Universidad del País Vasco Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA
Konputazio Ingeniaritza eta Sistema Adimentsuak Unibertsitate Masterra
Máster Universitario en Ingeniería Computacional y Sistemas Inteligentes

INGENIERÍA COMPUTACIONAL Y SISTEMAS INTELIGENTES

EXPLORACIÓN Y ANÁLISIS DE DATOS

Entrega 4

Estudiante: Eneko Perez **Estudiante:** Alan García

Curso: 2024-2025 **Fecha:** 23 de octubre de 2024

Índice

1. Laboratorio 1

1.1. Ejercicio 1

En este ejercicio se propone realizar una comparación entre el error experimental y teórico de un clasificador de Bayes. Se supone una configuración inicial con $\mu_0 = 0$, $\mu_1 = 1$ y $\sigma_0 = \sigma_1 = 1$, con lo que el problema descrito se corresponde con la Figura 1. En ella se representan las distribuciones de probabilidad cuando $c = 0$ y cuando $c = 1$. De esta manera, se puede observar que un clasificador Bayesiano producirá una clasificación errónea en el caso de que por ejemplo la instancia $x = 1,5$ pertenezca realmente a la clase $c = 0$, ya que, debido a que la densidad de probabilidad de $F2$ es mayor que la de $F1$ en ese punto, el clasificador de Bayes devolverá $c = 1$. Es por ello que el error de clasificación para $c = 0$ es el área resaltada de la Figura 1 y que debido a la simetría del problema es el mismo error para $c = 1$.

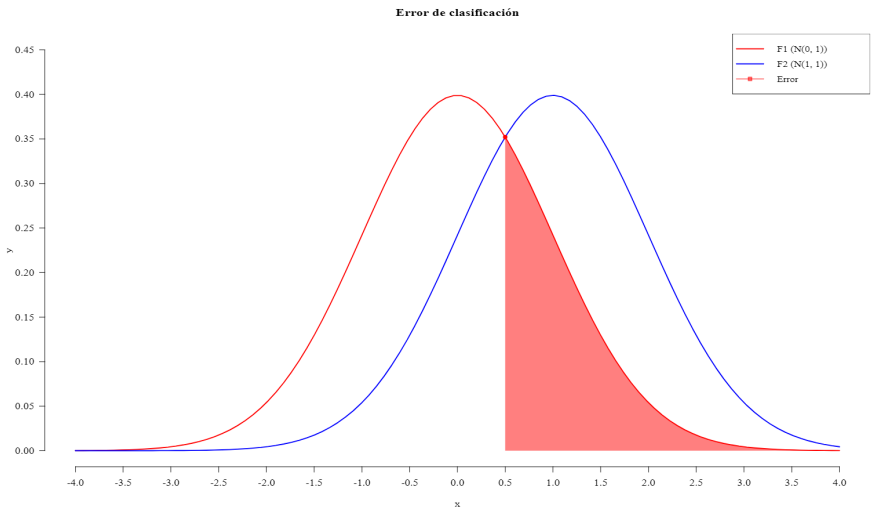

Figura 1: Primera configuración del problema

Para realizar el cálculo del error experimental se han dispuesto de varias muestras de creciente tamaño $n = 1000, 1100, 1200, ..., 10000$, se ha implementado un clasificador Bayesiano y se ha calculado el error de clasificación obteniendo la siguiente gráfica:

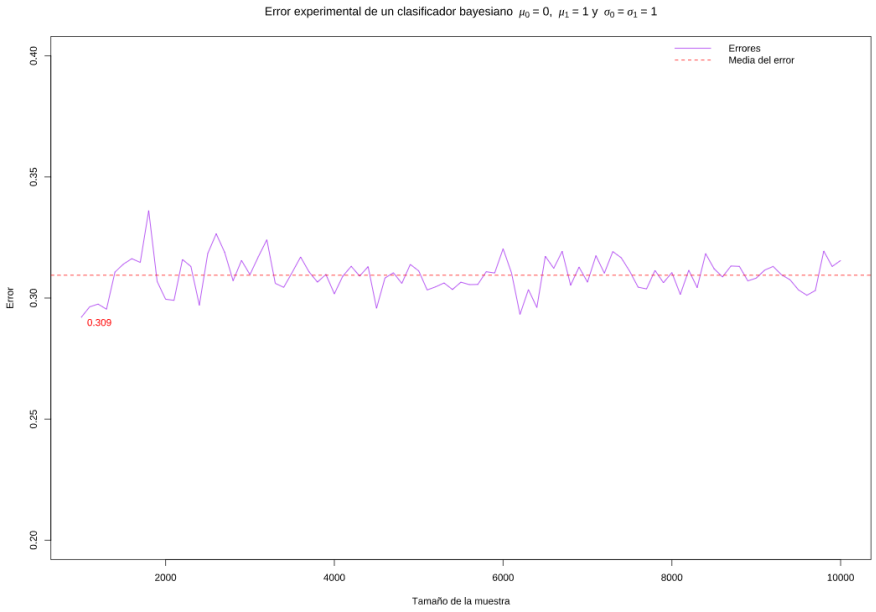

Figura 2: Error experimental del clasificador de Bayes. Error experimental medio = 0.30855

Como se puede apreciar, el error parece mantenerse acotado en torno a 0,309. Sin embargo, este es solo el error experimental. Como se ha explicado anteriormente, el error teórico que comete el clasificador Bayesiano en este problema cuando la clase real es $c = 0$ se corresponde con el área bajo la curva desde el punto de intersección de las dos funciones de densidad hasta ∞ y que, debido a la simetría del problema, es el mismo error para $c = 1$.

La función de densidad viene dada por:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Luego,

$$f_0(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right); f_1(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-1)^2}{2}\right)$$

Por lo tanto, para hallar el punto de corte, igualamos $f_0(x) = f_1(x)$:

$$\begin{aligned} f_0(x) &= f_1(x) \\ \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) &= \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-1)^2}{2}\right) \\ -\frac{x^2}{2} &= -\frac{(x-1)^2}{2} \\ -2x + 1 &= 0 \\ x &= \frac{1}{2} \end{aligned}$$

Con esto podemos calcular finalmente el error teórico:

$$\int_{\frac{1}{2}}^{+\infty} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) dx = 0,3085$$

Si representamos el error teórico y el experimental obtenemos una gráfica bastante similar a la anterior:

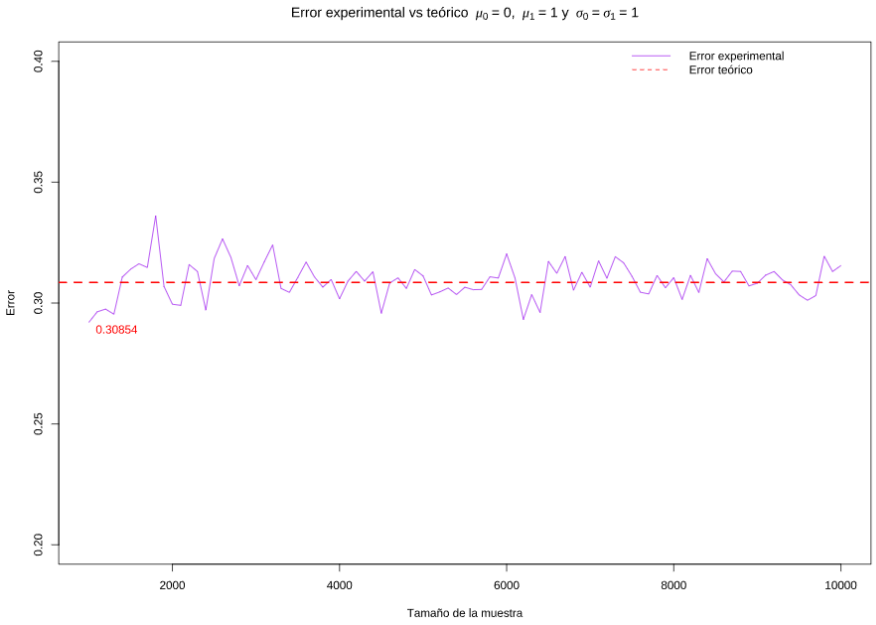

Figura 3: Error experimental y teórico del clasificador de Bayes

Sin embargo, cuando se adopta una configuración en la cual las desviaciones típicas difieren ya no se puede considerar que los errores para las clases $c = 1$ y $c = 0$

son iguales. Para ilustrar esto se ha realizado el mismo experimento considerando $\mu_0 = 0$, $\mu_1 = 1$, $\sigma_0 = 1$ y $\sigma_1 = 2$. Así, se ha comenzado calculando el punto de corte de las dos funciones:

$$
\begin{aligned}
f_0(x) &= f_1(x) \\
\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) &= \frac{1}{2\sqrt{2\pi}} \exp\left(-\frac{(x-1)^2}{8}\right) \\
2\exp\left(-\frac{x^2}{2}\right) &= \exp\left(-\frac{(x-1)^2}{8}\right) \\
\ln(2) - \frac{x^2}{2} &= -\frac{(x-1)^2}{8} \\
\ln(2) - \frac{x^2}{2} &= -\frac{x^2 - 2x + 1}{8} \\
-8\ln(2) + 4x^2 &= x^2 - 2x + 1 \\
3x^2 + 2x - (1 + 8\ln(2)) = 0 \quad x_1 = -1{,}847545 \quad x2 &= 1{,}180878
\end{aligned}
$$

Ilustrando de nuevo el problema se ha obtenido la Figura 4, con la que podemos llegar a la conclusión de que $E(c_{\text{pred}} = 1 \mid c_{\text{real}} = 0) \neq E(c_{\text{pred}} = 0 \mid c_{\text{real}} = 1)$.

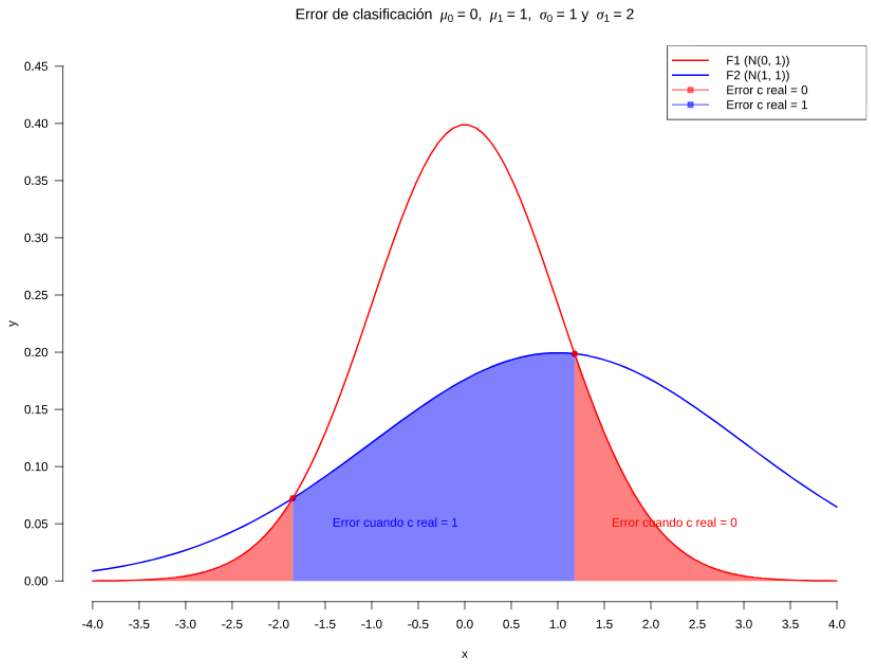

Figura 4: Segunda configuración del problema

Por ello, conviene expresar el error que comete el clasificador de Bayes de forma general:

$$E = P(F_1) \cdot P(\text{error} \mid F_1) + P(F_2) \cdot P(\text{error} \mid F_2)$$

con lo que el error teórico de la nueva configuración sería:

$$E = \frac{1}{2} \cdot \left(1 - \int_{x_1}^{x_2} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) dx\right) + \frac{1}{2} \cdot \int_{x_1}^{x_2} \frac{1}{2\sqrt{2\pi}} \exp\left(-\frac{(x-1)^2}{8}\right) dx = 0{,}304967$$

Finalmente, se ha realizado la comparación entre el error teórico y experimental considerando $\mu_0 = 0$, $\mu_1 = 1$, $\sigma_0 = 1$ y $\sigma_1 = 2$:

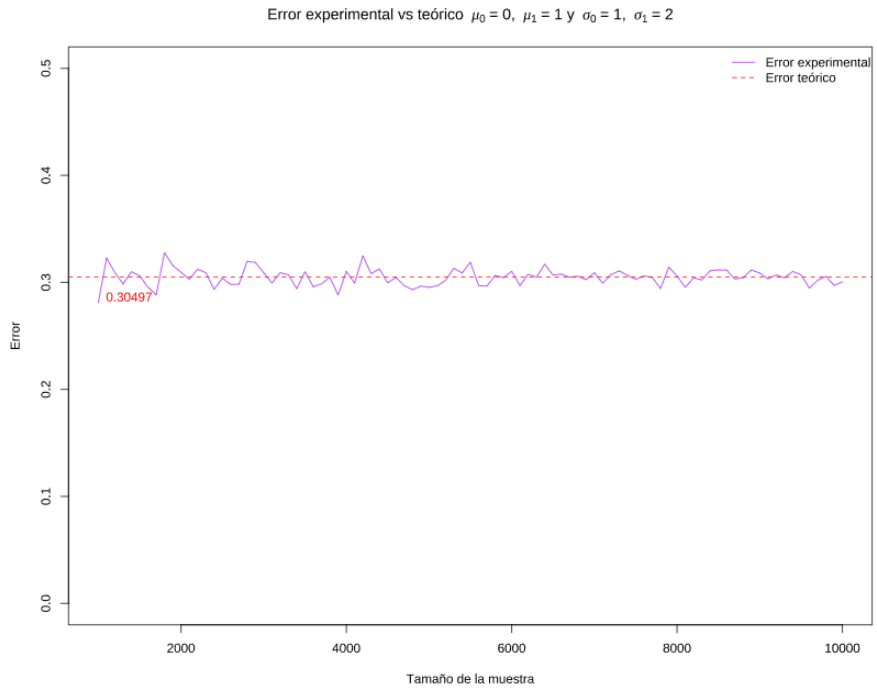

Figura 5: Error experimental y teórico de la segunda configuración. Error experimental medio = 0.30485

Finalmente, se puede considerar que es posible extender este análisis a $X \in R^d$. Se utilizaría una distribución normal multivariante, también llamada distribución gaussiana multivariante, que es una generalización de la distribución normal unidimensional a dimensiones superiores.

Las distribuciones condicionales para cada clase serían gaussianas multivariante $\mathcal{N}(\mu_0, \Sigma_0)$ y $\mathcal{N}(\mu_1, \Sigma_1)$, con $X \in R^d$. Donde Σ_0 y Σ_1 van a ser matrices de co-

varianza que determinarán la forma de las distribuciones en el espacio de alta dimensionalidad.

1.2. Ejercicio 2

Este ejercicio tiene un objetivo dual. En primer lugar, trata de demostrar que un regresor por mínimos cuadrados no siempre obtiene una solución óptima. Para ello, se ha propuesto la creación de un conjunto de entrenamiento y test compuestos por 100 y 1000 instancias respectivamente donde $X \in R^{10}$ y el resultado del regresor $Y \in R$ viene dado por la siguiente expresión:

$$y = \mathbf{w}^T\mathbf{x} + \epsilon \quad \text{donde} \quad \epsilon \sim N(0, 1).$$

Es decir, los conjuntos de datos presentan un ruido intrínseco que comprometerá el desempeño del regresor. En segundo lugar, se propone realizar la implementación de un regresor por mínimos cuadrados penalizados para comparar el desempeño de ambos y, finalmente, realizar una búsqueda del λ óptimo para el problema.

De esta manera, se han generado los conjuntos de datos de train y test con la estructura mostrada en la Tabla 1 donde las 5 primeras features de cada instancia se encuentran en el intervalo $[0, 1]$ y son muestreadas de una distribución uniforme; mientras que las 5 siguientes features son muestreadas también de manera uniforme en el intervalo $[100, 1000]$.

	x_1	x_2	x_3	$\cdots$	x_{10}	y
1	0,35	0,74	0,24	$\cdots$	335,28	1364,15
2	0,46	0,53	0,59	$\cdots$	307,64	973,95
3	0,86	0,70	0,88	$\cdots$	434,14	1547,51
4	0,04	0,94	0,38	$\cdots$	207,99	1182,16
5	0,02	0,02	0,20	$\cdots$	269,75	706,37
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
100	0,08	0,14	0,85	$\cdots$	715,60	989,12

Tabla 1: Muestra del conjunto de datos (conjunto train)

Con estos datos se ha entrenado un regresor por mínimos cuadrados y se ha realizado la búsqueda del λ para el regresor por mínimos cuadrados penalizado. Graficando los errores de este último se ha obtenido la siguiente gráfica:

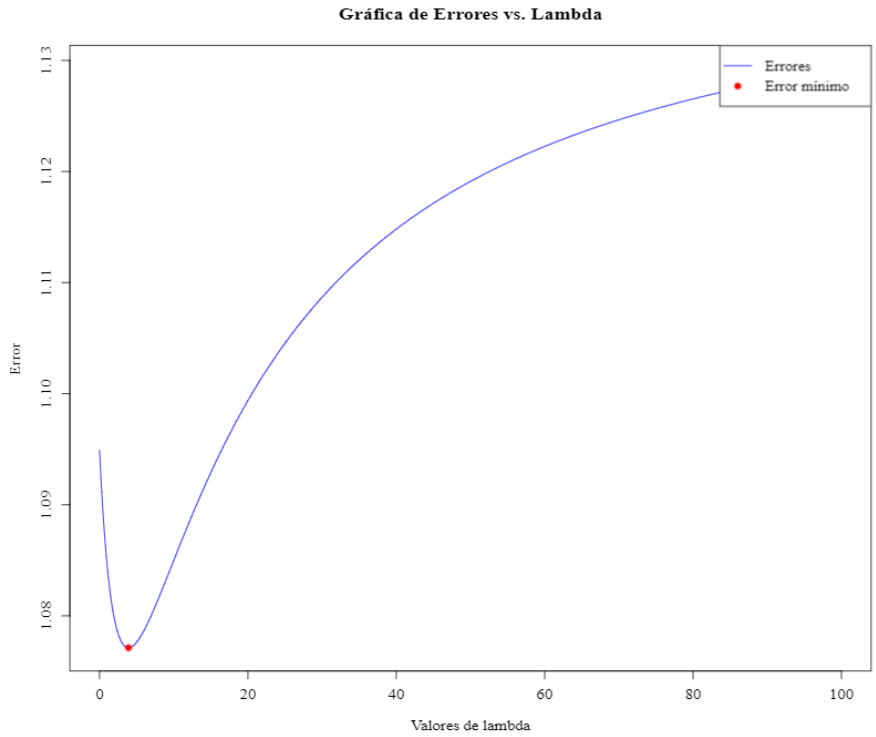

Figura 6: Errores en función de landa en un regresor penalizado

Con ello, obtenemos los siguientes pesos y errores para ambos regresores:

Índice	w_1	w_2
1	0.865	0.570
2	-0.170	-0.110
3	-0.247	-0.122
4	0.348	0.244
5	0.603	0.390
6	0.952	0.952
7	0.246	0.246
8	0.054	0.054
9	0.579	0.579
10	0.448	0.448

Tabla 2: Valores de los vectores w_1 y w_2

Regresor	**Error**
Mínimos Cuadrados (w_1)	$8{,}00 \times 10^{-6}$
Penalizado (w_2, $\lambda = 3{,}9$)	1.077

Tabla 3: Errores de los regresores por mínimos cuadrados y penalizados.

Con estos resultados se pueden extraer algunas conclusiones. En primer lugar, se

puede apreciar que los pesos de tanto el regresor por mínimos cuadrados (w_1) como el regresor por mínimos cuadrados penalizados con $\lambda = 3{,}9$ óptima (w_2) son pequeños, lo que implica que los regresores pueden ser relativamente invariables ante pequeños cambios en el conjunto de entrenamiento. Sin embargo, al prestar atención al error de los modelos sobre el conjunto de entrenamiento es evidente que el regresor sin penalizaciones presenta un overfitting muy severo ya que el error es muy cercano a 0 y, por lo tanto, está pasando muy cerca de los datos de entrenamiento.

Para comprobar esta hipótesis se ha introducido inestabilidad en los datos multiplicando los tres autovalores menos significativos de los datos de entrenamiento por 10^{-2} para reconstruir de nuevo los datos desencadenando un posible mayor protagonismo del ruido. Con esta transformación de datos obtenemos los siguientes vectores de pesos de los regresores:

Índice	w_1	w_2
1	-0.020	0.366
2	**6.830**	0.121
3	0.579	0.056
4	**7.586**	0.453
5	**-12.113**	0.270
6	0.950	0.952
7	0.246	0.246
8	0.054	0.054
9	0.580	0.580
10	0.447	0.448

Tabla 4: Valores de los vectores w_1 y w_2 después de introducir inestabilidad

Como se puede comprobar, los valores de w_1 han variado considerablemente y presenta pesos muy altos. Este hecho unido a que el error de este regresor es 0,00084, confirma la sospecha del overfitting que está realizando. Por el contrario, el regresor penalizado sigue mostrando una tolerancia mayor al ruido y al overfitting, ya que sigue manteniendo unos pesos pequeños y el $error = 1{,}077$ y $\lambda = 3{,}9$ son iguales que antes de introducir inestabilidad en los datos. Con todo ello, se puede concluir que el regresor por mínimos cuadrados penalizados presenta una mayor capacidad de generalización.

2. Laboratorio 2

2.1. Ejercicio 3

Ahora se presenta un nuevo problema de regresión en R^2 definido por la siguiente función:

$$y(x_1, x_2) = 3x_1^2 - 2x_1x_2 + 2x_2^2 + \varepsilon, \quad \text{donde } \varepsilon \sim N(0, 1)$$

El objetivo del ejercicio es el de encontrar un regresor lineal que pase por todos los puntos de una muestra. Para generar el conjunto de datos, se propone realizar un muestreo de una distribución uniforme $[0, 1]$ para las variables x_1 y x_2, siendo y determinada por la función descrita anteriormente. En un primer momento, se ha inicializado un conjunto de datos con $n = 10$ elementos:

	x_1	x_2	y
1	0.734	0.180	1.939
2	0.574	0.530	1.939
3	0.238	0.507	1.939
4	0.112	0.283	1.939
5	0.819	0.672	1.939
6	0.926	0.482	1.939
7	0.263	0.610	1.939
8	0.829	0.643	1.939
9	0.927	0.523	1.939
10	0.202	0.347	1.939

Tabla 5: Conjunto de datos redondeados a tres decimales

Sin embargo, al entrenar un regresor con estos datos se obtiene un error cuadrático medio de 0,069 que no es muy cercano a 0. Esto nos indica que la recta resultado del regresor no está pasando por todos los puntos, razón por la cual se comete error. Además, hay que tener en cuenta que la dimensión del conjunto de datos es $n \times d$, donde en este caso $n = 10$ y $d = 2$. Por ello, se proponen varias funciones Φ para transformar los datos en una dimensión R^p en la que exista una recta que pase exactamente por los 10 puntos de la muestra considerando distintas dimensiones:

$$\Phi_1(x_1, x_2) = \{x_1, x_2, x_1^2, x_2^2, x_1x_2\}$$
$$\Phi_2(x_1, x_2) = \{x_1, x_2, x_1^2, x_2^2, x_1x_2, x_1^3, x_2^3, x_1^2x_2, x_1x_2^2\}$$
$$\Phi_3(x_1, x_2) = \{x_1, x_2, x_1^2, x_2^2, x_1x_2, x_1^3, x_2^3, x_1^2x_2, x_1x_2^2, x_1^4, x_2^4, x_1^3x_2, x_1x_2^3, x_1^2x_2^2\}$$
$$\Phi_4(x_1, x_2) = \{x_1, x_2, x_1^2, x_2^2, x_1^3, x_2^3, x_1^4, x_2^4, x_1x_2, \sqrt{x_1}\}$$

Los errores obtenidos para cada transformación se resumen en la siguiente Tabla:

Transformación	Dimensión p	Error
Ninguna	2	0.0693153
Φ_1	5	0.4507991
Φ_2	9	0.0081557
Φ_3	14	1.644282×10^{-30}
Φ_4	10	8.110156×10^{-22}

Tabla 6: Errores para cada transformación y dimensiones del espacio

Como se puede observar, cuando la dimensión del espacio resultante después de haber aplicado la función de transformación es $p = 10$, el error es prácticamente 0 indicando que el regresor pasa por todos los puntos de la muestra. Por ello, se puede concluir que el número mínimo de dimensión d para que exista una recta que pase por todos los puntos de la muestra es n o el número de elementos de la muestra. De esta forma se consigue un sistema compatible determinado siempre y cuando todos los features de X transformado sean linealmente independientes o, lo que es lo mismo, siempre que las transformaciones aplicadas a los datos no sean combinaciones lineales de las variables iniciales.

Por esta razón, aplicar transformaciones de los datos es útil para mejorar el desempeño del regresor siempre y cuando el dataset sea de pocos elementos. En el momento que n empieza a aumentar de tamaño realizar una transformación de datos implica la creación de una matriz $n \times n$ y, por lo tanto, multiplicaciones de matrices muy grandes. Es aquí donde reside la potencia del Kernel Trick ya que permite calcular el producto interno de los datos en un espacio transformado sin necesidad de realizar la transformación explícita.

d	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}
2	0.61	0.72												
5	-0.68	-0.66	0.68	-5.56	0.22									
9	-1.61	-9.62	-7.03	-1.20	-3.18	-6.45	-0.93	-3.10	-0.53					
14	-0.14	-1.15	-0.14	-0.01	-0.65	-1.83	-2.41	-1.84	-0.89	5.87	-0.76	0.53	1.40	-0.84
10	-1.11	2.80	2.44	-7.30	-3.04	8.20	1.38	-3.26	-0.28	4.08				

Tabla 7: Tabla de pesos para diferentes dimensiones

Los pesos de los regresores que se reflejan en la Tabla 7 muestran valores altos incluso en $d = 5$, lo que demuestra un claro overfitting. Sin embargo, es complicado afirmar si el overfitting es mayor conforme la dimensión aumenta. Por esta razón, se propone aumentar el tamaño de la muestra a $n = 20$. Esto supone también calcular una transformación más compleja.

Con el fin de poder establecer una manera de transformar los datos para cualquier dimensión, se ha realizado la implementación de una función que dada la dimensión calcula las p *Random Fourier Features* consiguiendo p features linealmente independientes.

Tras repetir el experimento con las *Random Fourier Features* para $n = 20$ se ha obtenido un resultado similar: cuando $d = n$ el error del regresor es prácticamente 0.

Finalmente, se ha dispuesto un nuevo experimento en el que se cuenta con un conjunto de entrenamiento de $n = 20$ y otro conjunto de datos de evaluación con 1000 instancias. El objetivo es el de evaluar el comportamiento de un regresor entrenado para un numero variable de dimensiones frente a unos datos para los que no se ha ajustado previamente. Los resultados que se han conseguido se resumen en la siguiente Figura:

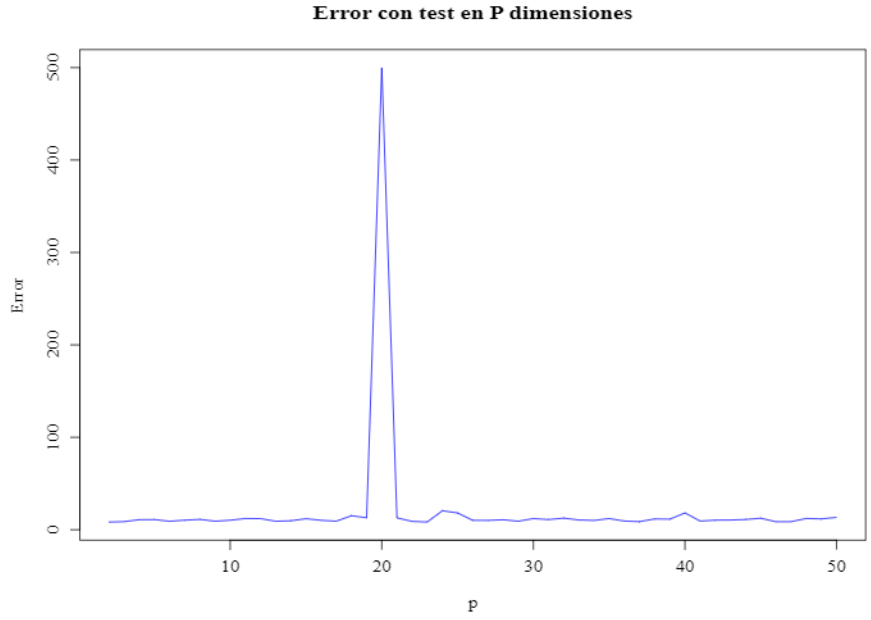

Figura 7: Error del regresor frente al conjunto de test

Cuando la dimensión es pequeña, el modelo no se puede sobre-ajustar tanto a los datos. Es por ello, que cuando la dimensión aumenta y $d = 20$ se produce un modelo que pasa por todos y cada uno de los puntos de entrenamiento, demostrando el pico máximo de error frente a los datos de test (ya que el overfitting es tan grande que el modelo no es capaz de generalizar nada). Sin embargo, con dimensiones superiores a 20, el error sigue siendo muy alto al seguir existiendo overfitting, pero el modelo no llega a pasar exactamente por todos los puntos de entrenamiento, lo que ocasiona un modelo algo más generalista que cuando $n = d$. Si nos fijamos en la Tabla 8, para $p = 20$, w presenta valores muy grandes que indican un overfitting máximo. Sin embargo, para el resto de los casos el overfiting no es tan severo.

Dimensión	**Pesos (w)**
10	(12,84, 63,25, 4,29, −70,02, −3,65, −15,03, 1,78, 1,78, 8,22, −3,82)
15	(−0,55, 2,62, 0,93, 0,66, −0,33, −3,11, 0,38, −26,47, −1,43, −0,25, . . .)
20	(**230.65**, **81940.86**, **7677.28**, **-108.88**, **287.99**, **-83784.64**, **-18604.00**, **150791.30**, **6717.20**, **88727.40**, . . .)
25	(1,73, −0,27, 2,45, 0,80, 1,26, 0,53, 0,30, −0,58, 0,38, −0,46, −1,62, 1,52, 1,24, 0,07, −0,33, 1,13, . . .)
30	(3,55, 0,12, −1,69, 0,91, 1,68, 4,77, −0,12, −1,56, −1,44, −1,69, −4,16, −3,00, −4,54, −0,99, 3,11, . . .)

Tabla 8: Pesos en función de las dimensiones

Cabe destacar que estos resultados dependen en gran medida de las transformaciones de los datos, por lo que en otras configuraciones se pueden encontrar valores de w bastante altos para otras dimensiones distintas a 20. Sin embargo, el máximo error con el conjunto de test se alcanza cuando $n = d$

3. Laboratorio 3

3.1. Ejercicio 4

Para el desarrollo de este ejercicio se facilita una muestra de datos $S_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ donde $x \in R^d$ e $y \in \{-1, 1\}$. Concretamente, la muestra cuenta con $n = 10$ instancias y x tiene dimensión $d = 2$. El objetivo de este ejercicio es realizar la implementación de clasificadores binarios y evaluar su comportamiento frente a los datos proporcionados.

3.1.1. Parte 1

Al representar los datos proporcionados obtenemos la gráfica de la siguiente Figura:

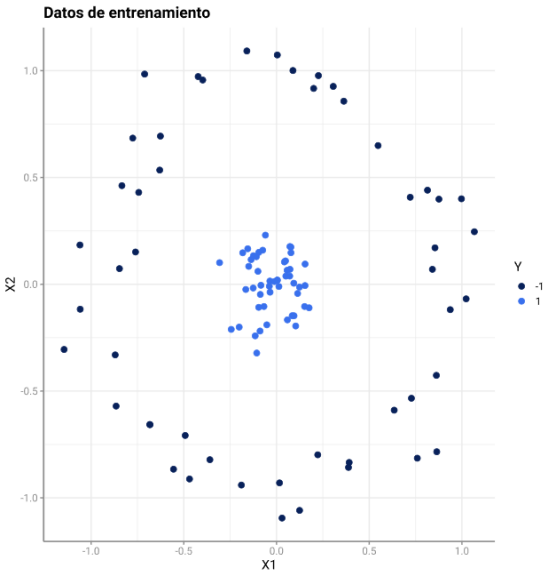

Figura 8: Datos de entrenamiento

Se puede comprobar a simple vista que los datos no son linealmente separables y, por lo tanto, no existe ningún clasificador lineal que sea capaz de separarlos en este espacio. Sin embargo, para demostrar empíricamente esta afirmación se ha realizado la implementación de un clasificador lineal empleando *log loss*, descenso del gradiente, considerando $\lambda = 1$, *Learning Rate* $\alpha = 0{,}1$ y calculando los pesos iniciales de forma aleatoria a partir de una distribución uniforme $w_0 \sim \text{unif}([-0{,}5, 0{,}5]^2)$. Con todo ello, se ha conseguido la siguiente evolución del error por cada una de las iteraciones:

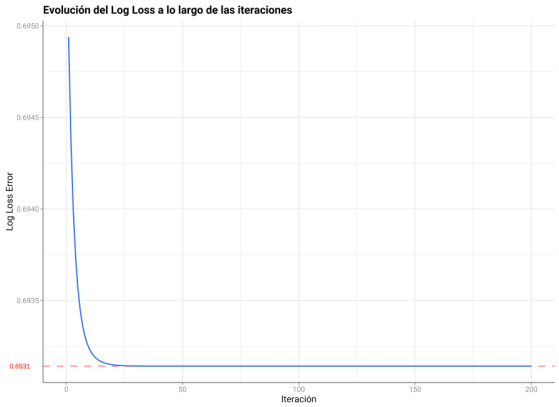

Figura 9: Evolución del log loss

De esta manera, el error del clasificador converge con un *log loss error* de 0,6931 y un *accuracy* de 0,52 resultando en los pesos $w = (-0{,}00170, -0{,}00018)$. Se puede comprobar que los pesos y el *accuracy* son muy bajos, demostrando que el clasificador no se ha adaptado a penas al conjunto de datos y teniendo una tasa de acierto del 50 % (como si fuese un clasificador aleatorio con una probabilidad de 0,5). Al graficar el clasificador se obtiene la siguiente Figura:

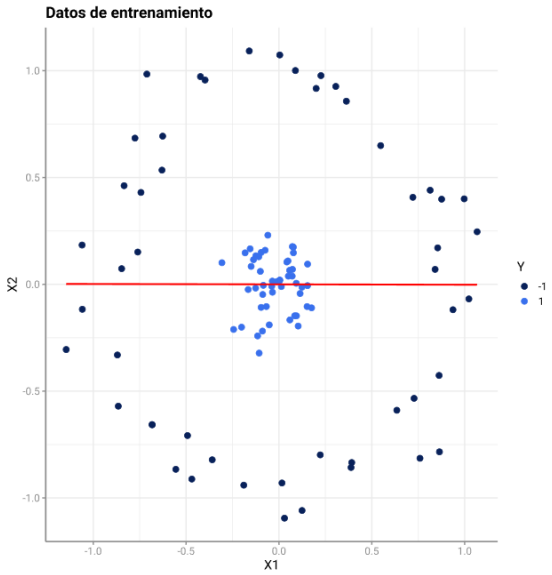

Figura 10: Representación del clasificador (rojo) sobre los datos

Con estos datos se puede comprobar que la hipótesis inicial a cerca de que un clasificador lineal no sería capaz de separar los datos es correcta.

3.1.2. Parte 2

En esta segunda parte se propone transformar los datos con el fin de que sean separables aumentando su dimensión. En concreto, se propone la siguiente función de transformación:

$$\Phi_4(x_1, x_2) = \left(x_1, x_2, x_1^2 + x_2^2 - \frac{1}{2}\right)$$

Al aplicarla sobre la muestra de datos, conseguimos que sean linealmente separables en R^3:

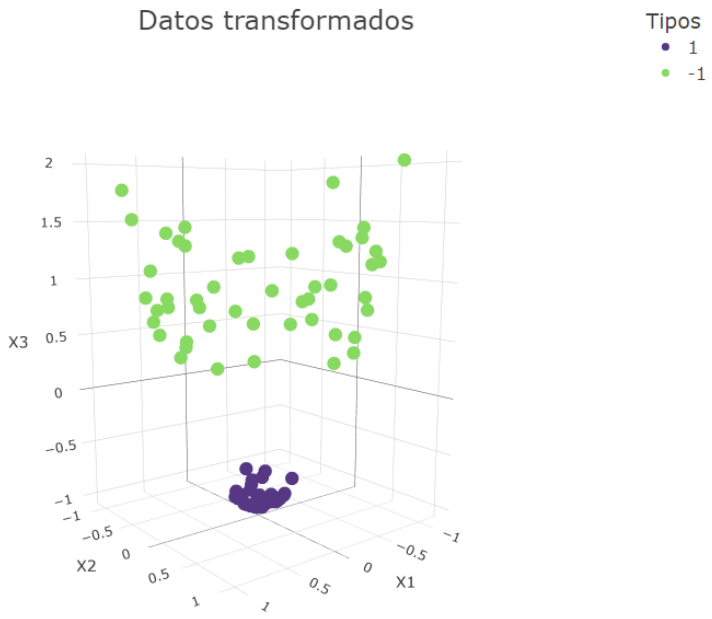

Figura 11: Representación de los datos transformados

Finalmente, se ha realizado el mismo proceso del apartado anterior considerando un clasificador lineal con *log loss* como función de pérdida, descenso del gradiente y $w_0 \sim \text{unif}([-0{,}5, 0{,}5]^3)$ como vector de pesos inicial.

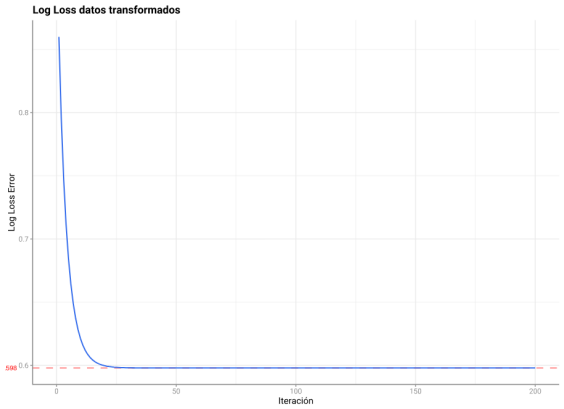

Figura 12: Evolución del log loss sobre los datos transformados

En la Figura 12 se muestra la evolución del error, consiguiendo un *log loss error* final de 0,5980, un *accuracy* de 1 y unos pesos $w = (-1{,}91\times10^{-3}, 1{,}47\times10^{-5}, -0{,}213)$. Este clasificador es capaz de separar todos los datos y consigue mantener unos pesos con valores bajos, mostrando poco overfitting. Al graficar el plano del clasificador se obtiene la siguiente Figura:

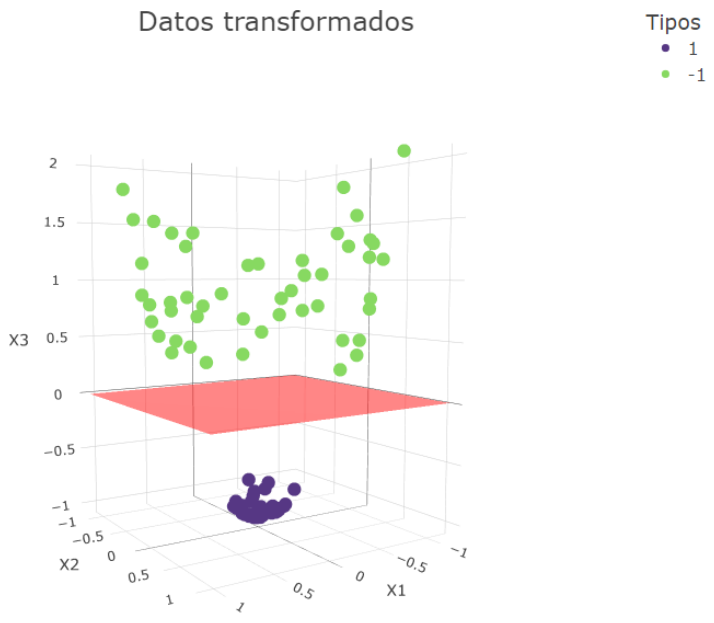

Figura 13: Representación del clasificador (rojo) sobre los datos transformados

3.1.3. Parte 3

En este ultimo apartado se pide realizar una comparación entre los resultados obtenidos con el clasificador lineal implementado anteriormente, en el que se itera actualizando los pesos w, con una nueva implementación del mismo clasificador de manera que se actualicen los multiplicadores c del teorema de representación. El cálculo de los pesos finales y el proceso de actualización de los multiplicadores para la iteración t viene dado por la siguiente expresión:

$$w_t = \sum_{i=1}^{n} x_i c_i^t \quad \text{donde } c_i^{t+1} = c_i^t - \alpha \left(\frac{1}{n} \left(\frac{-y_i}{1 + e^{y_i f_t(x_i)}} \right) + 2\lambda c_t^i \right), \quad i = 1, \ldots, n$$

Una vez realizada la implementación, se ha llevado a cabo un experimento con el fin de comparar el algoritmo iterando sobre w y sobre c. Es por ello que se ha inicializado $c_0 \sim \text{unif}([0,1]^n)$ y se ha calculado el peso $w_0 = \sum_{i=1}^{n} x_i c_i^0$ para comparar los resultados desde un punto de inicio común. Los resultados se recogen en la siguiente tabla:

	Iterando sobre w		**Iterando sobre** c	
t	*Log Loss*	w	*Log Loss*	w
0	0.753	$(-2{,}338, 0{,}253, -0{,}019)$	0.753	$(-2{,}338, 0{,}253, -0{,}019)$
1	0.705	$(-1{,}860, 0{,}201, -0{,}063)$	0.705	$(-1{,}860, 0{,}201, -0{,}063)$
2	0.672	$(-1{,}479, 0{,}160, -0{,}097)$	0.672	$(-1{,}479, 0{,}160, -0{,}097)$
3	0.650	$(-1{,}176, 0{,}127, -0{,}123)$	0.650	$(-1{,}176, 0{,}127, -0{,}123)$
4	0.634	$(-0{,}934, 0{,}100, -0{,}144)$	0.634	$(-0{,}934, 0{,}100, -0{,}144)$
⋮	⋮	⋮	⋮	⋮
200	0.598	$(-0{,}002, 0{,}000, -0{,}212)$	0.598	$(-0{,}002, 0{,}000, -0{,}212)$

Tabla 9: Comparación entre iteraciones sobre w y sobre c

Como se puede observar, los dos métodos presentan los mismos resultados en todas y cada una de las iteraciones, demostrando así de forma experimental que ambos métodos son equivalentes y que conducen al mismo resultado presentado en el apartado anterior: $w = (-1{,}91 \times 10^{-3}, 1{,}47 \times 10^{-5}, -0{,}213)$.

x_1	x_2	x_3	x_4	x_5	x_6	x_7	...	x_{10}
-2.30 $\times 10^{-3}$	2.25 $\times 10^{-3}$	-2.16 $\times 10^{-3}$	-2.31 $\times 10^{-3}$	-2.20 $\times 10^{-3}$	-2.15 $\times 10^{-3}$	-2.27 $\times 10^{-3}$	...	-2.30 $\times 10^{-3}$

Tabla 10: Valores de los multiplicadores

Finalmente, se pide solucionar el problema de clasificación para los datos transformados empleando *Hinge Loss* como función de pérdida y añadiendo el *Bias term* b. De esta manera, el problema de optimización quedaría escrito de la siguiente manera con $l(w, b)$ como la función *Hinge Loss*:

$$(\mathbf{w}^*, \mathbf{b}^*) = \arg\min_{w,b} \frac{1}{n} \sum_{i=1}^{n} l(w, b) + \lambda \|w\|^2 \quad \text{donde } l(w, b) = |1 - y_i(w^T x_i + b)|$$

Debido a que $l(w, b)$ no es derivable cuando $1 - y_i(w^T x_i + b) = 0$, hay que aplicar el método del subgradiente para poder encontrar una solución al problema de optimización. De esta manera, w y b se expresarían de la siguiente forma para cada iteración $t + 1$:

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{t}} \left(\frac{1}{n} \sum_{i=1}^{n} S_w(w_t, b_t) + 2\lambda w_t \right)$$

$$b_{t+1} = b_t - \frac{\alpha}{\sqrt{t}} \left(\frac{1}{n} \sum_{i=1}^{n} S_b(w_t, b_t) \right)$$

donde

$$S_w(w_t, b_t) = \frac{\partial l(w, b)}{\partial w} = \begin{cases} -y_i x_i & \text{si } y_i(w^T x_i + b) \leq 1 \\ 0 & \text{si } y_i(w^T x_i + b) > 1 \end{cases}$$

$$S_b(w_t, b_t) = \frac{\partial l(w, b)}{\partial b} = \begin{cases} -y_i & \text{si } y_i(w^T x_i + b) \leq 1 \\ 0 & \text{si } y_i(w^T x_i + b) > 1 \end{cases}$$

El resultado al aplicar *Hinge Loss* como función de pérdida, utilizando los hiperparámetros $\lambda = 1$ y $\alpha = 0{,}01$ y estableciendo los pesos iniciales como $w_0 = (-2{,}338, 0{,}253, -0{,}019)$, se refleja en la Figura 14. El clasificador converge en $w = (-0{,}795, -0{,}075, -2{,}518)$ y al introducir la actualización del *bias* se obtiene un valor final de $b = -1{,}05$.

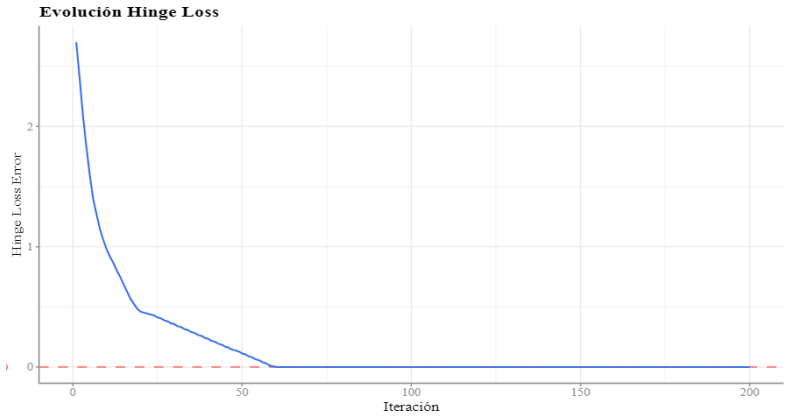

Figura 14: Evolución del hinge loss sobre los datos transformados

Al representar estos resultados sobre los datos se obtiene la gráfica de la Figura 15. Como se puede comprobar, el valor *bias* desplaza el plano de forma que el margen aumente y, en comparación a los resultados empleando *Log Loss*, la inclinación del plano dada por los pesos w crea un margen mayor entre las dos clases de los datos. En general, se obtiene un clasificador con un margen mayor haciéndolo algo más robusto frente a valores atípicos.

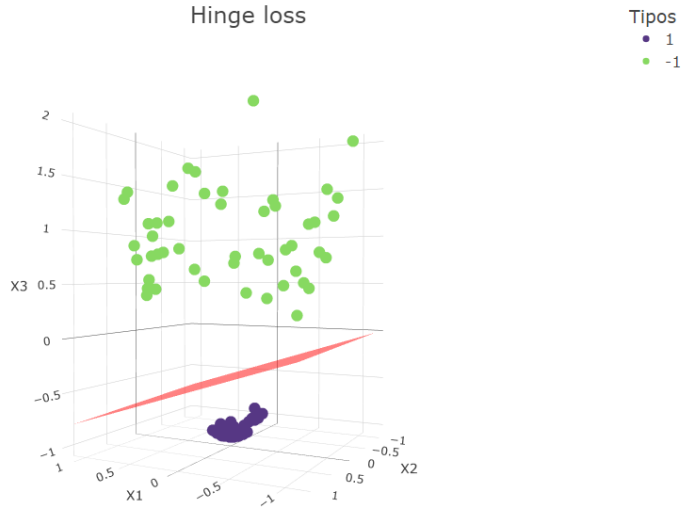

Figura 15: Representación del clasificador con hinge loss (rojo) sobre los datos transformados

4. Laboratorio 4

4.1. Ejercicio 5

En este laboratorio se ha utilizado el método de Support Vector Machine (SVM), de la librería e1071 de R, utilizando distintos conjuntos de datos. En la primera parte del experimento, se trabajó con datos linealmente separables, mientras que en la segunda parte se emplearon datos que no son linealmente separables.

4.1.1. Parte 1

En esta primera parte se ha conseguido la representación en un espacio 3D de los datos.

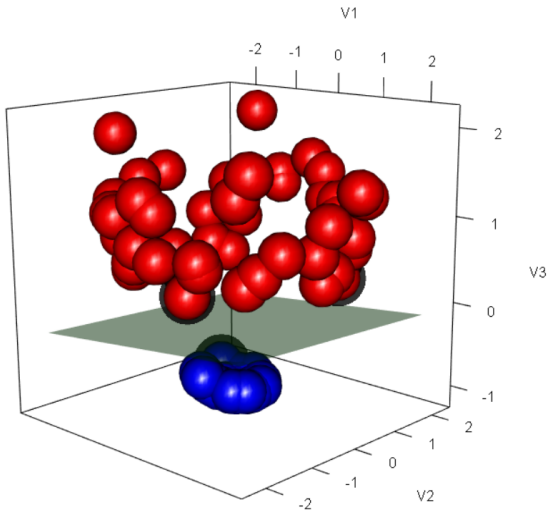

Figura 16: Gráfico 3D con el método SVM

En este gráfico se observa cómo los datos de diferentes clases pueden separarse mediante un plano. Al aplicar el método SVM, los **planos marginales** se colocan cerca del **hiperplano**, y se identificaron 4 **vectores de soporte**. Estos vectores de soporte son los puntos de datos más cercanos al hiperplano de separación.

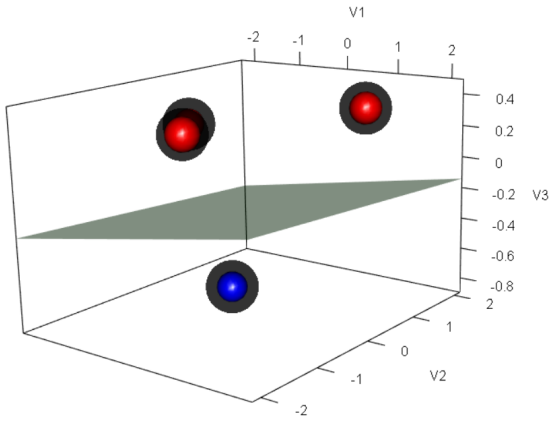

Figura 17: Gráfico 3D de los vectores de soporte

En el gráfico anterior, se pueden ver claramente los 4 vectores de soporte, y cómo el plano que separa las clases permanece igual. Esto ocurre porque el hiperplano de separación depende únicamente de los vectores de soporte, es decir, si se eliminan los demás datos, el plano que divide las clases seguiría siendo el mismo.

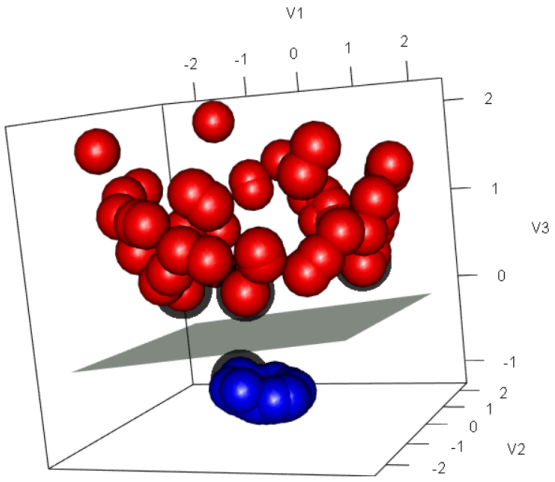

Figura 18: Gráfico 3D quitando un vector de soporte

Por último, si se elimina uno de los vectores de soporte del conjunto completo, el hiperplano de separación cambiará. Esto ocurre porque los vectores de soporte son los puntos clave que definen el margen y la posición del hiperplano. Al retirar uno de ellos, el SVM deberá recalcular el hiperplano óptimo utilizando los vectores de soporte restantes, lo que modificará tanto el margen como la orientación del hiperplano (Figura 18).

4.1.2. Parte 2

En esta parte hay que utilizar datos que no son linealmente separables.

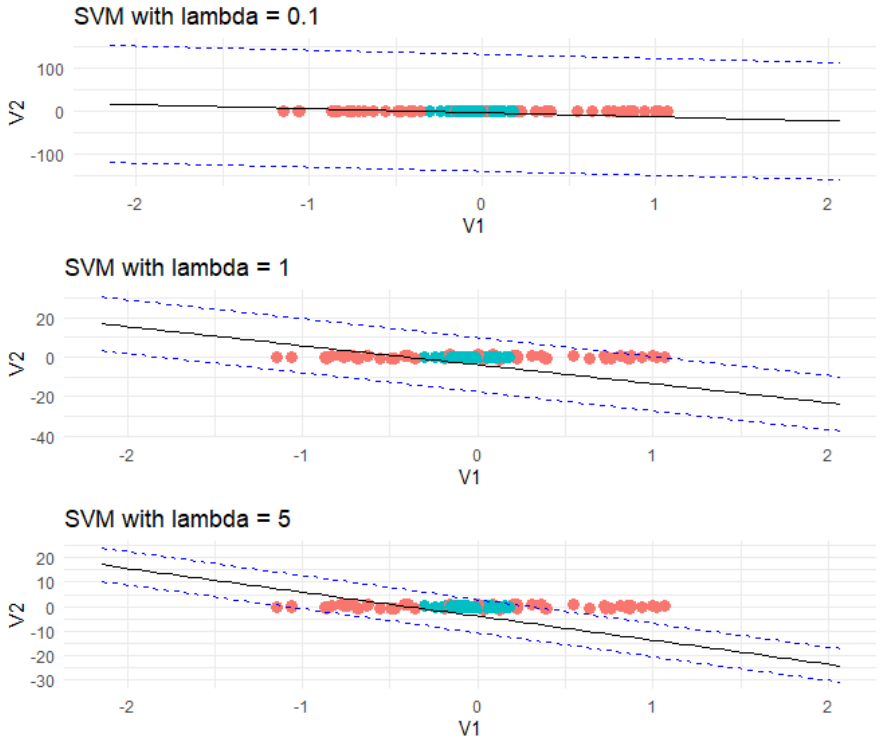

Figura 19: Gráfico 2D con el método SVM

En este gráfico se puede observar como los datos no son linealmente separables en un espacio de 2 dimensiones. Al aumentar los valores de λ (el parámetro de regularización o C) se puede observar como los **planos marginales** se van acercando al hiperplano de separación.

Es decir, cuando λ es 0.1 el hiperplano es casi horizontal y los márgenes son amplios. Este valor bajo de lambda permite un margen mayor, permitiendo que algunos puntos violen el margen, lo que genera un modelo más flexible. En cambio, cuando es 5 el hiperplano está inclinado y los margenes son aún más estrechos, ajustando más estrictamente el modelo a los datos.