Universidad Nacional de Rosario

Introducción al Aprendizaje Automatizado

Trabajo Práctico I

Villagra Martín

9 de mayo de 2017

Introducción

En este trabajo se utiliza el algoritmo c4.5 para generar árboles de decisiones y analizar su desempeño sobre los datasets generados en el TP anterior y sobre el problema XOR.

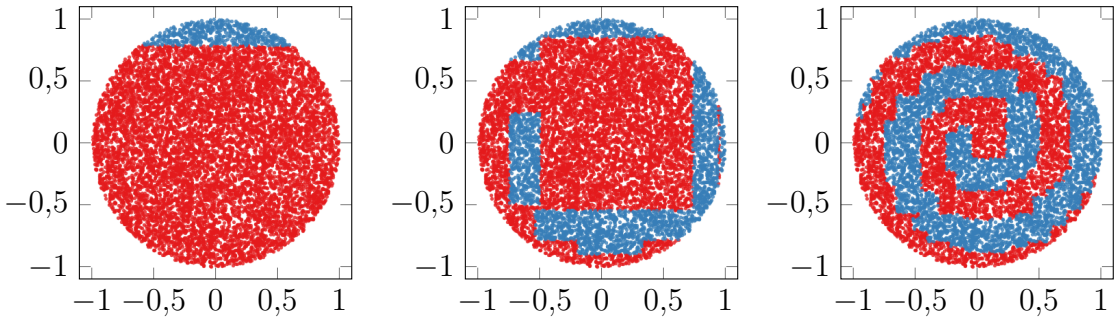
Se utilizaron los programas del TP anterior. Se usaron scripts en *Python* para extraer los errores y gráficar los resultados.

Para crear masivamente los datasets, promediarlos y realizar los gráficos correspondientes se confeccionaron scripts en *Bash*.

El comando *fdupes* fue de utilidad para verificar que no existieran datasets duplicados (que puede suceder si se usa accidentalmente la misma semilla).

Ejercicio 4

Lo primero que se observa en la Figura 1 es que cuánto más grande el conjunto de entrenamiento, mejor se vuelven las predicciones. Se puede apreciar como los límites entre las clases están bien delimitados y son paralelos a alguno de los ejes, este comportamiento lineal es inherente a los árboles de decisión con los que tratamos: Las condiciones sólo son de la forma $x \oplus K$ o $y \oplus K$, siendo K una constante y $\oplus$ un operador de comparación ($\leq, <, >$ o $\geq$).



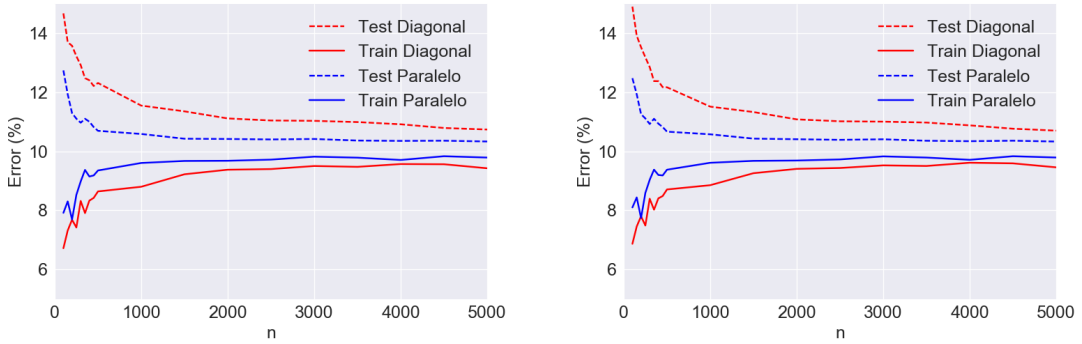
(a) $n = 150$ (b) $n = 600$ (c) $n = 3000$

Figura 1: Predicciones sobre el conjunto de test utilizando los diferentes conjuntos de entrenamiento propuestos.

Ejercicio 5

Se aumentaron la cantidad de conjuntos de entrenamiento a 60 y se probaron con más valores de n para obtener una mejor calidad en las gráficas.

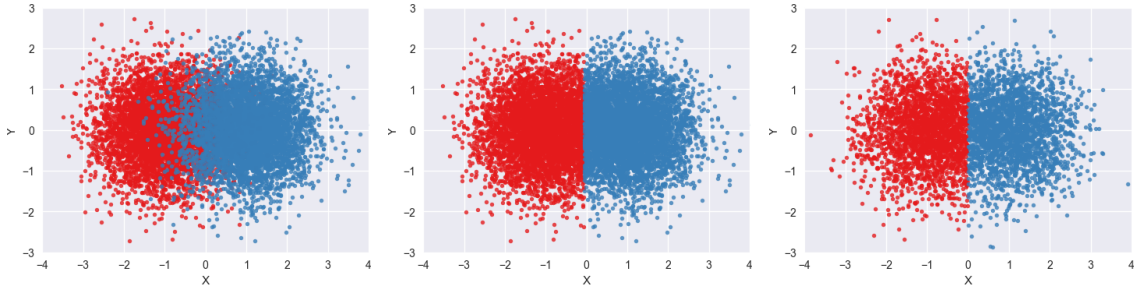
En la Figura 2 se aprecia una convergencia hacia valores cercanos al 10 %. Vemos que la predicción no mejora mucho para $n > 2000$. Luego del prunning el error de los árboles no cambia significativamente.



(a) Antes del prunning. (b) Después del prunning.

Figura 2: Error porcentual vs. tamaño del conjunto de entrenamiento.

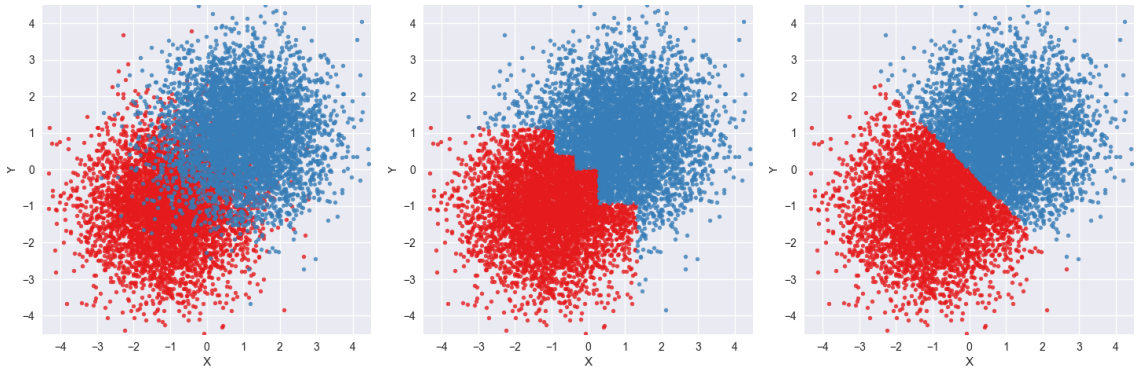
El conjunto Diagonal realiza más overfitting, logrando un menor error en el conjunto de entrenamiento. Sin embargo, es claro mirando al conjunto de test que el dataset Paralelo funciona considerablemente mejor que el Diagonal. Esta diferencia en los errores es explicada por las limitaciones en las condiciones, explicadas en el Ejercicio 4. En Paralelo, una simple condición de $x \geq 0$ produce una división muy satisfactoria y fácilmente generable por c4.5 (ver Figura 3).



(a) Dataset real. (b) Predicción del c4.5. (c) Predicción de Bayes.

Figura 3: Predicciones para el conjunto de test Paralelo.

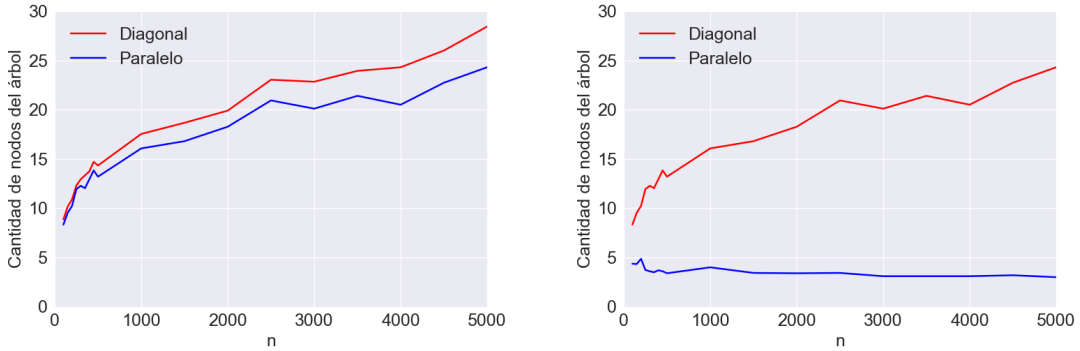
No sucede lo mismo para el Diagonal donde se requiere hacer una recta con pendiente $-\frac{1}{2}$. C4.5 sólo puede generar rectas paralelas a los ejes, lo que explica el escalonamiento observado en la Figura 4 (b).



(a) Dataset real. (b) Predicción del c4.5. (c) Predicción de Bayes.

Figura 4: Predicciones para el conjunto de test Diagonal.

Por otro lado, en la Figura 5 podemos ver que para el árbol del Diagonal el prunning no es muy efectivo: se requieren más nodos para poder representar la recta inclinada. La reducción en el tamaño del árbol del Paralelo es muy grande pues la recta óptima a representar es muy simple, como puede verse en la Figura 3.



(a) Antes del prunning. (b) Después del prunning.

Figura 5: Tamaño del árbol vs. tamaño del conjunto de entrenamiento.

Ejercicio 6

En primer lugar, como se intuyó en los ejercicios anteriores el clasificador Bayesiano es elegir la clase según cual de las dos medias esté más cerca de él, esto produce clasificadores que separan al dataset mediante una recta. Notar que si rotamos el conjunto Diagonal 45° obtendríamos el Paralelo por lo que ambos clasificadores tienen el mismo error como se ve en la Figura 6 (hay pequeñas diferencias por la cantidad límitadas de pruebas que podemos hacer).

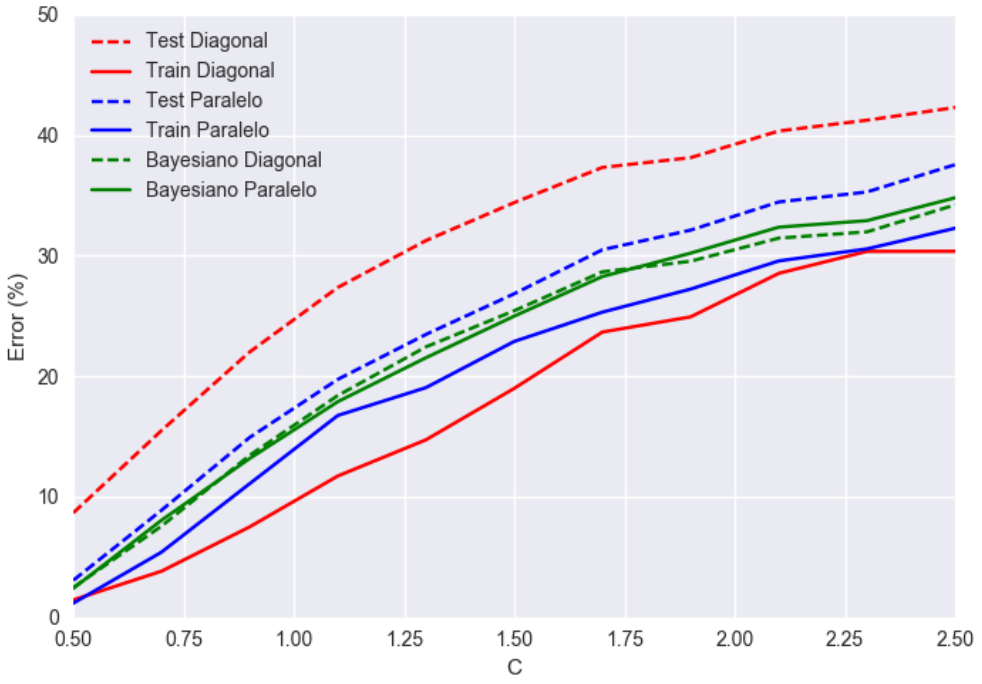


Figura 6: Error porcentual vs. dispersión del dataset.

Según la Figura 6 nuestro clasificador obtiene un error menor al bayesiano, esto sólo se debe al overfitting. Notar nuevamente como las predicciones del Paralelo se acercan mucho más al bayesiano, pues este es mucho más fácil de representar como se explicó en el Ejercicio 5.

Ejercicio 7

Observando la Figura 7, podemos observar comportamientos similares a los del Ejercicio 6: la superioridad en la predicción del conjunto Paralelo, a pesar de que Paralelo es sólo una rotación del Diagonal.

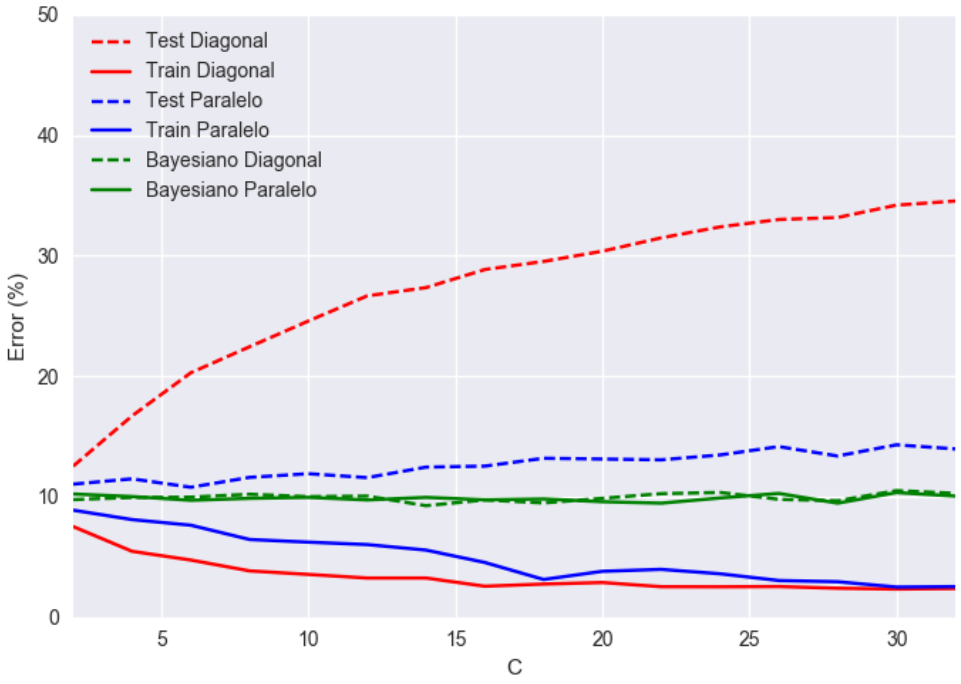


Figura 7: Error porcentual vs. dimensionalidad.

También se evidencia que el error aumenta con la dimensionalidad para el caso del Diagonal, mientras que para el Paralelo se mantiene relativamente constante. Esto se explica de la siguiente forma: para el Diagonal, cada árbol de decisión tiene que representar un hiperplano con dimensiones cada vez más grandes, mientras que en Paralelo sólo se debe representar la recta $x_1 \geq 0$ siempre.

Otro detalle es que a medida que se tienen más dimensiones se tienen más datos para hacer overfitting, por eso el error se reduce notablemente para el conjunto de entrenamiento pero en realidad no sucede lo mismo para el de test.

Ejercicio 8

El problema XOR siempre fue un desafío para los algoritmos de clasificación tradicionales. El usado para este ejercicio es el que se ve en la Figura 8.

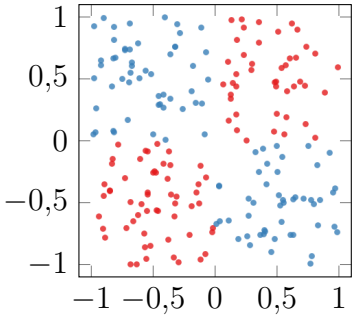


Figura 8: Dataset XOR.

Al correr el algoritmo de c4.5 obtenemos un árbol que clasifica todos los nodos en una misma clase (logrando un error del 50 %).

Para explicar este resultado recordemos que este algoritmo recursivo basado en *ID3* funciona buscando dos particiones del conjunto de forma de maximizar la ganancia en la entropía. Para nuestro caso continuo las particiones son generadas por rectas paralelas a los ejes. El problema subyace en que para este conjunto, ninguna recta individual de este estilo va a generar particiones con entropía estrictamente menor. Es decir para este caso, el approach goloso no funciona.

Sin embargo es fácil escribir un árbol que clasifique correctamente siempre como se muestra en la Figura 9.

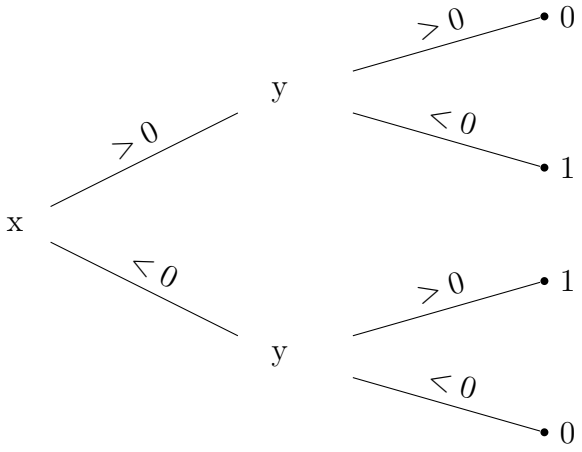


Figura 9: Árbol de decisión óptimo para XOR.