Técnicas de Inteligencia Artificial

Práctica 2 – Aprendizaje

Víctor Guzmán Pedrajas
DNI: 48717414–X
Turno: Jueves, 11:00-12:30

Índice

Introducción

A continuación se encuentra la documentación de la implementación del algoritmo Adaboost, que busca obtener un clasificador de puntos aprendiendo a partir de un conjunto de entrenamiento.

El algoritmo recibe el conjunto de puntos sobre los que entrenar, y devuelve un clasificador fuerte basado en estos puntos. Pueden consultarse también los clasificadores débiles generados, sean o no parte del clasificador fuerte, y las líneas generadas para crear cada clasificador débil.

Desarrollo del algoritmo Adaboost

El algoritmo se implementa en una clase llamada Adaboost, cuyo constructor recibe la lista de puntos, el número máximo de iteraciones, el número de líneas que se generan para obtener un clasificador débil y los límites del canvas donde se mostrarán los resultados.

Algoritmo

El algoritmo se basa en el que está en las transparencias de la asignatura, con lo que será:

1 Inicializamos los pesos de los puntos.

2 Para cada iteración:

  2.1 Obtenemos un clasificador débil, es decir:

    2.1.1 Generamos rectas aleatorias.

    2.1.2 Clasificamos todos los puntos del conjunto de entrenamiento, para calcular el epsilon y el alpha.

    2.1.3 Elegimos la recta con mayor alpha.

  2.2 Añadimos el clasificador débil al clasificador fuerte acumulado.

  2.3 Clasificamos los puntos del conjunto de entrenammiento con el clasificador fuerte.

3 Terminamos cuando terminan las iteraciones o cuando el clasificador fuerte clasifica correctamente el 100% de los puntos.

Clases necesarias

Para representar los datos necesarios para el algoritmo, se ha creado varias clases extra: Linea, clasificadorDebil, clasificadorFuerte y la clase Punto que ya venía con el proyecto inicial.

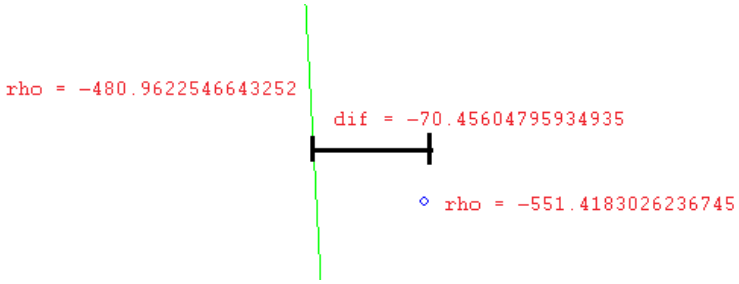
La clase Linea almacena una línea en forma paramétrica, almacenando un campo rho y uno thetha. Cuando se construye una línea, se le pasa también el tamaño del canvas, a partir del cual calcula los puntos de corte con el mismo, para poder dibujarla rápidamente.

La clase clasificadorDebil es un contenedor alrededor de una Linea y dos datos reales, epsilon y alpha. Esta clase tiene dos métodos *clasificar*. Si el método recibe un punto, lo clasifica y comprueba si ha acertado o no. Si el método recibe una lista de puntos y una lista de sus pesos correspondientes, intenta clasificar cada uno de estos puntos y establece el epsilon del clasificador según los errores cometidos.

La clase clasificadorFuerte es prácticamente igual a la clase clasificadorDebil, aunque en lugar de clasificar con una única línea, lo hará con un conjunto de líneas.

Método de clasificación

Para clasificar cada punto aprovechamos la forma en la que guardamos la recta del clasificador. Así, para saber si un punto está clasificado como positivo o como negativo, creamos una recta paralela al clasificador que pasa por el punto que buscamos, y restamos al rho de esta nueva recta el rho de la recta del clasificador. El signo de esta operación es la clasificación del punto.

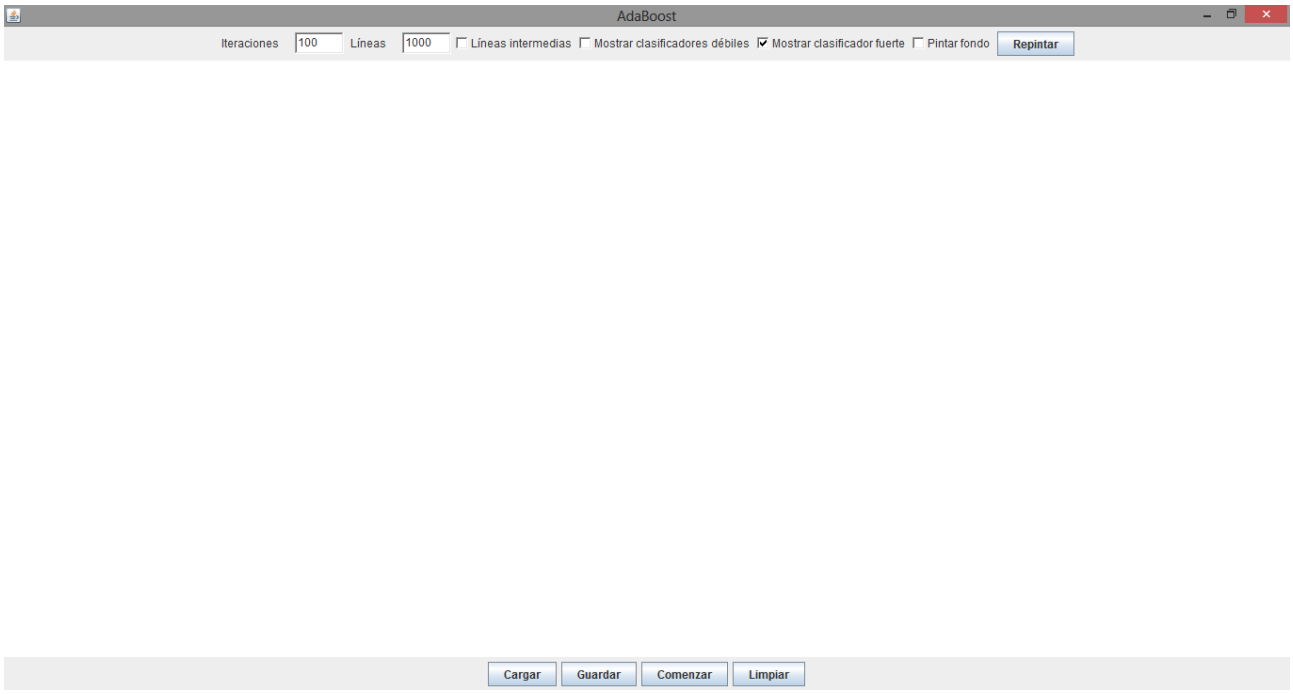


Clasificación errónea de un punto positivo.
La rho del punto corresponde a la de la línea paralela al clasificador que pasa por ese punto. La diferencia de rhos da negativo, clasificando al punto como negativo (rojo).

Para clasificar con el clasificador fuerte hacemos la operación anterior para cada clasificador débil perteneciente al fuerte, pero en lugar de devolver como resultado el valor devuelto por la clasificación, devolvemos la multiplicación del factor de confianza por el resultado. De esta forma se da más importancia a los que más confianza tienen.

Representación de resultados

Los resultados se representan en una interfaz gráfica como la siguiente:



Elementos de la interfaz

La interfaz está formada por tres zonas diferenciadas:

Área de dibujo

El área de dibujo es la zona en blanco en el centro de la ventana. Los clicks con el botón izquierdo del ratón pintan un punto azul en el área, y los clicks con el botón derecho pintan un punto rojo.

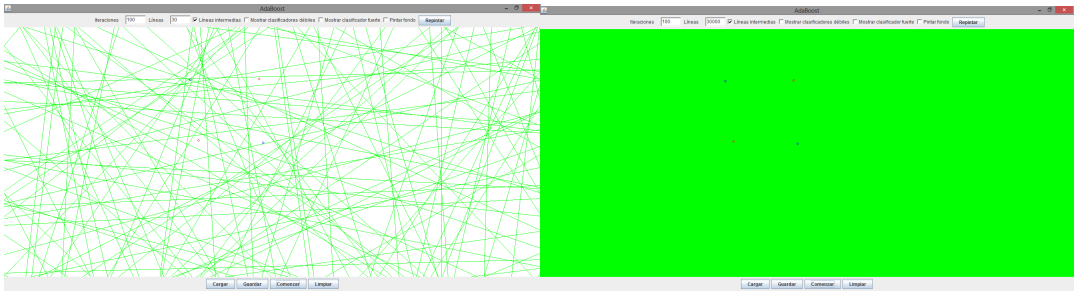
Barra superior

La barra superior sirve para modificar la ejecución y presentación del algoritmo.

Al principio aparecen dos campos de entrada de texto, en los que se puede introducir el número de iteraciones y el número de líneas deseadas. Estos campos están inicializados a 100 y 1000 respectivamente, pero el usuario puede elegir qué valores quiere. Si el valor introducido no es válido (no es un número, por ejemplo) se asumirán los valores por defecto.

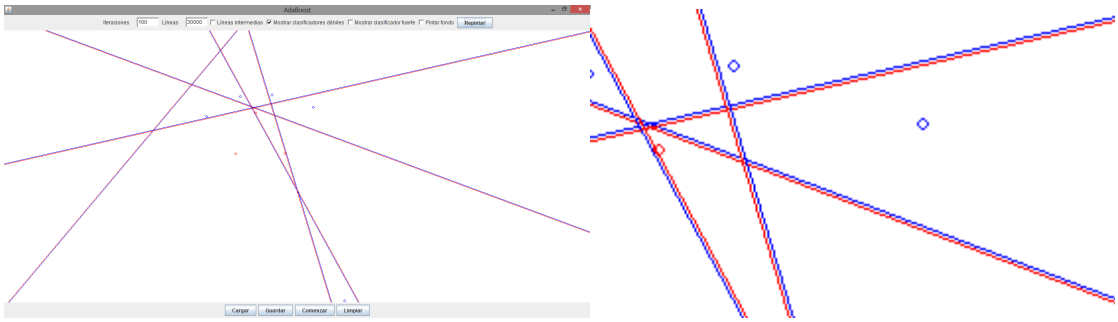
A continuación aparecen una serie de selectores de presentación, que modificarán la forma en que los resultados se presentan:

- Mostrar líneas intermedias. Cuando está seleccionado se mostrarán las líneas generadas para calcular los clasificadores débriles. Estas líneas se representan con una línea de un píxel de ancho de color verde. Para ver algo característico se recomienda introducir valores de iteraciones y líneas bajos, ya que si hay demasiadas líneas no pueden distinguirse.



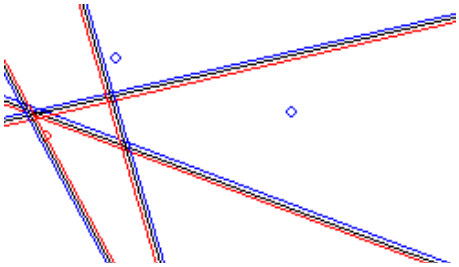
Comparativa entre mostrar 30 o mostrar 30.000 líneas

- Mostrar clasificadores débiles. Cuando está seleccionado se mostrarán los clasificadores débiles calculados. Estos se representan como dos líneas (una roja y una azul) de un píxel de ancho separadas por un espacio de un píxel.



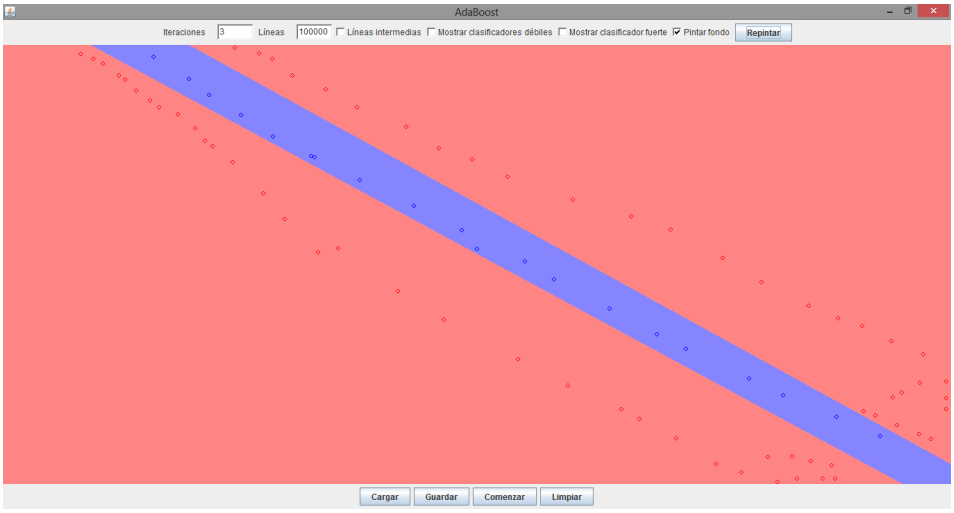
Muestra de los clasificadores débiles. A la izquierda la imagen total, a la derecha una sección ampliada

- Mostrar clasificador fuerte. Cuando está seleccionado se mostrará el clasificador fuerte resultante. Se representa como tres líneas: una roja y una azul a cada lado, y una negra en el centro.



Parte del clasificador fuerte de la imagen anterior

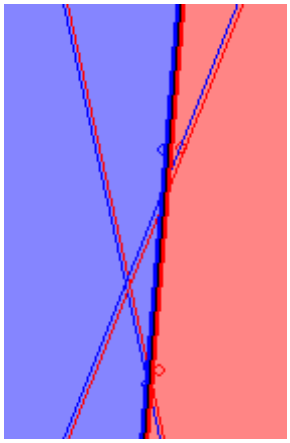
- Pintar fondo. Cuando esté seleccionado pintará el fondo según la clasificación de cada píxel de la imagen. Dado que hay que recorrer todos los píxeles, esta es una operación costosa, por lo que solo se hará cuando se llame al método de pintado con esta opción seleccionada. Conforme se pinta el fondo, este se almacena en memoria, de forma que si se quiere repintar el mismo fondo, se cargará el fondo almacenado, repintando de forma instantánea.



Fondo pintado de la solución a un ejemplo sencillo

Tras los selectores aparece el botón de repintado, que actualiza el canvas según las opciones seleccionadas. Este botón actualiza la vista con los últimos resultados obtenidos, es decir, no vuelve a ejecutar adaboost. Para ello es necesario pulsar el botón "Comenzar" de la barra inferior.

Todos los selectores pueden seleccionarse o deseleccionarse independientemente, de esta forma es posible mostrar el clasificador fuerte, los clasificadores débiles y el fondo pintado al mismo tiempo:



En esta imagen se muestra un clasificador fuerte (vertical) y tres débiles (dos desechados y uno aceptado) sobre un fondo coloreado. Se puede observar que cuando un clasificador débil no está incluido en el fuerte, aparece dibujado más débil.

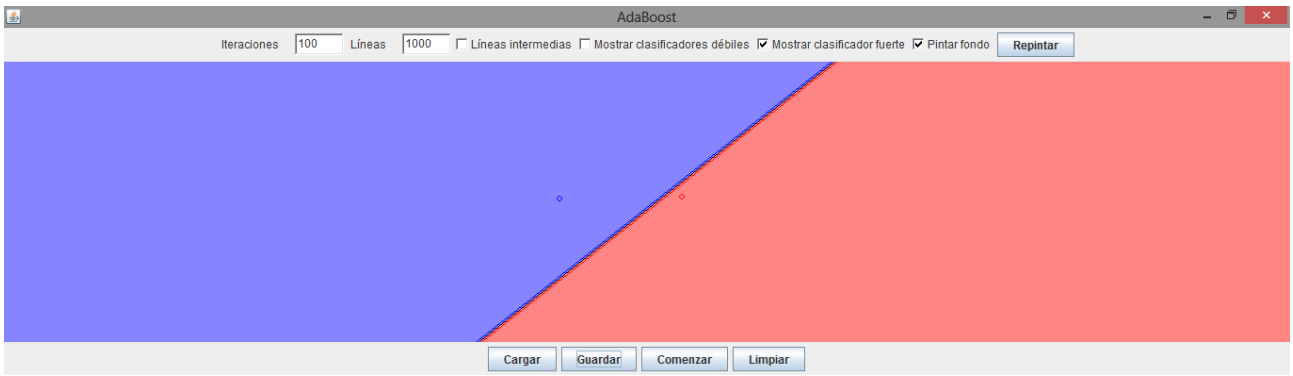
Barra inferior

La barra inferior muestra cuatro botones:

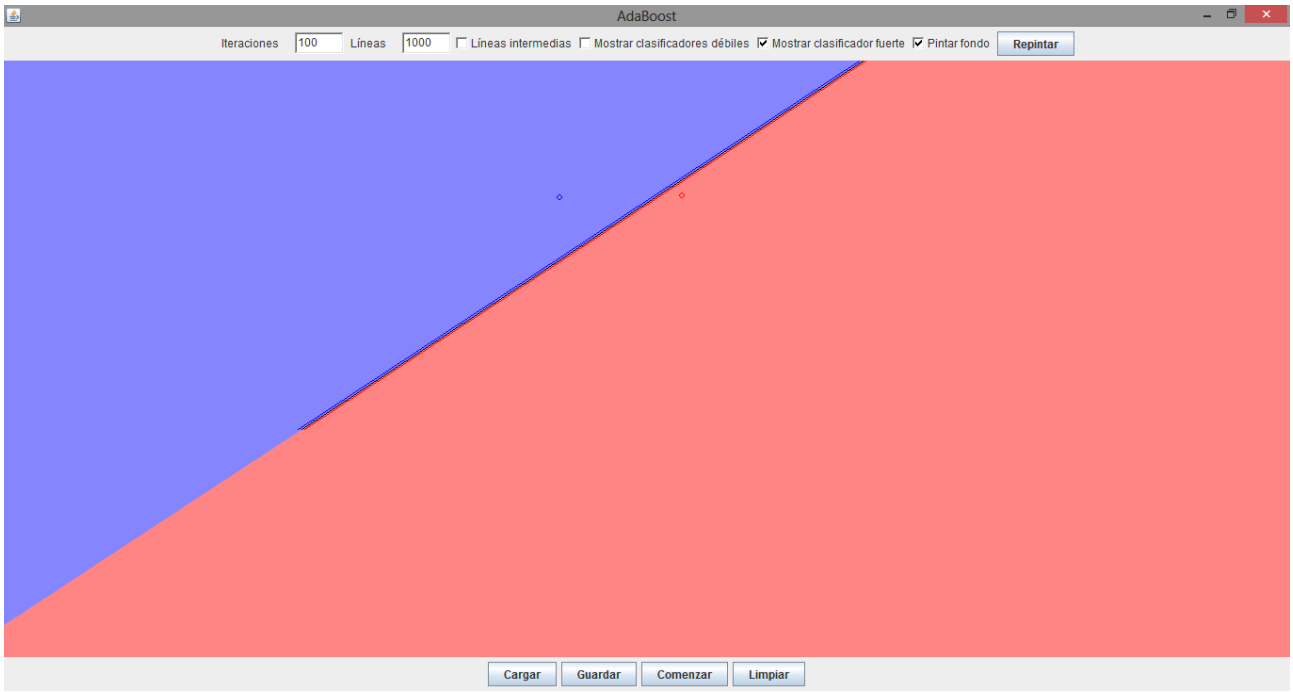
- Cargar. Limpia la lista de puntos y la rellena con los contenidos de un fichero. Los datos del fichero tienen que estar en formato "coordenadaX coordenadaY clase". Siendo la clase -1.0 ó 1.0.
- Guardar. Guarda la lista de puntos actual en un fichero con el formato anterior.
- Comenzar. Inicia el algoritmo adaboost con los datos actuales. Para ello lee las dos entradas de texto de la barra superior, borra los datos de la anterior iteración y aplica el algoritmo sobre la lista de puntos.
- Limpiar. Vacía la lista de puntos y los resultados de la ejecución anterior.

Tamaño de la interfaz

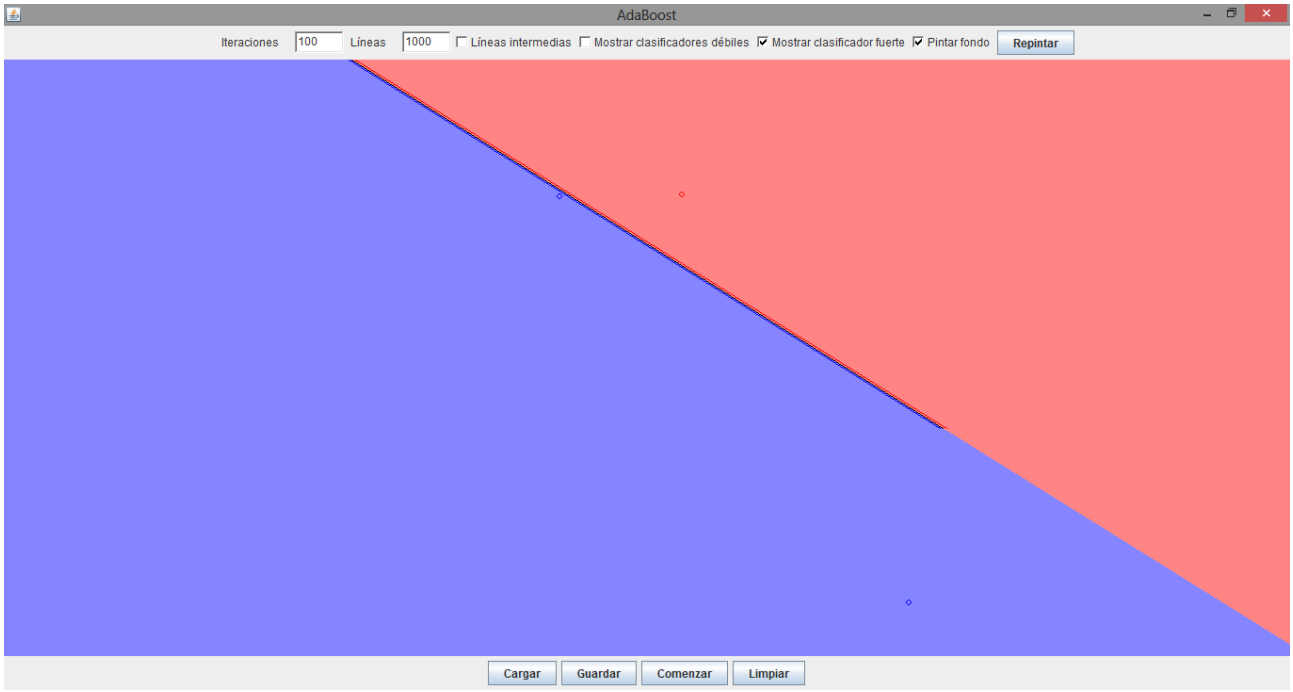
La interfaz gráfica ajusta su tamaño original al tamaño disponible en la pantalla, es decir, al tamaño total de la pantalla sin contar con la barra del sistema. En el momento de crear la interfaz se obtiene ese valor, que es el que se suministra a las demás funciones para pintar los elementos. Eso significa que si se redimensiona la pantalla después de crear la interfaz, aunque los puntos que se encuentren fuera del tamaño original se tendrán en cuenta para los cálculos, las líneas de resultado no se pintarán fuera del tamaño original. Sí que se pintará el fondo y los puntos añadidos después, ya que actúan directamente sobre el canvas.



Ventana inicial, con tamaño reducido (barra del sistema más grande)



Ventana tras aumentar su tamaño y volver a ejecutar el algoritmo



Ventana tras añadir un nuevo punto (azul) fuera del tamaño original y volver a ejecutar el algoritmo

Salida textual

El algoritmo también muestra por la consola el progreso actual del algoritmo. Al iniciar, se muestra un texto que indica "*Empezando el algoritmo AdaBoost*".

Tras esto, cada iteración se muestra los resultados obtenidos, mostrando el factor de error (epsilon) y el de confianza (alpha) de cada clasificador débil y el factor de error del clasificador fuerte acumulado hasta ese momento.

Si se encuentra un clasificador débil con alpha infinito (es decir, este clasificador solo ya clasifica el conjunto al 100%) se indica que se desechan los anteriores y se elige como clasificador fuerte.

Una vez terminadas las iteraciones, o al encontrar un clasificador fuerte que clasifique perfectamente, se indica en qué caso estamos, indicando el factor de error en caso de no haber encontrado el clasificador perfecto.

Por último, se muestra el número total de clasificadores débiles que componen el clasificador fuerte final.

```
=====================================
Empezando el algoritmo AdaBoost

== Iteración 0 ==
El factor de confianza es alpha =
0.11983642663271013
Y su epsilon = 0.44036697247706424
El epsilon del clasificador fuerte acumulado es
0.44036697247706424

.
.
.

== Iteración 99 ==
El factor de confianza es alpha =
0.08456272025368676
Y su epsilon = 0.45781913503867744
El epsilon del clasificador fuerte acumulado es
0.2280267759844286
No se ha encontrado un clasificador perfecto, el
mejor encontrado tiene epsilon= 0.18458292656910658
Número de clasificadores = 91
```

Ejemplo de salida textual de un problema que tras 100 iteraciones no se ha resuelto.

Experimentación

La experimentación tiene como objetivo ajustar los valores de iteraciones y líneas, así como encontrar los casos extremos del algoritmo.

Iteraciones

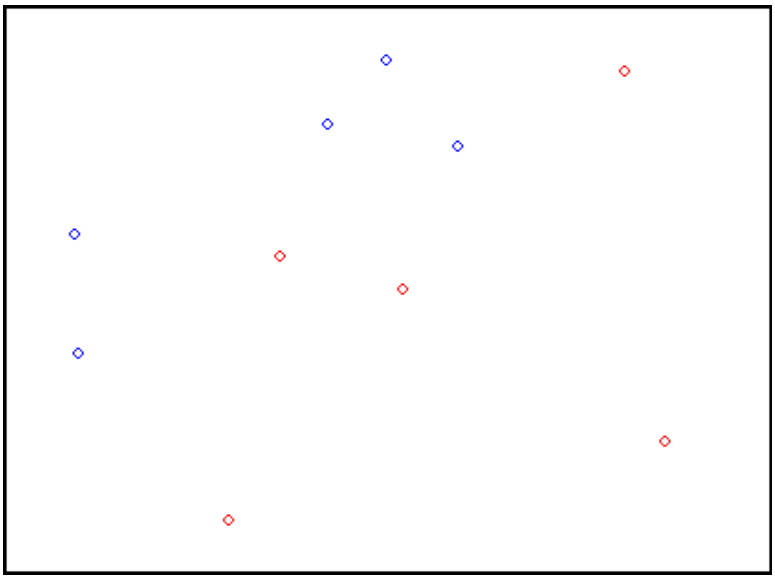
El número de iteraciones significa la cantidad de clasificadores débiles que, como máximo, tendrá el clasificador fuerte.

No es un parámetro excesivamente determinante, ya que a partir de un mínimo de iteraciones, los resultados que se obtienen no son especialmente característicos. Esto es así porque en la mayoría de casos, un clasificador compuesto por máximo 20 ó 30 clasificadores débiles es capaz de encontrar un clasificador perfecto. Normalmente, si en ese punto no hay una clasificación

suficientemente buena, es que no existirá.

Para experimentar con el valor de iteraciones, se ha fijado el número de líneas a 1000, y se ha probado con distintos valores para cada distribución en la que se ha probado.

Prueba 1

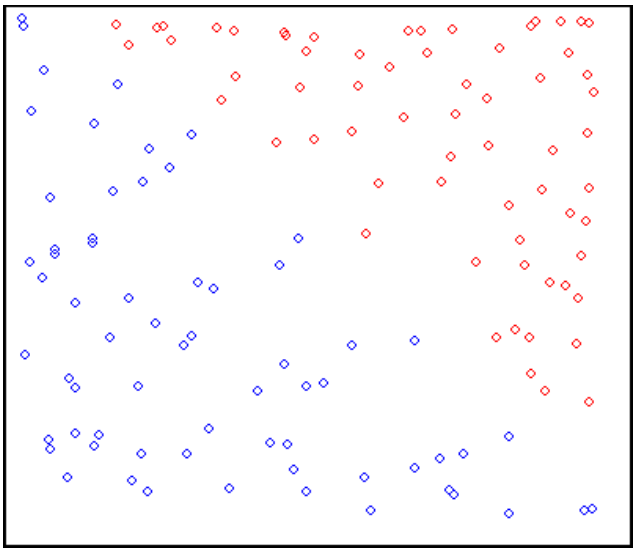


En esta distribución se puede encontrar una única línea que divide todos los puntos perfectamente. A continuación figuran los resultados de ejecutar 10 veces con cada dato de iteraciones:

Iteraciones	Líneas	Clasificaciones correctas	Clasificador óptimo
1	1000	0	0
2	1000	0	0
4	1000	8	0
8	1000	10	1
16	1000	10	1
32	1000	10	1
64	1000	10	1
128	1000	10	0
1000	1000	10	0

Se puede observar que la cantidad de clasificaciones correctas aumenta de golpe hasta el máximo, mientras que las clasificaciones óptimas no siguen una regla concreta.

Prueba 2

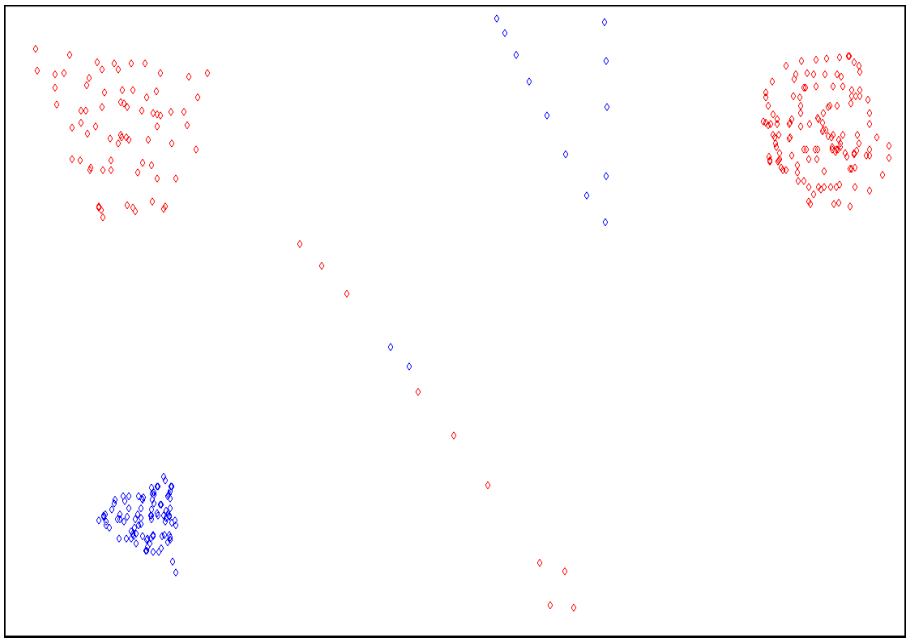


Este ejemplo, más sencillo, también tiene una posible línea que divide completamente la muestra.

Iteraciones	Líneas	Clasificaciones correctas	Clasificador óptimo
1	1000	3	3
2	1000	4	4
4	1000	9	6
8	1000	10	8
16	1000	10	8
32	1000	10	8
64	1000	10	9
128	1000	10	7
1000	1000	10	7

En este caso, aunque hay una mayor cantidad de casos óptimos, sigue sin establecerse una relación clara entre iteraciones y optimalidad.

Prueba 3



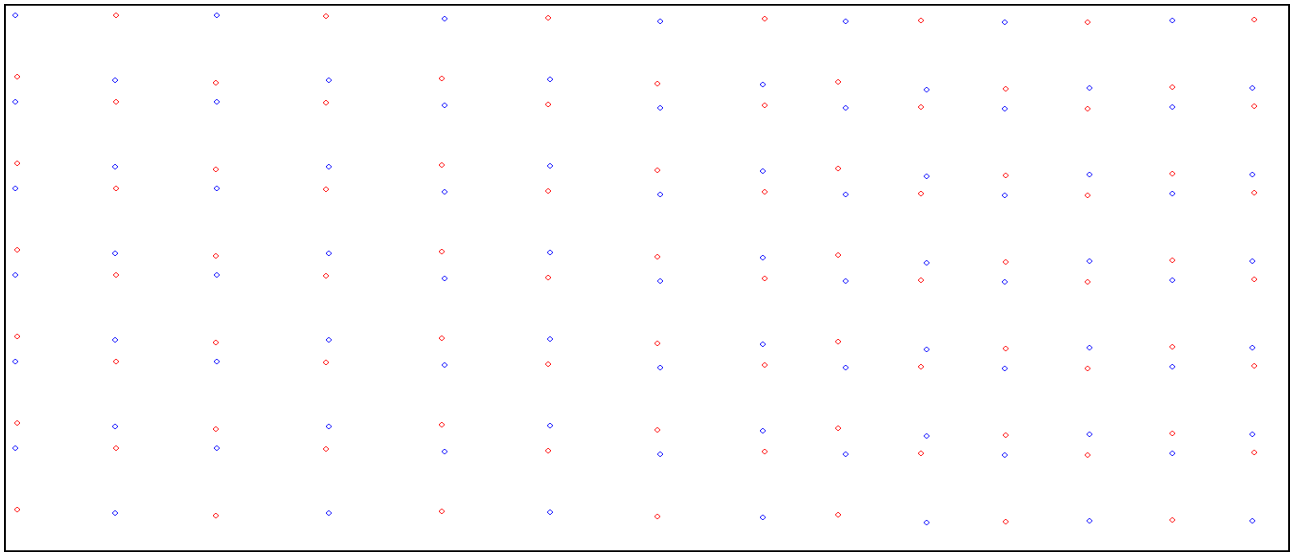
Dado que en los anteriores se observaba un límite en el número de casos correctos a partir del momento en que hay más iteraciones que clasificadores necesarios, se buscó un caso sin solución, para agotar las iteraciones lo más posible. Este ejemplo no ha encontrado una solución perfecta en ninguna de las pruebas que he pasado, hasta que el tiempo de cálculo era prohibitivo.

Dado que ningún clasificador ha sido perfecto, se indica el dato de cantidad de error obtenida para cada entrada:

Iteraciones	Líneas	Media de errores
1	1000	0,04
2	1000	0,04
4	1000	0,03
8	1000	0,03
16	1000	0,02
32	1000	0,02
64	1000	0,02
128	1000	0,02
1000	1000	0,03

Se puede observar una pequeña mejora conforme avanzan las iteraciones, pero no es constante, y a partir de cierto punto vuelve a empeorar.

Prueba 4



Este ejemplo busca un caso como el anterior, pero sin llegar a ser infinito. Lógicamente, en este caso, el número de iteraciones sí determina el éxito o no del clasificador, ya que hay un número mínimo de clasificadores necesarios para clasificar el 100%. Todos los valores de iteraciones por debajo del mínimo obtendrán un resultado.

Iteraciones	Líneas	Clasificaciones correctas	Media de errores
1	1000	0	0,5
2	1000	0	0,46
4	1000	0	0,42
8	1000	0	0,4
16	1000	0	0,33
32	1000	0	0,25
64	1000	0	0,1
128	1000	3	0,01
256	1000	10	0
1000	1000	10	0

Se observa claramente cómo hasta alcancar el mínimo de iteraciones (que según los resultados está alrededor de 128) el valor de error va bajando, y en el momento en que llega a 0, un aumento de las iteraciones no mejora el resultado.

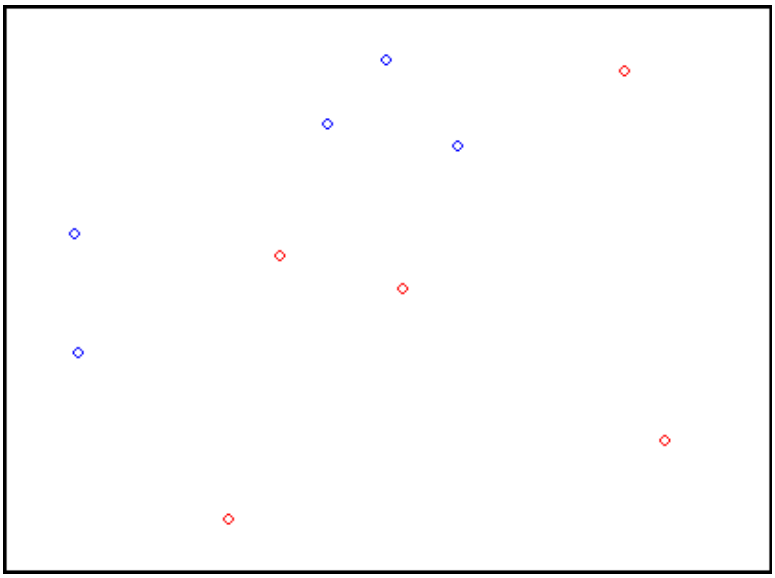
En resumen, la mayoría de los problemas no necesitarán muchas iteraciones para resolverse, aunque habrá algunos que hará falta ir aumentando las iteraciones hasta encontrar un valor válido. Un valor por defecto de 100 iteraciones hace que la mayoría de los problemas se solucionen correctamente, teniendo un rango suficiente para los problemas más complicados.

Líneas

El número de líneas indica cuántas líneas aleatorias se probarán a la hora de generar los clasificadores débiles. Aumentar el número de líneas hace que se pueda obtener resultados más óptimos, ya que aumenta la probabilidad de que una línea pase por el punto correcto.

A continuación figuran las mismas pruebas que el caso anterior, pero manteniendo fijas las iteraciones a 100, y modificando el valor de las líneas.

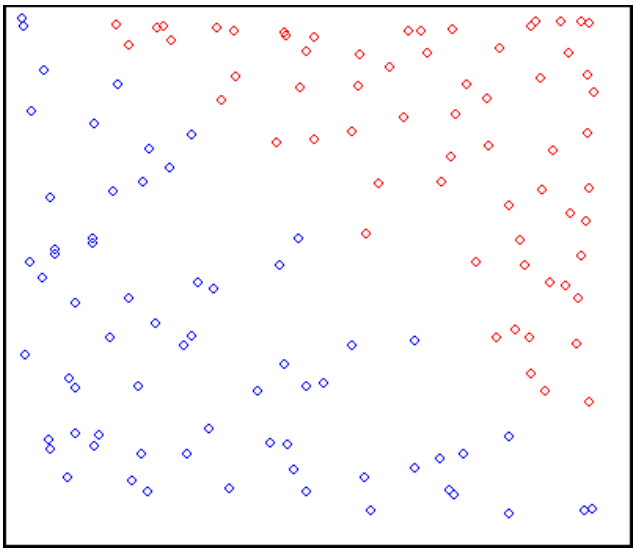
Prueba 1



Iteraciones	Líneas	Clasificaciones correctas	Clasificador óptimo
100	1	10	0
100	10	10	0
100	50	10	0
100	100	10	0
100	500	10	0
100	1000	10	1
100	5000	10	3
100	10000	10	5
100	100000	10	10

Se observa que, aunque siempre clasifica correctamente debido al número de iteraciones, la probabilidad de encontrar el clasificador óptimo va aumentando con el número de líneas.

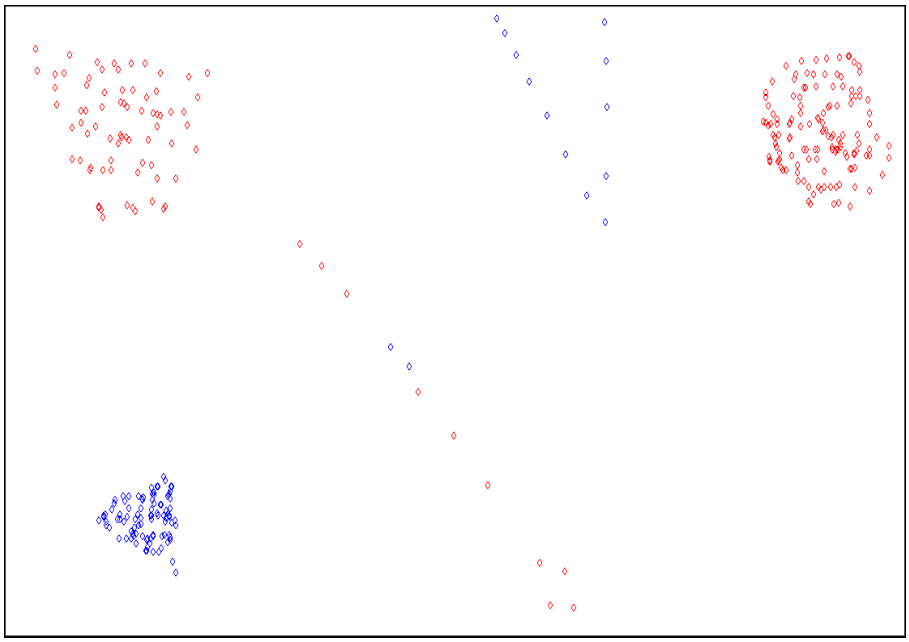
Prueba 2



Iteraciones	Líneas	Clasificaciones correctas	Clasificador óptimo
100	1	10	0
100	10	10	1
100	50	10	1
100	100	10	1
100	500	10	7
100	1000	10	10
100	5000	10	10
100	10000	10	10
100	100000	10	10

Se observa lo mismo que en la anterior: la probabilidad de encontrar el clasificador óptimo va aumentando con el número de líneas.

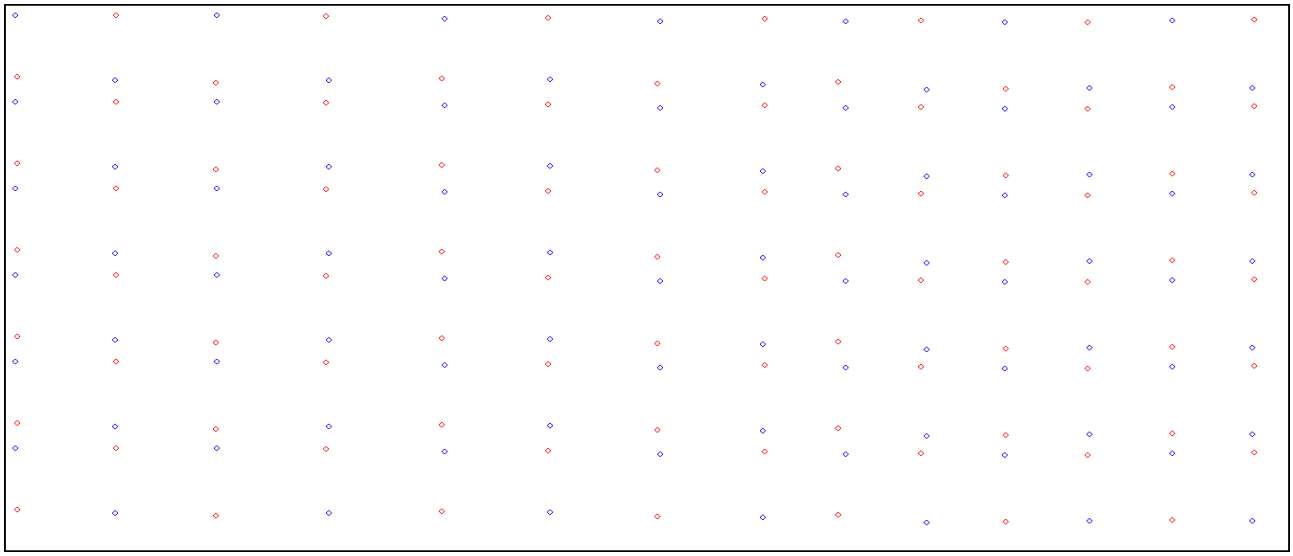
Prueba 3



Iteraciones	Líneas	Media de errores
100	1	0,08
100	10	0,07
100	50	0,04
100	100	0,04
100	500	0,03
100	1000	0,025
100	5000	0,02
100	10000	0,02
100	100000	ERROR DE MEMORIA

Se puede observar que la media de error, aunque empieza ya en un nivel bajo, va bajando poco a poco conforme aumentan las líneas, hasta que da error de memoria por la excesiva cantidad de líneas generadas.

Prueba 4



Dado que el número de líneas busca aproximarse al clasificador óptimo, en esta prueba mediremos la cantidad de líneas presentes en el clasificador final.

Iteraciones	Líneas	Clasificaciones correctas	Cantidad de líneas en el clasificador
100	1	0	175
100	10	0	195
100	50	0	193
100	100	0	180
100	500	7	172
100	1000	10	153,6
100	5000	10	140
100	10000	10	122,6
100	100000	10	ERROR DE MEMORIA

En esta tabla se observa como conforme aumentan las líneas, la cantidad de líneas se va acercando al valor óptimo (la mejor ejecución que he obtenido es de 108). También se ve que por debajo de cierto límite, los clasificadores son tan imprecisos que dejan de reconocer el 100% de los puntos.

En resumen, el número de líneas indica lo preciso que será cada clasificador débil, por lo que es recomendable un valor alto. Sin embargo, un valor excesivamente alto, además de incrementar exponencialmente el tiempo de cálculo del algoritmo, consume demasiada memoria. Por esta razón se pone como valor por defecto 1000, que es un valor suficiente pero no excesivo.