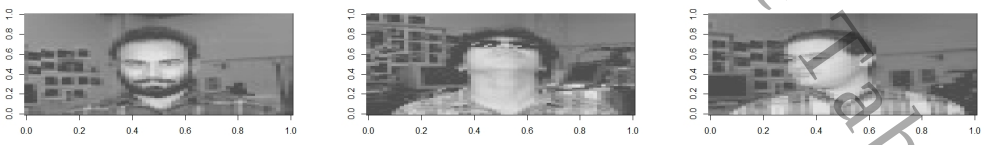
Reales \ **Predichos**	1	2
1	11	0
2	0	177

Tabla 16. *Tabla de confusión de la red neuronal con función de activación logística y arcotangente de* 1 *capa oculta con* 1 *neurona oculta.*

Se tiene también que la proporción de error con ambas funciones de activación es 0. Entonces para estos datos es suficiente considerar una red neuronal de una capa oculta y cualquiera de las funciones de activación. En las figuras 3.6 y 3.7 se muestra cómo cada red neuronal clasifica a los sujetos.

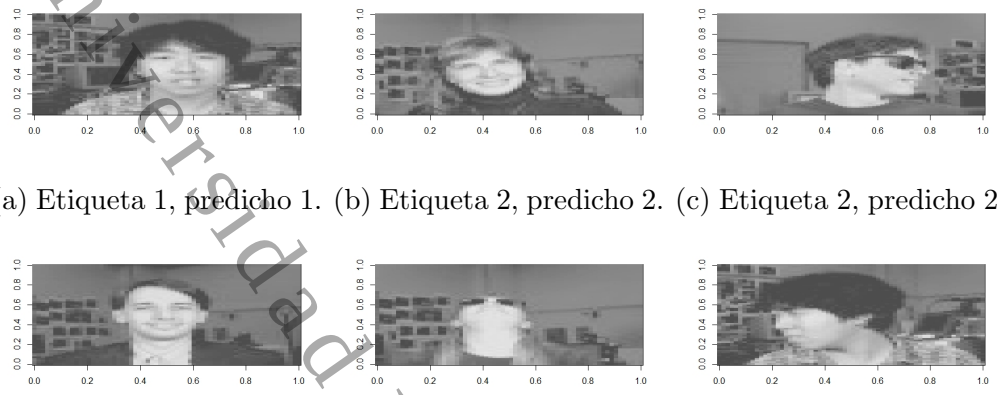


(a) Etiqueta 2, predicho 2. (b) Etiqueta 2, predicho 2. (c) Etiqueta 1, predicho 1.



(d) Etiqueta 2, predicho 2. (e) Etiqueta 1, predicho 1. (f) Etiqueta 2, predicho 2.

Figura 3.6: Clasificación de sujetos correspondientes a una red neuronal de 1 capa oculta con 1 neurona oculta y función de activación sigmoide logística. "sujeto an2i"=1, "otro"=2.



(a) Etiqueta 1, predicho 1. (b) Etiqueta 2, predicho 2. (c) Etiqueta 2, predicho 2.

(d) Etiqueta 2, predicho 2. (e) Etiqueta 2, predicho 2. (f) Etiqueta 1, predicho 1.

Figura 3.7: Clasificación de sujetos correspondientes a una red neuronal de 1 capa oculta con 1 neurona oculta y función de activación sigmoide arcotangente. "sujeto an2i"=1, "otro"=2.

3.2.2. Clasificación de uso de gafas de sol.

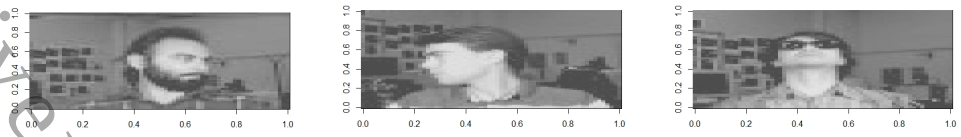
Para la clasificación de uso de gafas de sol de los sujetos, se procedió a etiquetarlos de la siguiente manera: "sin gafas de sol"=1, "con gafas de sol"=2. Se ajustaron redes neuronales utilizando la función sigmoide logística y la función sigmoide arcotangente; en cada caso se consideró una capa oculta con una neurona oculta. Las tablas de confusión con ambas funciones sigmoides fueron idénticas. A continuación se presenta la tabla de confusión.

Predichos / **Reales**	1	2
1	91	1
2	1	95

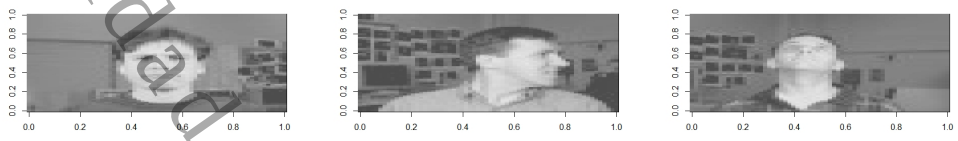
Tabla 17. *Tabla de confusión de la red neuronal con función de activación logística y arcotangente de* 1 *capa oculta con* 1 *neurona oculta.*

Una vez obtenidas las tablas de confusión, se observó que la proporción de error con ambas función de activación fue 0.0106. Cabe mencionar que al aumentar el número de neuronas ocultas dentro de la capa oculta se obtuvo la misma proporción de error que con una neurona oculta, con ambas funciones de activación, por lo que es suficiente utilizar una red neuronal de una capa oculta con una sola neurona oculta para la clasificación y cualquiera de las dos funciones sigmoide.

Por último se muestran ejemplos de cómo la red neuronal clasifica a los sujetos que portan gafas de sol o no en las imágenes, las cuales se presentan en las figuras 3.8 y 3.9.



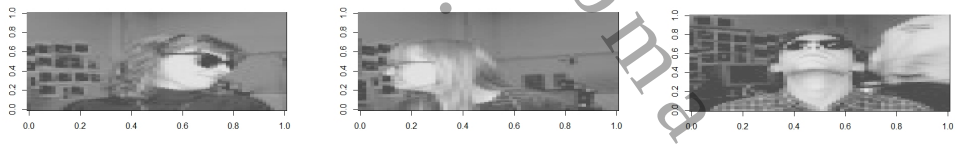
(a) Gafas 1, predicho 2. (b) Gafas 1, predicho 1. (c) Gafas 2, predicho 2.



(d) Gafas 1, predicho 1. (e) Gafas 1, predicho 1. (f) Gafas 1, predicho 1.

Figura 3.8: Clasificación de sujetos que portan gafas de sol o no, correspondiente a una red neuronal de 1 capa oculta con 1 neurona oculta y función de activación sigmoide logística. "sin gafas de sol"=1, "con gafas de sol"=2.

(a) Gafas 1, predicho 1. (b) Gafas 2, predicho 2. (c) Gafas 1, predicho 1.



(d) Gafas 2, predicho 2. (e) Gafas 1, predicho 1. (f) Gafas 2, predicho 1.

Figura 3.9: Clasificación de sujetos que portan gafas de sol o no, correspondiente a una red neuronal de 1 capa oculta con una neurona 1 y función de activación sigmoide arcotangente. "sin gafas de sol"=1, "con gafas de sol"=2.

3.2.3. Clasificación de dirección de miradas.

Para el caso de clasificar la dirección en la que miran los sujetos, se etiquetó dicha dirección de la siguiente forma: "izquierda"=1, "derecha"=2, "de frente"=3 y "arriba"=4. Se ajustaron redes neuronales usando la función sigmoide logística y la función sigmoide arcotangente. En cada caso se consideró una capa oculta, y

se varió el número de neuronas ocultas de 1 al 6. A continuación se presentan las tablas de confusión para 1, 2 y 3 neuronas ocultas.

Reales \ **Predichos**	1	2	3	4
1	41	0	0	0
2	6	31	11	0
3	5	15	25	7
4	0	0	0	47

Tabla 18. *Tabla de confusión de la red neuronal con función de activación logística de* 1 *capa oculta con* 1 *neurona oculta.*

Reales \ **Predichos**	1	2	3	4
1	40	1	0	0
2	0	48	0	0
3	3	6	30	13
4	5	1	2	39

Tabla 19. *Tabla de confusión de la red neuronal con función de activación arcotangente de* 1 *capa oculta con* 1 *neurona oculta.*

Reales \ **Predichos**	1	2	3	4
1	40	1	0	0
2	0	48	0	0
3	1	0	49	2
4	0	0	0	47

Tabla 20. *Tabla de confusión de la red neuronal con función de activación logística y arcotangente de* 1 *capa oculta con* 2 *neuronas ocultas.*

Reales \ **Predichos**	1	2	3	4
1	40	0	1	0
2	0	47	1	0
3	0	1	47	4
4	0	0	0	47

Tabla 21. *Tabla de confusión de la red neuronal con función de activación logística y arcotangente de* 1 *capa oculta con* 3 *neuronas ocultas.*

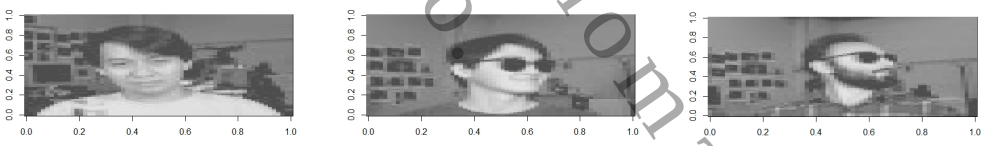
Una vez obtenidas las tablas de confusión se obtuvieron las proporciones de error de clasificación que se muestran en la Tabla 22.

Neuronas ocultas \ Función de activación	Logística	Arcotangente
1	0.2340	0.1649
2	0.0213	0.0213
3	0.0372	0.0372
4	0.0372	0.0372
5	0.0213	0.0213
6	0.0372	0.0372

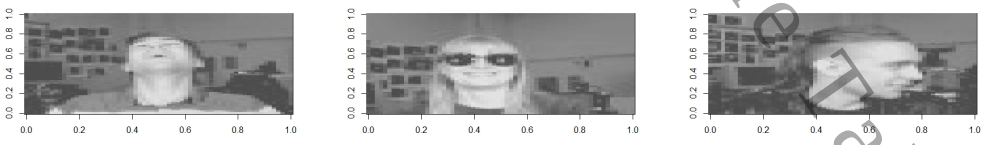
Tabla 22. *Proporciones de error de clasificación considerando una capa oculta y variando de* 1 *a* 6 *neuronas ocultas.*

Obsérvese que al considerar una neurona oculta, las proporciones de error con ambas funciones de activación son significativamente elevadas, por lo que no son recomendables para la clasificación de la dirección de la mirada de los sujetos. No obstante, a partir de 2 neuronas ocultas las proporciones de error son muy pequeñas y similares al aplicar ambas funciones de activación, por lo que es recomendable utilizar una red neuronal de una capa oculta con al menos 2 neuronas ocultas y cualquiera de las dos funciones sigmoide para clasificar la dirección de la mirada.

Finalmente, en las figuras 3.10 y 3.11 se muestran ejemplos de cómo la red neuronal clasifica la dirección de la mirada de los sujetos.

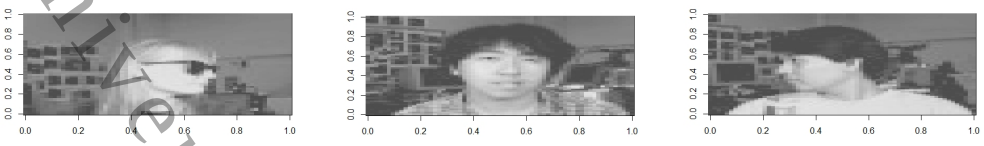


(a) Mirada 3, predicho 3. (b) Mirada 1, predicho 1. (c) Mirada 1, predicho 3.

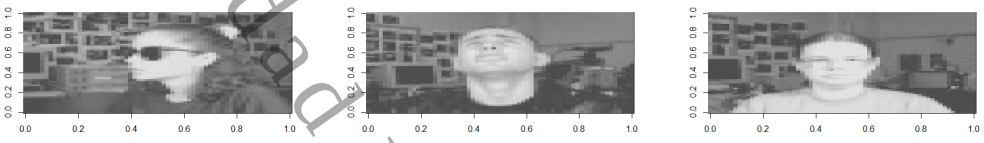


(d) Mirada 4, predicho 4. (e) Mirada 3, predicho 3. (f) Mirada 1, predicho 1.

Figura 3.10: Clasificación de dirección de la mirada de los sujetos, correspondiente a una red neuronal de 1 capa oculta con 2 neuronas ocultas y función de activación sigmoide logística. “izquierda”=1, “derecha”=2, “de frente”=3, “arriba”=4.



(a) Mirada 1, predicho 1. (b) Mirada 3, predicho 4. (c) Mirada 2, predicho 2.



(d) Mirada 2, predicho 2. (e) Mirada 4, predicho 4. (f) Mirada 3, predicho 3.

Figura 3.11: Clasificación de la dirección de la mirada de los sujetos, correspondiente a una red neuronal de 1 capa oculta con 2 neuronas ocultas y función de activación sigmoide arcotangente. "izquierda"=1, "derecha"=2, "de frente"=3, "arriba"=4.

3.2.4. Clasificación de emociones.

Finalmente, para clasificar las emociones de los sujetos, se etiquetaron las emociones de la siguiente manera: "enojado"=1, "feliz"=2, "neutral"=3 y "triste"=4. Se ajustaron redes neuronales usando la función sigmoide logística y la función sigmoide arcotangente. En cada caso se consideró una capa oculta y se varió el número de neuronas ocultas de 1 a 6. A continuación se presentan las tablas de confusión para 1 y 6 neuronas ocultas.

Reales \ **Predichos**	1	2	3	4
1	55	0	0	0
2	24	20	2	0
3	1	5	13	21
4	0	0	0	47

Tabla 23. *Tabla de confusión de la red neuronal con función de activación logística de* 1 *capa oculta con* 1 *neurona oculta.*

Reales \ **Predichos**	1	2	3	4
1	55	0	0	0
2	24	20	2	0
3	1	4	14	21
4	0	0	0	47

Tabla 24. *Tabla de confusión de la red neuronal con función de activación arcotangente de* 1 *capa oculta con* 1 *neurona oculta.*

Reales \ Predichos	1	2	3	4
1	55	0	0	0
2	0	44	1	1
3	0	2	37	1
4	0	0	0	47

Tabla 25. *Tabla de confusión de la red neuronal con función de activación logística de* 1 *capa oculta con* 6 *neuronas ocultas.*

Reales \ Predichos	1	2	3	4
1	55	0	0	0
2	1	44	0	1
3	0	2	38	0
4	0	0	0	47

Tabla 26. *Tabla de confusión de la red neuronal con función de activación arcotangente de* 1 *capa oculta con* 6 *neuronas ocultas.*

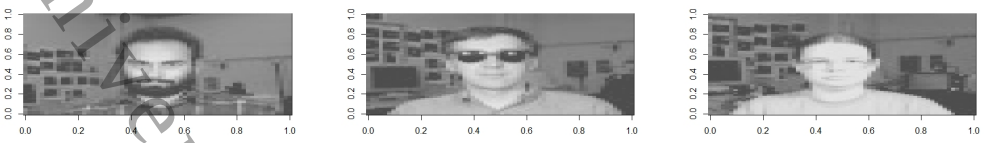
Después de obtener las tablas de confusión se obtuvieron las proporciones de error que se muestran en la Tabla 27.

Neuronas ocultas \ Función de activación	Logística	Arcotangente
1	0.2819	0.2766
2	0.0638	0.1543
3	0.0319	0.0745
4	0.0266	0.0319
5	0.0213	0.0372
6	0.0266	0.0213

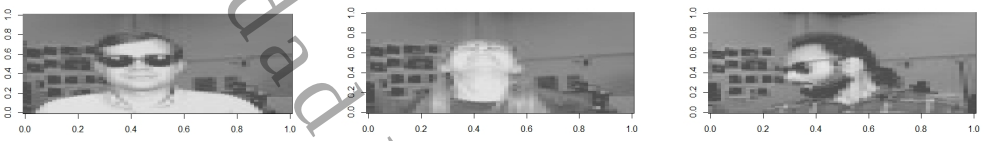
Tabla 27. *Proporciones de error de clasificación considerando una capa oculta y variando de* 1 *a* 6 *neuronas ocultas.*

Al observar la tabla anterior y considerar una neurona oculta, las proporciones de error con ambas funciones de activación son significativamente elevadas, por lo que este tipo de red neuronal no es recomendable para la clasificación de las emociones de los sujetos. No obstante, a partir de 2 neuronas ocultas, es recomendable utilizar la función de activación sigmoide logística debido a que su proporción de error de clasificación, en general, es menor que la correspondiente a la función de activación sigmoide arcotangente. Sin embargo, a partir de 3 neuronas ocultas con la función de activación sigmoide arcotangente se tienen proporciones de error de clasificación muy pequeñas.

Finalmente, en las figuras 3.12 y 3.13 se muestran ejemplos de cómo la red neuronal clasifica las emociones de los sujetos.



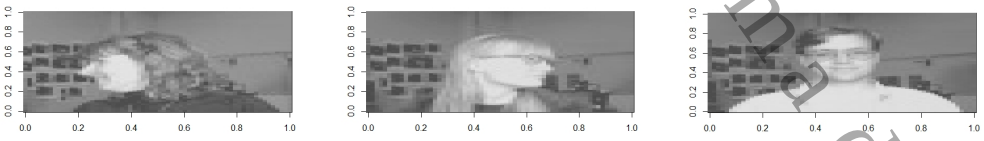
(a) Emoción 4, predicho 4. (b) Emoción 3, predicho 3. (c) Emoción 1, predicho 1.



(d) Emoción 3, predicho 4. (e) Emoción 2, predicho 2. (f) Emoción 1, predicho 1.

Figura 3.12: Clasificación de las emociones de los sujetos, correspondiente a una red neuronal de 1 capa oculta con 6 neuronas ocultas y función de activación sigmoide logística. "enojado"=1, "feliz"=2, "neutral"=3, "triste"=4.

(a) Emoción 1, predicho 1. (b) Emoción 4, predicho 4. (c) Emoción 3, predicho 3.



(d) Emoción 2, predicho 2. (e) Emoción 4, predicho 4. (f) Emoción 2, predicho 1.

Figura 3.13: Clasificación de las emociones de los sujetos, correspondiente a una red neuronal de 1 capa oculta con 6 neuronas ocultas y función de activación sigmoide arcotangente. "enojado"=1, "feliz"=2, "neutral"=3, "triste"=4.

Conclusiones.

En este trabajo, mediante redes neuronales artificiales, aplicando una función de activación logística o arcotangente y variando el número de neuronas ocultas dentro de una capa oculta, se clasificaron patrones encontrados en la literatura, y se evaluaron las redes neuronales mediante sus tablas de confusión y sus proporciones de error de clasificación.

En el ejemplo presentado de los datos de los dígitos escritos manualmente, tomados del conjunto de datos del Modified National Institute of Standards and Technology (MNIST), se observó que es suficiente aplicar una red neuronal con una capa oculta e ir variando el número de neuronas ocultas, y que al considerar 3 ó 6 neuronas ocultas las proporciones de error con ambas funciones de activación sigmoide fueron elevadas, por lo que no es recomendable utilizar estas cantidades de neuronas. Sin embargo, a partir de 10 neuronas ocultas no hay mucha diferencia en utilizar ambas funciones de activación, porque resulta que tienen proporciones de error similares y pequeñas, por lo que puede utilizarse cualquiera de ellas para ajustar la red neuronal. Si se desea tener proporciones de error más pequeñas, se tendría que agregar más neuronas ocultas dentro de la capa oculta.

En el ejemplo de la base de datos de las imágenes de rostros de varias personas en varias poses, se observó que basta con aplicar una red neuronal de una capa oculta e ir variando el número de neuronas ocultas para todos los escenarios considerados. Cuando se consideró la clasificación de un sujeto de interés de la base de datos, bastó con aplicar una red neuronal de una capa oculta con una neurona oculta, con ambas funciones de activación, para alcanzar una proporción de error de 0, por lo que es muy recomendable utilizar una sola neurona. De igual modo, en el escenario de clasificación de uso de gafas de sol, solo fue necesario una red neuronal de una capa oculta con una neurona oculta, con ambas funciones de activación, para obtener una proporción de error muy baja; al aumentar el número de neuronas ocultas dentro de la capa oculta, para este caso se obtuvo la misma proporción de error con ambas funciones de activación. Para el caso de la clasificación de la dirección de miradas de las caras, se observó que al considerar una neurona oculta las proporciones de error, con ambas funciones de activación, fueron significativamente elevadas, por lo que no es recomendable utilizar una sola neurona oculta para la clasificación de la dirección de las miradas; no obstante, a partir de 2 neuronas ocultas se obtuvieron proporciones de error muy pequeñas

y similares al aplicar ambas funciones de activación, por lo que es recomendable utilizar al menos 2 neuronas ocultas en la clasificación de la dirección de la mirada. Por último, en la clasificación de las emociones de los sujetos, al considerar una neurona oculta las proporciones de error, con ambas funciones de activación, son significativamente elevadas, de modo que utilizar una neurona oculta no es recomendable para la clasificación de las emociones de los sujetos; en cambio, para esta clase de datos, a partir de 2 neuronas ocultas es recomendable utilizar la función de activación sigmoide logística; sin embargo, a partir de 3 neuronas ocultas con la función de activación sigmoide arcotangente se puede observar una proporción de error pequeña, por lo que es recomendable para la clasificación de las emociones de los sujetos.

En los ejemplos de aplicación presentados en este trabajo se observó un buen comportamiento de las redes neuronales artificiales para clasificar patrones. También vimos en estos ejemplos que es posible tener una buena clasificación considerando una sola capa oculta y pocas neuronas ocultas. Cuando el número de clases aumenta, por lo general se requieren más neuronas ocultas para tener una proporción de error de clasificación baja.

Apéndice.

A continuación se presentan los códigos del software Rstudio que se utilizaron en este trabajo, donde se empleó principalmente la paquetería "nnet".

Ejemplo 2: Red neuronal.

```
library(neuralnet)
library(ggplot2)

neuronas_ocultas=10
n_0=1000
error_iteracion=c()
MSE.RN_iteracion=c()
MSE.ML_iteracion=c()

prediccionesX=matrix(rep(0,
(n_0*0.3)*neuronas_ocultas),
ncol=neuronas_ocultas)
prediccionesY=matrix(rep(0,
(n_0*0.3)*neuronas_ocultas),
ncol=neuronas_ocultas)

for(i in 1:neuronas_ocultas){

set.seed(65)# Genera números aleatorios

#DATOS DE PRUEBA.

#Generamos las entradas X's.
X01=rnorm(1000)
X02=rnorm(1000)
X0=matrix(c(X01,X02),nrow =2,ncol=1000,
byrow =T)

#Valores de la a's.
a1=matrix(c(3,3))
a2=matrix(c(3,-3))

#Valores de la a's.
A01=t(a1)%*%X0
A02=t(a2)%*%X0

#Función sigmoide estándar
sigma01=1/(1+exp(-A01))

#Generamos la Z normal estándar

Z0=rnorm(1000)

Y01=sigma01+((A02)^2)+0.30*Z0

#DATOS DE ENTRENAMIENTO.
data0=matrix(c(X01,X02,Y01),nrow =1000,
      ncol=3,byrow=F)
datos=as.data.frame(data0)
colnames(datos)[1]='X1'
colnames(datos)[2]='X2'
colnames(datos)[3]='Y'

n=nrow(datos)

# se toma una # muestra aleatoria
muestra=sample(n,n*0.7)


# datos de entrenamiento
train=datos[muestra,]

# datos de prueba
prueba=datos[-muestra,]

lm.fit=glm(Y~., data=train)
#summary(lm.fit)

# MODELO DE REGRESIÓN LINEAL

pr.lm=predict(lm.fit,prueba)
MSE.ML=sum((pr.lm - prueba$Y)^2)/nrow(prueba)
MSE.ML_iteracion[i]=MSE.ML

#NORMALIZACION DE VARIABLES-------------------

# valores máximos  para el entrenamiento
maxs=apply(train,2,max)

# valores mínimos  para el entrenamiento
mins=apply(train,2,min)

#Mostrando maxs y mins
#maxs
#mins
```

```
datos_nrm=as.data.frame(scale(datos,
center=mins,scale=maxs - mins))
#datos_nrm

train_nrm=datos_nrm[muestra,]
#train_nrm

test_nrm=datos_nrm[-muestra, ]
#test_nrm


#Modelo---------------------------------
rn=neuralnet(Y~X1+X2,
             data=train_nrm,
             hidden=i,
             threshold= 0.05,
             act.fct="logistic",
             linear.output=F,
             algorithm     = "rprop+"
)
#rn

vector_error=rn$result.matrix
errores=vector_error[1,]
error_iteracion[i]=errores


# Predicción-----------------------------
rnprediccion=compute(rn,
             within(test_nrm,rm(Y)))
#a=rnprediccion$net.result

# se transforma el valor escalar al
# valor nominal original
```

3.1. Dígitos escritos a mano (MNIST).

```
#librerias
library(nnet)
library(ggplot2)
library(lattice)
library(caret)

#Directorio
setwd("C:/Users/Edgar/Dropbox
/Correcciones-Edgar-Alamilla/
Maestria Edgar/
Clasificaciondedigitos_mnist")

#Datos de entrenamiento
train=read.csv("mnist_train.csv",header=T)
clase=factor(train$label)

# Visualización de dígitos

#En sample_2 se observa la concatenación
#del dígito 2
sample_2= matrix(as.numeric(train[17,-1]),
          nrow = 28,byrow = TRUE)
```

```
Y.predict=rnprediccion$net.result*(max(datos$Y)
          -min(datos$Y))+min(datos$Y)

Y.real=(test_nrm$Y)*(max(datos$Y)-min(datos$Y))
        +min(datos$Y)

##################################################
prediccionesX[,i]=matrix(c(Y.real),byrow=F)
prediccionesY[,i]=matrix(c(Y.predict),byrow=F)
##################################################

#---SUMA DE ERROR CUADRÁTICO DE LA RED NEURONAL---

mse.RN=sum((Y.real - Y.predict)^2)/nrow(test_nrm)
MSE.RN_iteracion[i]=mse.RN

}

error_iteracion
MSE.ML_iteracion
MSE.RN_iteracion
#prediccionesX
#prediccionesY

#--------GRÁFICOS-------------------------------

#Red neuronal
plot(rn)

#GRÁFICA DE ERRORES.
capas=1:10
plot(capas,error_iteracion, type='o',
     xlab="Capas Ocultas en la red neuronal",
     ylab="Errores de entrenamiento")



sample_2
image(sample_2, col = grey.colors(255))

#Rotación de la imagen
rotar=function(x) t(apply(x, 2, rev))
image(rotar(sample_2),col=grey.colors(255))

#visualización de 6 dígitos
par(mfrow=c(2,3))
lapply(1:6,
function(x) image(rotar(matrix(unlist(
             train[x,-1]),nrow = 28,
             byrow = TRUE)),
             col=grey.colors(255),
             xlab=train[x,1]
   )
)

#Las redes neuronales  con función de
#activación sigmoide logístico.

#Aquí el número de capas ocultas va ir
#variando en 3, 6, 10, 13, 15 y 20. En
#este caso se muestra una red neuronal
```

```
#con 3 neuronas ocultas.

mod_log_3=nnet(clase~ ., data=train, size=3,
               maxit=100000, MaxNWts=100000,
               decay=0.001, rang=0.05,
               na.action = na.omit)

#Datos de prueba
test=read.csv("mnist_test.csv",header=T)
clase_test=factor(test$label)

#Predicción
pred_log_3=predict(mod_log_3,newdata =test,
                   type = "class" )

#Tabla de confusión
tabla_log_3=table(clase_test, pred_log_3,
           dnn=c("Reales", "Predichos") )
tabla_log_3

#Proporción de error en la tabla de
#confusión tabla_log_3

n3_log=length(tabla_log_3[1,])
sumacol_log3=c()
maxcol_log3=c()
for(i in 1:n3_log)
  {
  sumacol_log3[i]=sum(tabla_log_3[,i])
  maxcol_log3[i]=max(tabla_log_3[,i])
  }
#sumacol_log3
#maxcol_log3
sumtotal_log3=sum(sumacol_log3)
error_x_digito_log_3=sumacol_log3-
                           maxcol_log3
proporcion_error_x_digito_log3=
    error_x_digito_log_3/sumtotal_log3
error_log3=sum(proporcion_error_x_
               digito_log3)
error_log3
########################################
#Las redes neuronales  con función de
activación sigmoide arcotangente.

#Aquí el número de capas ocultas va ir
#variando en 3, 6, 10, 13, 15 y 20. En
#este caso se muestra una red neuronal
#con 3 neuronas ocultas.

artan=function(x) ((2/pi)*atan(x))

mod_arctan_3=nnet(clase~ .,
                  data=train, size=3,
maxit=100000,MaxNWts=100000,
decay=0.001, rang=0.05,
na.action = na.omit, act.fct="artan")

pred_arctan_3=predict(mod_arctan_3,
        newdata =test,type = "class")

#Tabla de confusión
tabla_arctan_3=table(clase_test,
        pred_arctan_3,
        dnn=c("Reales", "Predichos"))
tabla_arctan_3

#Proporción de error en la tabla de
#confusión tabla_arctan_3

n3_arctan=length(tabla_arctan_3[1,])
sumacol_arctan3=c()
maxcol_arctan3=c()
for(i in 1:n3_arctan)
{
sumacol_arctan3[i]=sum(tabla_arctan_3[,i])
maxcol_arctan3[i]=max(tabla_arctan_3[,i])
}
sumtotal_arctan3=sum(sumacol_arctan3)
error_x_digito_arctan_3=sumacol_arctan3-
                           maxcol_arctan3
proporcion_error_x_digito_arctan3=
error_x_digito_arctan_3/sumtotal_arctan3
error_arctan3=sum(proporcion_error_x_
                  digito_arctan3)
error_arctan3

##De igual manera se anexa la predicción de la
## red neuronal con función de activación
##logístico con 20 neuronas ocultas dado una
##sola muestra de los datos de prueba.

num_aleatorio=sample(1:sumtotal_log20,1,
                                replace=F)
num_selec=c("El número seleccionado es =>",
            test[num_aleatorio,1])
num_selec
# Visualización del dígito
sample_aleatorio= matrix(as.numeric(
             test[num_aleatorio,-1]),
             nrow = 28, byrow = TRUE)
rotar=function(x) t(apply(x, 2, rev))
image(rotar(sample_aleatorio),
      col = grey.colors(255))

prediccion_logistc_20=predict(mod_log_20,
           newdata =test[num_aleatorio,],
           type = "class")
pre_logistic20=c("La predicción de la
                  red neuronal es:",
                 prediccion_logistc_20)
pre_logistic20

##De igual manera se anexa la predicción
##de la red neuronal con función de
##activación arcotangente con 20 neuronas
##ocultas dado una sola muestra de los
##datos de prueba.

num_aleatorio=sample(1:sumtotal_arctan20,1,
                     replace=F)
num_selec=c("El número seleccionado es =>",
```

```
          test[num_aleatorio,1])
num_selec
# Visualización del dígito
sample_aleatorio= matrix(as.numeric(
            test[num_aleatorio,-1]),
            nrow = 28, byrow = TRUE)
rotar=function(x) t(apply(x, 2, rev))
image(rotar(sample_aleatorio),
      col = grey.colors(255))
```

3.2.1. Clasificación de sujetos.

```
library(nnet)
library(lattice)
library(ggplot2)
library(caret)
library(readxl)

#Directorio #cambiar directorio al dropbox
setwd("C:/Users/Edgar/Desktop
/Rutinas R/Mitchell/faces1.0")

#Data
data=read_excel("faces1.0.xlsx")

#Categorias
#Clasificación del sujeto:
#(sujeto1=1,sujeto2=...=sujeto20=2)
sujeto=factor(data$label)

#Visualización de sujetos
n_0=length(sujeto)#número total de fotos
n=sample(1:n_0,1,replace=FALSE)

#En cara_n se observa la concatenación
# de la foto n
cara_n= matrix(as.numeric(data[n,-1]),
       nrow =60, byrow = TRUE)
#image(cara_n, col = grey.colors(255))
#Rotación de la imagen
rotar=function(x) t(apply(x, 2, rev))
image(rotar(cara_n), col = grey.colors(255))


#Vector aleatorio de longitud
muestra=sample(1:n_0,6,replace=FALSE)
#visualización de 6 sujetos
par(mfrow=c(2,3))
lapply(muestra,
 function(x) image(rotar
 (matrix(unlist(data[x,-1]),
                 nrow = 60,
                 byrow = TRUE)),
                 col=grey.colors(255),
                 xlab=data[x,1]
  )
)
```

```
prediccion_arctan_20=predict(mod_arctan_20,
          newdata=test[num_aleatorio,],
          type = "class")
pre_arc20=c("La predicción de la red neuronal es:"
           , prediccion_arctan_20)
pre_arc20

prob_acierto_arc20=
           sum(maxcol_arctan20)/sumtotal_arctan20
proba_exito_arc20=c("La probabilidad de acierto es =>"
                   ,prob_acierto_arc20)
proba_exito_arc20


#Datos de entrenamiento
muestra01=sample(n_0,n_0*0.7)
train=data[muestra01,]

sujeto_train=factor(train$label)

#Datos de prueba
test=data[-muestra01,]
sujeto_test=factor(test$label)

################################################
#Una red neuronal de una capa oculta con 1
#neurona oculta función de activación logistica

mod_log_1=nnet(sujeto_train~.,data=train,
          size=1,maxit=1.0e+9,
          MaxNWts=1.0e+9,decay=0.001,
          rang=0.05,na.action = na.omit)

pred_log_1=predict(mod_log_1,newdata=test,
                   type = "class" )

#tabla de confusión
tabla_log_1=table(sujeto_test, pred_log_1,
                  dnn=c("Reales", "Predichos") )
tabla_log_1

#Proporción de error en la tabla de
#confusión tabla_log_1

n1_log=length(tabla_log_1[1,])
sumacol_log1=c()
maxcol_log1=c()
for(i in 1:n1_log)
{
  sumacol_log1[i]=sum(tabla_log_1[,i])
  maxcol_log1[i]=max(tabla_log_1[,i])

}
sumtot_log1=sum(sumacol_log1)
error_x_digito_log1=sumacol_log1-maxcol_log1
prop_error_lg1=error_x_digito_log1/sumtot_log1
error_log1=sum(prop_error_lg1)
error_log1

##Prediccion de la red neuronal
##para logistic 1
num_aleatorio=sample(1:sumtot_log1,1,
```

```
                    replace=F)
num_selec=c("La dirección del rostro es==>",
            test[num_aleatorio,1])
num_selec
# Visualización del dígito
sample_aleatorio= matrix(as.numeric(
                  test[num_aleatorio,-1]),
                  nrow = 60, byrow = TRUE)
rotar=function(x) t(apply(x, 2, rev))
image(rotar(sample_aleatorio),
      col = grey.colors(255))

prediccion_logistc_1=predict(mod_log_1,
             newdata =test[num_aleatorio,],
             type = "class")
pre_logistic1=c("La predicción de la red
                neuronal es:",
                prediccion_logistc_1)
pre_logistic1

########################################

#Una red neuronal de una capa oculta con 1
#neurona oculta función de activación
#sigmoide arcotangente

artan=function(x) ((2/pi)*atan(x))

mod_arctan_1=nnet(sujeto_train~.,data=train,
    size=1,maxit=1.0e+9,MaxNWts=1.0e+9,
    decay=0.001, rang=0.05,
    na.action = na.omit, act.fct="artan")

pred_arctan_1=predict(mod_arctan_1,
              newdata=test,type = "class")

#tabla de confusión
tabla_arctan_1=table(sujeto_test,
               pred_arctan_1,
```

3.2.2. Clasificación del uso de gafas de sol.

```
library(nnet)
library(lattice)
library(ggplot2)
library(caret)
library(readxl)

#Directorio
setwd("C:/Users/Edgar/Desktop/
       Rutinas R/Mitchell/faces4")

#Data
data=read_excel("faces4.xlsx")

#Categorias
#El sujeto porta gafas de sol(No=1,Si=2)
lentes=factor(data$label)
```

```
               dnn=c("Reales","Predichos"))
tabla_arctan_1

#Proporción de error en la tabla de
#confusión tabla_arc_1

n1_arctan=length(tabla_arctan_1[1,])
sumacol_arctan1=c()
maxcol_arctan1=c()
for(i in 1:n1_arctan)
{
sumacol_arctan1[i]=sum(tabla_arctan_1[,i])
maxcol_arctan1[i]=max(tabla_arctan_1[,i])

}
sumtot_arctan1=sum(sumacol_arctan1)
error_x_digito_atg1=sumacol_arctan1-maxcol_arctan1
prop_error_x_dig_atg1=error_x_dig_atg1/sumtot_atg1
error_arctan1=sum(prop_error_x_dig_atg1)
error_arctan1

##Prediccion de la red neuronal para arctan 1
num_aleatorio=sample(1:sumtot_arctan1,1,
                     replace=F)
num_selec=c("El estado de animo del sujeto es==>",
            test[num_aleatorio,1])
num_selec
# Visualización del dígito
sample_aleatorio=matrix(as.numeric(test[num_aleatorio,-1]),
                        nrow = 60, byrow = TRUE)
rotar=function(x) t(apply(x, 2, rev))
image(rotar(sample_aleatorio),
      col = grey.colors(255))

prediccion_arctan_1=predict(mod_arctan_1,
                    newdata =test[num_aleatorio,],
                    type = "class")
pre_arctan1=c("La predicción de la red neuronal es:",
              prediccion_arctan_1)
pre_arctan1



#Visualización de sujetos
n_0=length(lentes)#número total de fotos
n=sample(1:n_0,1,replace=FALSE)

#En cara_n se observa la concatenación
#de la foto n
cara_n= matrix(as.numeric(data[n,-1]),
               nrow =60, byrow = TRUE)
#image(cara_n, col = grey.colors(255))
#Rotación de la imagen
rotar=function(x) t(apply(x, 2, rev))
image(rotar(cara_n),col=grey.colors(255))


#Vector aleatorio de longitud
muestra=sample(1:n_0,6,replace=FALSE)
#visualización de 6 sujetos
par(mfrow=c(2,3))
lapply(muestra,
```

```
function(x) image(rotar(matrix(unlist(
                data[x,-1]),
                nrow = 60,
                byrow = TRUE)),
                col=grey.colors(255),
                xlab=data[x,1]
  )
)

#Datos de entrenamiento
muestra01=sample(n_0,n_0*0.7)
train=data[muestra01,]

lentes_train=factor(train$label)

#Datos de prueba
test=data[-muestra01,]
lentes_test=factor(test$label)

#######################################
#Una red neuronal de una capa oculta con
#una 1 neurona oculta función de
#activación logistica

mod_log_1=nnet(lentes_train~.,data=train,
               size=1, maxit=1.0e+9,
               MaxNWts=1.0e+9,decay=0.001,
               rang=0.05,
```

3.2.3. Clasificación de dirección de mirada.

```
library(nnet)
library(lattice)
library(ggplot2)
library(caret)
library(readxl)

#Directorio #cambiar directorio al dropbox
setwd("C:/Users/Edgar/Desktop/
      Rutinas R/Mitchell/faces2")

#Data
data=read_excel("faces2.xlsx")

#Categorias
#La orientación de la mirada del sujeto:
#(Izquerda=1,Derecha=2,De frente=3,Arriba=4)
mirada=factor(data$label)

#Visualización de sujetos
n_0=length(mirada)#número total de fotos
n=sample(1:n_0,1,replace=FALSE)

#En cara_n se observa la concatenación
#de la foto n
cara_n= matrix(as.numeric(data[n,-1]),
               nrow =60, byrow = TRUE)
#image(cara_n, col = grey.colors(255))
#Rotación de la imagen
rotar=function(x) t(apply(x, 2, rev))
                na.action = na.omit)

pred_log_1=predict(mod_log_1,newdata=test,
                   type = "class" )

#tabla de confusión
tabla_log_1=table(lentes_test,pred_log_1,
           dnn=c("Reales", "Predichos"))
tabla_log_1

#Proporción de error en la tabla de
#confusión tabla_log_1

n1_log=length(tabla_log_1[1,])
sumacol_log1=c()
maxcol_log1=c()
for(i in 1:n1_log)
{
sumacol_log1[i]=sum(tabla_log_1[,i])
maxcol_log1[i]=max(tabla_log_1[,i])

}
sumtotal_log1=sum(sumacol_log1)
error_x_digito_log_1=sumacol_log1-maxcol_log1
proporcion_error_x_digito_log1=
          error_x_digito_log_1/sumtotal_log1
error_log1=sum(proporcion_error_x_digito_log1)
error_log1



image(rotar(cara_n),col = grey.colors(255))

#Vector aleatorio de longitud
muestra=sample(1:n_0,6,replace=FALSE)
#visualización de 6 sujetos
par(mfrow=c(2,3))
lapply(muestra,
function(x) image(rotar(matrix(
            unlist(data[x,-1]),
            nrow = 60,
            byrow = TRUE)),
            col=grey.colors(255),
            xlab=data[x,1]
   )
)

#Datos de entrenamiento
muestra01=sample(n_0,n_0*0.7)
train=data[muestra01,]

mirada_train=factor(train$label)

#Datos de prueba
test=data[-muestra01,]
mirada_test=factor(test$label)

#########################################
#Una red neuronal de una capa oculta con una
#1 neurona oculta
#Función de activación logistico
```

```
mod_log_1=nnet(mirada_train~.,data=train,
               size=1, maxit=1.0e+9,
               MaxNWts=1.0e+9,decay=0.001,
               rang=0.05,
               na.action = na.omit)

pred_log_1=predict(mod_log_1,newdata=test,
                   type="class" )

#tabla de confusión
tabla_log_1=table(mirada_test, pred_log_1,
           dnn=c("Reales", "Predichos"))
tabla_log_1

#Proporción de error en la tabla de
#confusión tabla_log_1

n1_log=length(tabla_log_1[1,])
sumacol_log1=c()
maxcol_log1=c()
for(i in 1:n1_log)
{
sumacol_log1[i]=sum(tabla_log_1[,i])
maxcol_log1[i]=max(tabla_log_1[,i])

}
sumtotal_log1=sum(sumacol_log1)
error_x_digito_log_1=sumacol_log1-
                     maxcol_log1
prop_error_x_dig_log1=
  error_x_digito_log_1/sumtotal_log1
error_log1=sum(prop_error_x_dig_log1)
error_log1

##Prediccion de la red neuronal para
##logistic 1
num_aleatorio=sample(1:sumtotal_log1,1
                           ,replace=F)
num_selec=c("La dirección del rostro es=>",
            test[num_aleatorio,1])
num_selec
# Visualización del dígito
sample_aleatorio= matrix(as.numeric(
            test[num_aleatorio,-1]),
            nrow = 60, byrow = TRUE)
rotar=function(x) t(apply(x, 2, rev))
image(rotar(sample_aleatorio),
        col = grey.colors(255))

prediccion_logistc_1=predict(mod_log_1,
         newdata=test[num_aleatorio,],
                      type = "class")
pre_logistic1=c("La predicción de la red
                neuronal es:",
               prediccion_logistc_1)
pre_logistic1
```

```
#Una red neuronal de una capa oculta
#con una 1 neurona oculta Función de
#activación arcotangente.
artan=function(x) ((2/pi)*atan(x))

mod_arctan_1=nnet(mirada_train~., data=train,
                  size=1,maxit=1.0e+9,
                  MaxNWts=1.0e+9,decay=0.001,
                  rang=0.05,na.action = na.omit,
                  act.fct="artan")

pred_arctan_1=predict(mod_arctan_1,
                      newdata=test,type="class")

#tabla de confusión
tabla_arctan_1=table(mirada_test,pred_arctan_1,
                     dnn=c("Reales", "Predichos"))
tabla_arctan_1

#Proporción de error en la tabla de
#confusión tabla_arc_1

n1_arctan=length(tabla_arctan_1[1,])
sumacol_arctan1=c()
maxcol_arctan1=c()
for(i in 1:n1_arctan)
{
sumacol_arctan1[i]=sum(tabla_arctan_1[,i])
maxcol_arctan1[i]=max(tabla_arctan_1[,i])

}
sumtotal_arctan1=sum(sumacol_arctan1)
error_x_digito_arctan_1=sumacol_arctan1-maxcol_arctan1
proporcion_error_x_digito_arctan1=
            error_x_digito_arctan_1/sumtotal_arctan1
error_arctan1=sum(proporcion_error_x_digito_arctan1)
error_arctan1

##Prediccion de la red neuronal para arctan 1
num_aleatorio=sample(1:sumtotal_arctan1,1,replace=F)
num_selec=c("El estado de animo del sujeto es=>",
            test[num_aleatorio,1])
num_selec
# Visualización del dígito
sample_aleatorio= matrix(as.numeric(
              test[num_aleatorio,-1]),
              nrow = 60, byrow = TRUE)
rotar=function(x) t(apply(x, 2, rev))
image(rotar(sample_aleatorio),
           col=grey.colors(255))

prediccion_arctan_1=predict(mod_arctan_1,
                    newdata=test[num_aleatorio,],
                                type = "class")
pre_arctan1=c("La predicción de la red neuronal es:",
              prediccion_arctan_1)
pre_arctan1
```

3.2.4. Clasificación de emociones.

```
library(nnet)
library(lattice)
library(ggplot2)
library(caret)
library(readxl)

#Directorio
setwd("C:/Users/Edgar/Desktop
       /Rutinas R/Mitchell/faces3")

#Data
data=read_excel("faces3.xlsx")

#Categorias
#El estado de animo del sujeto es
#(enojado=1,feliz=2,neutral=3,triste=4)
emocion=factor(data$label)

#Visualización de sujetos
n_0=length(emocion)#número total de fotos
n=sample(1:n_0,1,replace=FALSE)

#En cara_n se observa la concatenación
#de la foto n
cara_n= matrix(as.numeric(data[n,-1]),
       nrow =60, byrow = TRUE)
#image(cara_n, col = grey.colors(255))
#Rotación de la imagen
rotar=function(x) t(apply(x, 2, rev))
image(rotar(cara_n),col=grey.colors(255))


#Vector aleatorio de longitud
muestra=sample(1:n_0,6,replace=FALSE)
#visualización de 6 sujetos
par(mfrow=c(2,3))
lapply(muestra,
function(x) image(rotar(matrix(
            unlist(data[x,-1]),
            nrow = 60,
            byrow = TRUE)),
            col=grey.colors(255),
            xlab=data[x,1]
   )
)

#Datos de entrenamiento
#set.seed(78)
muestra01=sample(n_0,n_0*0.7)
train=data[muestra01,]

emocion_train=factor(train$label)

#Datos de prueba
test=data[-muestra01,]

emocion_test=factor(test$label)

##########################################
#Una red neuronal de una capa oculta con una
#1 neurona oculta función de
#activación logistica

mod_log_1=nnet(emocion_train~.,data=train,
              size=1, maxit=1.0e+9,
              MaxNWts=1.0e+9,decay=0.001,
              rang=0.05,na.action = na.omit)

pred_log_1=predict(mod_log_1,newdata=test,
                   type = "class" )

#tabla de confusión
tabla_log_1=table(emocion_test, pred_log_1,
                dnn=c("Reales", "Predichos"))
tabla_log_1

#Proporción de error en la tabla de
#confusión tabla_log_1

n1_log=length(tabla_log_1[1,])
sumacol_log1=c()
maxcol_log1=c()
for(i in 1:n1_log)
{
sumacol_log1[i]=sum(tabla_log_1[,i])
maxcol_log1[i]=max(tabla_log_1[,i])
}
sumtotal_log1=sum(sumacol_log1)
error_x_digito_log_1=sumacol_log1 - maxcol_log1
proporcion_error_x_digito_log1=
          error_x_digito_log_1/sumtotal_log1
error_log1=sum(proporcion_error_x_digito_log1)
error_log1

##Prediccion de la red neuronal para logistic 1
num_aleatorio=sample(1:sumtotal_log1,1,replace=F)
num_selec=c("El estado de animo del sujeto es=>",
           test[num_aleatorio,1])
num_selec
# Visualización del dígito
sample_aleatorio= matrix(as.numeric(
                  test[num_aleatorio,-1]),
                  nrow = 60, byrow = TRUE)
rotar=function(x) t(apply(x, 2, rev))
image(rotar(sample_aleatorio),
      col = grey.colors(255))

prediccion_logistc_1=predict(mod_log_1,
                newdata=test[num_aleatorio,],
                type = "class")
pre_logistic1=c("La predicción de la red neuronal es:",
               prediccion_logistc_1)
pre_logistic1
```

Bibliografía

[1] E. Alpaydin. *Introduction to Machine Learning.* Adaptive Computation and Machine Learning series. MIT Press, 2014.

[2] C. Bishop. *Pattern Recognition and Machine Learning.* Information Science and Statistics. Springer, 2006.

[3] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition.* Stochastic Modelling and Applied Probability. Springer New York, 2013.

[4] J. García, J. Herrero, A. de Jesús, J. López, M. Á. Guisado, Á. Bustamante, and W. Padilla. *Ciencia de datos: técnicas analíticas y aprendizaje estadístico.* Altaria, 2018.

[5] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer series in statistics. Springer, 2009.

[6] F. Hernández A. *Cálculo de probabilidades.* Sociedad Matemática Mexicana, 2003.

[7] A. Izenman. *Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning.* Springer Texts in Statistics. Springer New York, 2013.

[8] X. Li and R. Xu. *High-Dimensional Data Analysis in Cancer Research.* Applied Bioinformatics and Biostatistics in Cancer Research. Springer New York, 2008.

[9] T. Mitchell. *Machine Learning.* McGraw-Hill international editions - computer science series. McGraw-Hill Education, 1997.

[10] D. R. Pashine, S. and R. Kushwah. Handwritten digit recognition using machine and deep learning algorithms. *International Journal of Computer Applications*, 176(42), 2020.

[11] A. Rencher. *Multivariate Statistical Inference and Applications.* A Wiley Interscience publication. Wiley, 1998.

[12] B. Ripley and W. Venables. Package 'nnet'. Manual disponible en https://cran.r-project.org/web/packages/nnet/nnet.pdf, May 2021.

[13] M. A. Rodríguez Flores. *Desarrollo de un sistema basado en un clasificador neuronal para reconocimientos de orugas*. PhD thesis, Universidad Nacional Autónoma de México, Ciudad de México, Abril 2010.

[14] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[15] H. Royden and P. Fitzpatrick. *Real Analysis*. Prentice Hall, 2010.