Facultad de Ciencias Exactas, Ingeniería y Agrimensura

Introducción a la Inteligencia Artificial

TRABAJO PRÁCTICO 3

Alumno: Rodríguez Jeremías

11 de junio de 2017

1. Ejercicio 1

Implementé las partes faltantes de nb_n.c, siguiendo las indicaciones dadas en clase: aproximar cada feature de cada clase por una distribución normal cuyos parámetros son inferidos del conjunto de training.

2. Ejercicio 2

En este ejercicio aplicaré el clasificador programado en el ejercicio 1 al los problemas diagonal y paralelo que estudiamos en los trabajos anteriores. Como siempre, se probará el algoritmo para distintas dimensiones.

Los cantidad de puntos de training, validación y test son los mismos que se usaron en el trabajo de redes neuronales (tanto para este ejercicio como para los siguientes), con el fin de poder comparar en condiciones similares como funcionan los distintos métodos. Se utilizaron los mismos 20 datasets que en los trabajos anteriores.

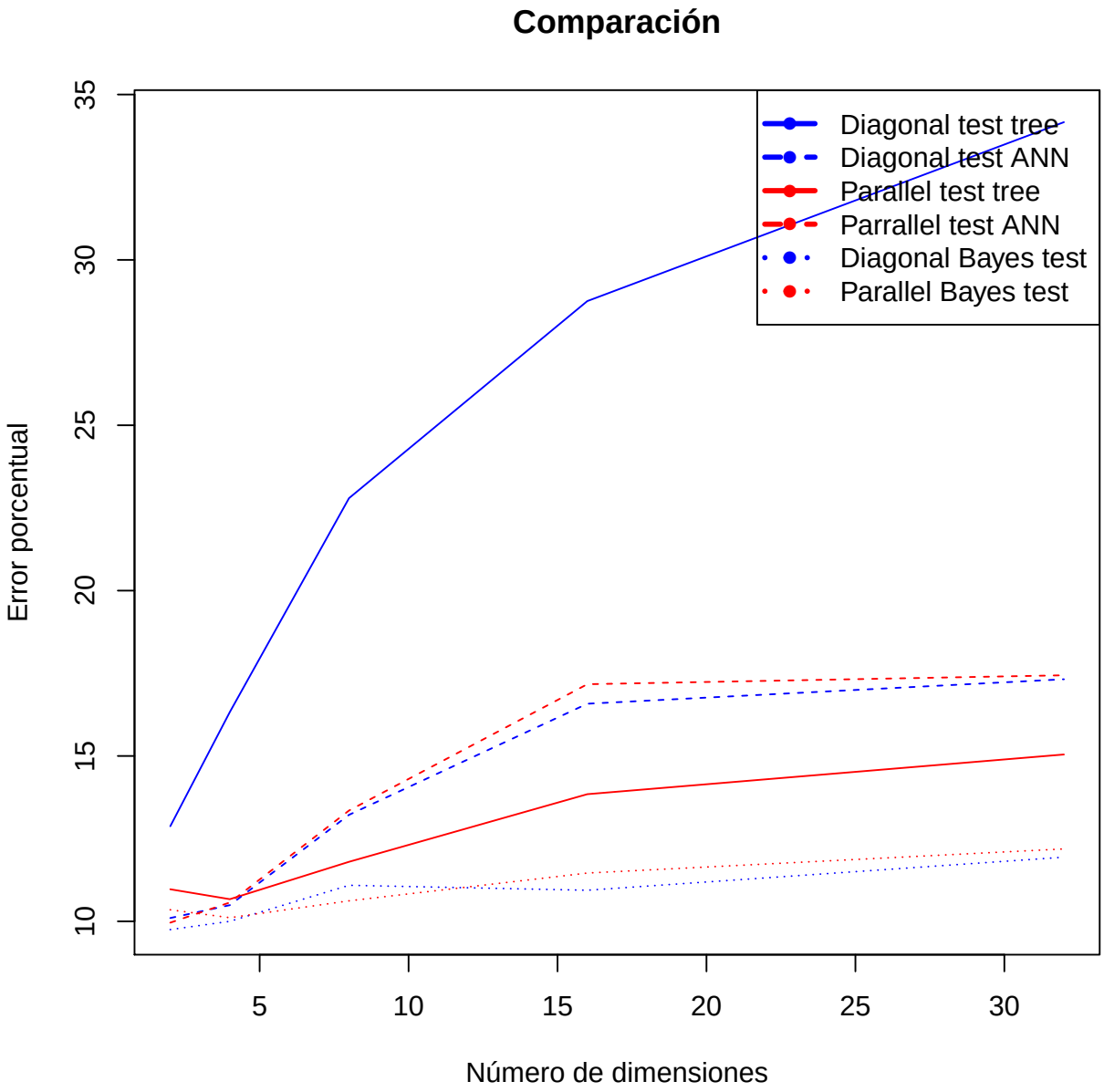
La siguiente tabla muestra los valores de error en test obtenidos para las distintas dimensiones:

d	Diagonal Error %	Paralelo Error %
2	11.650000	12.090000
4	11.510000	11.340000
8	12.650000	12.440000
16	13.600000	13.690000
32	15.930000	15.620000

Como podemos ver, al aumentar la cantidad de dimensiones, el error aumenta al igual que sucedía con los métodos estudiados anteriormente.

En la siguiente página se grafican estos datos y se comparan con redes neuronales y árboles de decisión. Puedo observar que:

- El clasificador Naive Bayes se da cuenta de que los problemas Diagonal y Paralelo son esencialmente el mismo, y los errores son prácticamente iguales.
- El clasificador de Bayes comete un error muy pequeño en comparación a redes y árboles. Esto se debe a que estamos clasificando bajo la suposición de que las distribuciones de cada feature son independientes, lo cuál es cierto en estos dos ejemplos; y a que estamos aproximando cada una de estas distribuciones por una normal, y estos datasets fueron generados justamente mediante una distribución normal. Por lo tanto, era esperable que nuestro clasificador Bayes produjera un resultado casi ópitmo. Si el tamaño del conjunto de training tendiera a infinito, la curva del error sería la óptima que dibujamos en los trabajos anteriores.
- Como ya se mencionó, el error aumenta al incrementar el número de dimensiones, al igual que sucedía en los otros métodos. Sin embargo, el aumento es muy ligero.

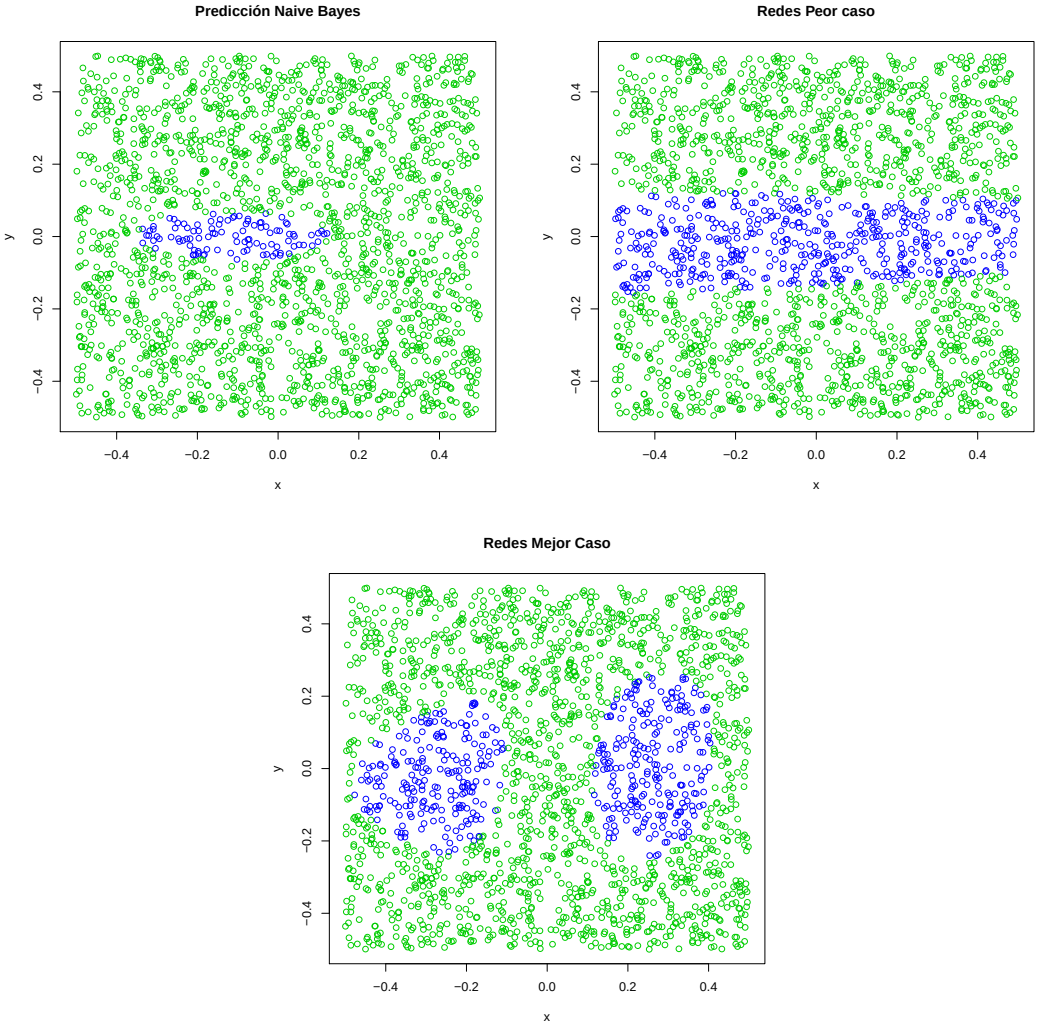

Comparación
Diagonal test tree
Diagonal test ANN
Parallel test tree
Parrallel test ANN
Diagonal Bayes test
Parallel Bayes test
Error porcentual
10
15
20
25
30
35
5
10
15
20
25
30
Número de dimensiones

3. Ejercicio C

En este ejercicio probaremos el clasificador Naive Bayes implementado en el apartado 1 con otros datasets donde los datos no fueron generados usando distribuciones normales. Se usaron los mismos parámetros que para entrenar las redes neuronales del trabajo práctico 2.

A continuación veamos los resultados para el problema de las dos elipses y su comparación con el resultado de redes neuronales:

Método	Error Test %
Naive Bayes con normales	23.450000
Redes Neuronales con Momentum/LR peor caso	19.400000
Redes Neuronales con Momentum/LR óptimo	6.650000

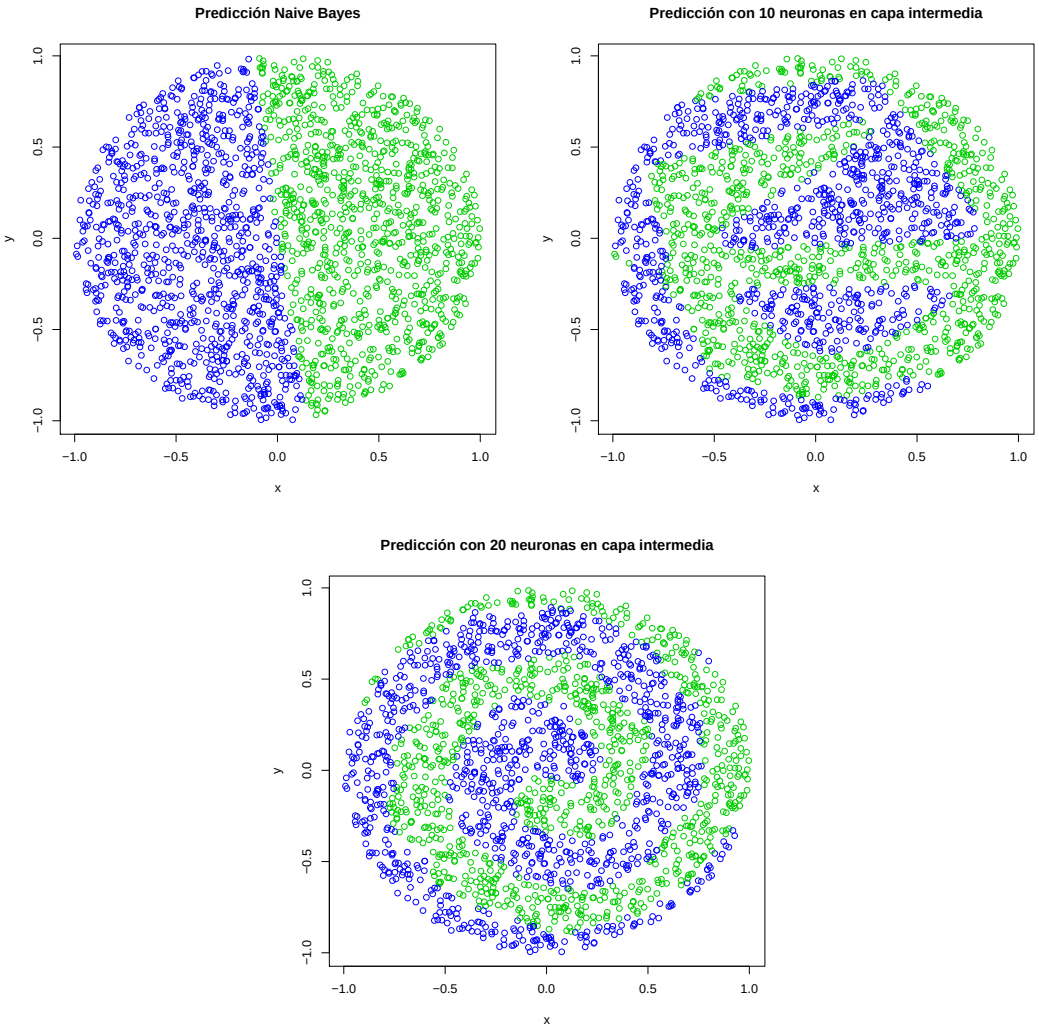


Como vemos, la predicción del clasificador naive Bayes es muy mala comparada con el mejor

caso de redes. Esto se debe a que los datos no tienen features independientes ni distribuciones normales en cada uno de ellos.

A continuación hago el mismo estudio sobre el dataset espirales anidadas:

Método	Error Test %
Naive Bayes con normales	43.750000
Redes Neuronales con 2 neuronas	39.649999
Redes Neuronales con 10 neuronas	23.000000
Redes Neuronales con 20 neuronas	9.800000



Se puede apreciar que la predicción del clasificador Bayesiano es muy básica, por el mismo motivo que con las espirales anidadas. Por lo tanto, concluyo que el clasificador del ejercicio 1 es muy poco flexible y no funciona nada bien con problemas que no siguen las hipótesis de features independientes y distribuciones normales en cada feature.

4. Ejercicio 4

En este ejercicio modificamos el clasificador del ejercicio 1. Seguiremos suponiendo features independientes pero ahora, en vez de estimar la probabilidad de cada feature fijada la clase con una distribución normal; usaremos un método más general: histogramas.

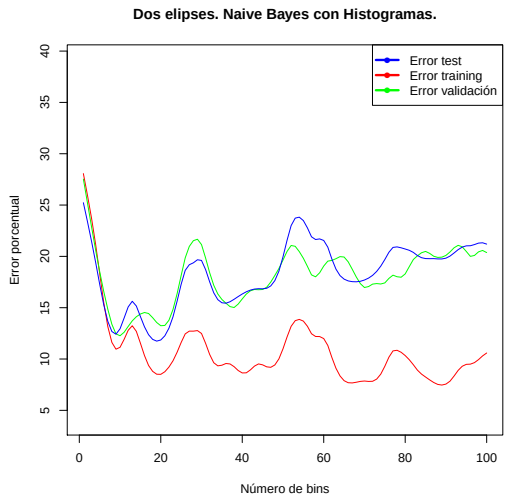
En mi implementación, **para cada clase** hallé el mínimo y máximo de cada feature. Luego, dividí el intervalo [min,max] en una cierta cantidad de cubetas (que se especifica en el archivo de parámetros .nb).

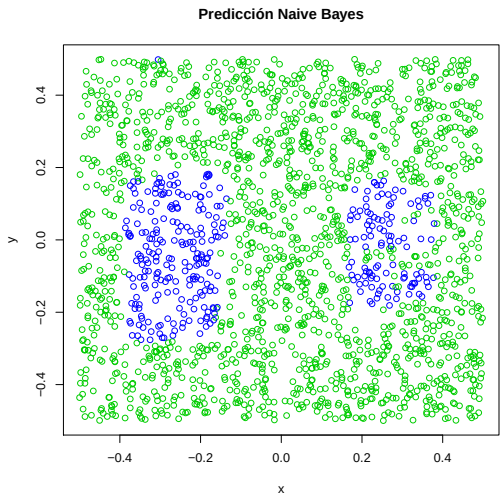
Finalmente armé los histogramas y apliqué la optimización indicada en el enunciado del trabajo práctico. Con esta nueva implementación, revisité los dos problemas del apartado C.

4.1. Dos elipses

Manteniendo siempre la misma configuración que en redes y variando la cantidad de cubetas entre 1 y 100, usé el clasificador 21 veces sobre cada configuración. Para cada número de bins seleccioné la ejecución que generó la mediana del error de test para confeccionar el gráfico de abajo a la izquierda. La curva es resultado de suavizar los puntos obtenidos.

De entre las 100 ejecuciones que representan el resultado para su número de bins, el ploteo de la derecha muestra la predicción de la clasificación que minimizó el error en test, usando 23 bins.





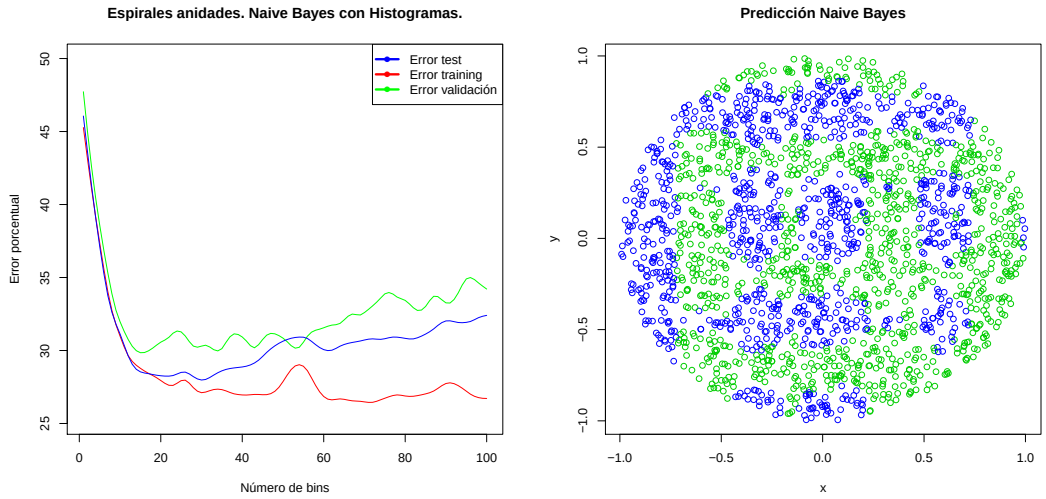
En primer lugar veo que es muy importante elegir el número correcto de bins, pues los errores varían mucho.

En segundo lugar, veo que la mejor predicción es muy buena en comparación a la predicción usando normales, y el error se aproxima al de redes neuronales con 20 neuronas. Aún así, el resultado no es del todo preciso. Esto se debe a que es más acertado predecir con histogramas que con normales en este problema particualar.

Finalmente, observando la gráfica vemos que si elegimos una cantidad grande de bins se produce sobreajuste. Esto sucede porque cada bin tendrá cada vez menos puntos.

4.2. Espirales Anidadas

Con la misma lógica, confeccioné el gráfico de la izquierda. El ploteo de la derecha muestra el mejor resultado, alcanzado con 9 bins.



Se puede observar, al igual que en el problema las dos elipses, que aproximar con histogramas resulta mucho más certero que con normales. Sin embargo, la predicción sigue siendo bastante mala, y por más que modifiquemos el número de bins no podremos llegar a algo mejor. Los features no son independientes, lo cuál contradice la asunción del algoritmo que estamos aplciando. Es muy mala en comparación a las predicciones generadas por redes neuronales más complejas.

También se produce sobreajuste al aumentar el número de bins, por el mismo motivo.

Para ver los datos completos producidos por el algoritmo, ver la última sección de este informe.

5. Ejercicio 5: Opcional 1

En este ejercicio, modifiqué brevemente el programa hecho en el apartado 4. Supongo que hay menos de 4 dimensiones en los datos de entrada.

La modificación consiste en dejar de suponer que los features son independientes. Seguiremos estimando la probabilidad a priori de cada clase como veníamos haciendolo. El cambio es que a la hora de estimar la probabilidad de que un punto (fila del .test por ejemplo) sea de una cierta clase c; dejaremos de calcularla como el producto de la probabilidad de que cada feature sea de clase c.

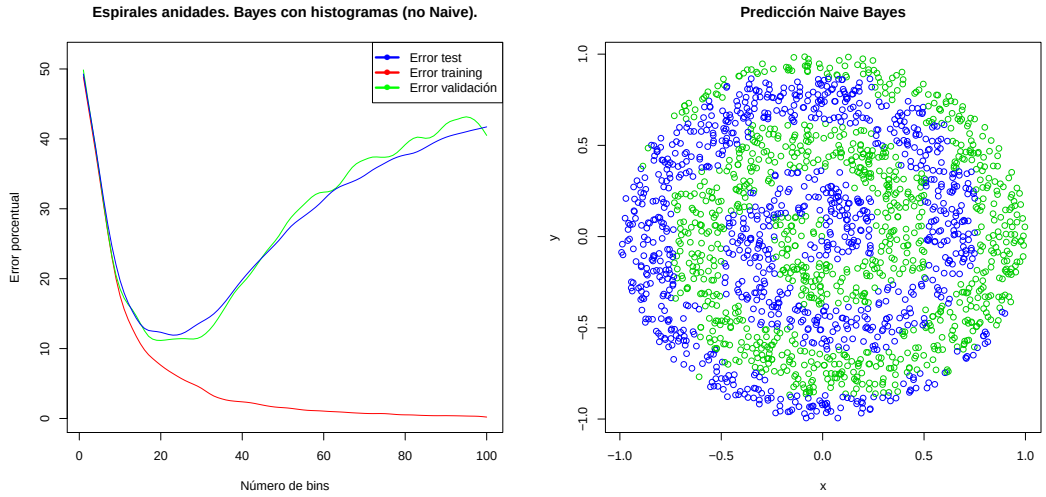
El nuevo enfoque es dividir el espacio en prismas (o el plano en rectángulos si el problema es 2-dimensional) y construir un histograma en 3 (2) dimensiones.

La forma de armar el histograma es la misma:

- Hallar el máximo y mínimo de cada feature. Con estos valores, armamos los extremos de un prisma (rectángulo) conteniendo todos los puntos de training.
- Subdividir este gran prisma (rectángulo) de modo que por cada dimensión tengamos N_BINS subdivisiones (N_BINS es un parámentro pasado por el usuario en .nb).

Por ejemplo, en el problema de las espirales anidadas, podemos visualizar la base del histograma como la espiral enmarcada en una cuadrilla de rectángulos, habiendo en total N_BINS^2 bins rectangulares.

Aplicando este algoritmo al problema de las espirales anidadas, se generó la gráfica presentada debajo. El ploteo corresponde a una predicción con 16 bins que obtuvo el menor error en test (9.950000 %). Se usó el mismo mecanismo que en los ejercicios anteriores (21 ejecuciones para cada número de bins y selección de la mediana).



Vemos que los errores disminuyen mucho respecto al ejercicio 4, y que el sobreajuste es muy pronunciado al aumentar la cantidad de bins. La predicción es muy acertada, rivalizando con la de redes neuronales.

La mejora respecto al clasificador naive se da porque en el problema de las espirales, los features no son independientes. El clasificador naive asumía esa independencia y en parte por eso aumentaba error.

Generalizar este programa a muchas dimensiones sería complejo. La candidad de bins es exponencial en el número de dimensiones. Esto conlleva una complejidad espacial y temporal considerable si pensamos en muchas dimensiones. Además se necesitarían cada vez más datos de training para que no estén casi todas las cubetas vacías. La implementación que yo realicé, para generalizarse, requeriría incluír muchos arreglos de longitud dependiente de la cantidad de features y muchos bucles; por lo que el código quedaría bastante más complejo.

Una optimización interesante sería poder indicar para cada dimensión, cuánto se desea subdividir ese eje. En mi implementación cada dimensión se divide por una misma constante N_BINS. Sin embargo ajustar estas constantes sería también complejo, pues habría que probar muchas combinaciones.

6. Código

En la carpeta *src* están los siguientes archivos:

- *nb_n.c* Código fuente del ejercicio 1.

- *nb_hist.c* Código fuente del ejercicio 4.
- *nb_hist_e.c* Código fuente del ejercicio 5.

En la carpeta *data* están los siguientes archivos:

- *Ej*4_*El.csv* y *Ej*4_*Es.csv*: Tablas mostrando la media de los errores para cada número de bins desde 1 hasta 100 ejecutando sobre el dataset Elipses / Espirales en el ejercicio 4.
- *Ej*5_*Es.csv*: Tablas mostrando la media de los errores para cada número de bins desde 1 hasta 100 ejecutando sobre el dataset Espirales en el ejercicio 5.