Introducción a la Minería de Datos

J. Mario García Valdez

Otras Técnicas de Clasificación

Clasificadores basados en reglas

En la sección anterior, el modelo utilizado para clasificar se representaba como un árbol de decisión. Los clasificadores que veremos a continuación representan al modelo como conjunto de reglas **IF-THEN**. Estas reglas son similares a las condiciones de los nodos de los árboles de decisión, una regla para clasificar hoteles podría ser:

```
Regla 1: IF alberca = sí THEN WifiGratis = Sí.
```

Las reglas tienen dos partes:

1. El antecedente o precondición **IF** en la cual hay expresiones condicionales sobre los atributos, de manera opcional utilizando operadores lógicos, por ejemplo, *alberca = sí* OR *estrellas < 5*.
2. El consecuente **THEN** donde se expresa la predicción de la clase o categoría a la que pertenece el objeto.

Ejemplo

Recordemos el ejemplo de los hoteles visto anteriormente:

id	hotel	estrellas	alberca	WiFi Gratis
1	Mariots	2	Sí	Sí
2	Díaz Inn	2	No	Sí
3	Mandarina	5	Sí	No
4	Le Hotel	3	Sí	No
5	Halton	4	No	Sí
6	Tromp	4	Sí	No

Un modelo basado en reglas para clasificar a los hoteles puede ser:

```
IF estrellas = 5 OR estrellas = 3 THEN WiFiGratis = Sí
IF alberca = No THEN WiFiGratis = Sí
IF alberca = Sí THEN WiFiGratis = No
```

Los algoritmos clasificadores basados en reglas extraen las reglas de los datos mediante:

1. La extracción de reglas a partir de árboles de decisión. Las rutas de la raíz a las hojas son las precondiciones las hojas son los consecuentes (**???**).
2. Se generan las reglas a partir de los datos, utilizando un algoritmo de covertura, por ejemplo RIPPER (**???**) o CN2 (**???**):

k vecinos más próximos

Este algoritmo se basa en una idea sencilla: asignar la clase que tengan los objetos del conjunto de entrenamiento que más se parezcan al objeto a clasificar. Para calcular la similaridad entre dos objetos, se puede cualcular simplmente la distancia euclidiana entre los vectores de características de cada objeto. El parámetro k especifica cuantos objetos (ordenados por similaridad) se van a considerar para la asignación. En el caso más sencillo se le asigna al objeto la clase mayoritaria.

El método es muy fácil de implmentar. La selección del valor de k pude afectar el desempeño del algoritmo. La figura muestra un ejemplo de clasificación para distintos valores de %k%. Si elegimos un valor de k muy pequeño puede ser afectado por el ruido o valores atípicos, por otro lado valores muy grandes se tenderá a considerar un número mayor de datos con otras clases. El valor del voto que asigna cada objeto puede ponderarse con respecto a la distancia.

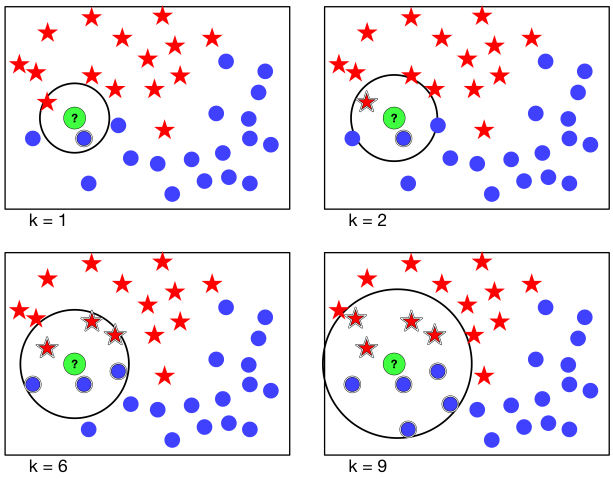


Figura 1: Efecto de la elección del número de vecinos k

Naïve Bayes

Este clasificador básico recibe el nombre de Naïve que se traduce al español como ingenuo. La razón de esto es que considera que los valores de los atributos de un objeto son variables independientes. Es fácil ver que esta consideración |no siempre es real. Por ejemplo, para los atributos de un hotel:

«número de estrellas» y «alberca» podemos imaginar que un hotel de más de 4 estrellas es muy probable que cuente con una o más. También es muy probable que un hotel de dos estrellas |no cuente con una. Entonces, un clasificador Naïve Bayes considera que para una clase C dado un objeto con los atributos $\{A_1, A_2, \dots, A_n\}$ podemos calcular la probabilidad condicional:

$$P\{C|A_1, A_2, \dots, A_n\}$$

Esto significa tratar de estimar las probabilidades a partir del conjunto de datos de entrenamiento. Para clasificar un registro se debe de calcular la probabilidad condicional para cada clase C_j y elegir la mayor:

$$P\{C_j|A_1, A_2, \dots, A_n\} = \frac{P\{A_1, A_2, \dots, A_n|C_j\} \cdot P(C_j)}{P\{A_1, A_2, \dots, A_n\}}$$

Como solo estamos interesados en elegir la mejor opción basta con calcular:

$$P\{A_1, A_2, \dots, A_n|C_j\} \cdot P(C_j)$$

Si consideramos (ingenuamente) los atributos como variables independientes, el primer término se simplifica:

$$P\{A_1, A_2, \dots, A_n|C_j\} = P\{A_1|C_j\} \cdot P\{A_2|C_j\} \cdots P\{A_n|C_j\}$$

Ejemplo

Como ejemplo vamos a utilizar un fragemento del conjunto de datos de enfermedades agudas y utilizaremos un clasificador Naïve Bayes para determinar si un paciente tienen una inflamación aguda de la vejiga.

Datos:

temp	nausea	dolor lumbar	necesidad constante	dolor al orinar	comezón en la uretra	infección
35.5	no	yes	no	no	no	no
36.0	no	yes	no	no	no	no
36.8	no	no	yes	yes	yes	yes
37.0	no	no	yes	yes	yes	yes

temp	nausea	dolor lumbar	necesidad constante	dolor al orinar	comezón en la uretra	infección
37.4	no	no	yes	no	no	yes
37.1	no	no	yes	no	no	yes
37.6	no	no	yes	yes	no	yes
37.8	no	no	yes	yes	yes	yes
38.0	no	yes	yes	no	yes	no
39.0	no	yes	yes	no	yes	no
40.4	yes	yes	no	yes	no	no
40.8	no	yes	yes	no	yes	no
41.5	yes	yes	no	yes	no	no
41.5	no	yes	yes	no	yes	no

Paciente:

temp	nausea	dolor lumbar	necesidad constante	dolor al orinar	comezón en la uretra	infección
36.6	no	no	yes	yes	yes	yes

Como primer paso vamos a calcular $P(C_j)$:

$P(C_j) = \frac{N}{N_c}$

$P('yes') = 6/14 = 0.429$

$P('no') = 8/14 = 0.571$

Para calcular los atributos discretos:

$$P(A_i|C_k) = \frac{|A_ik|}{N_Ck}$$

Donde,

- $|A_ik|$ es el número de registros con el atributo A_ik pertenecientes a la clase C_k
- N_Ck es el número de registros pertenecientes a la clase C_k

En la siguiente tabla tenemos las probabilidades de los atributos discretos:

clase	nausea	dolor lumbar	necesidad constante	dolor al orinar	comezón en la uretra
yes	0.14	0.57	0.71	0.43	0.5
no	0.86	0.43	0.29	0.57	0.5

Para el caso de datos continuos hay varias opciones @[tan2007introduction]:

- Se discretiza el rango creando particiones y asignando un solo valor a cada una.
- Se hace una división de dos vías a partir de un valor.
- Se estima la densidad de la probabilidad.
  - Se asume que los valores siguen una distribución normal.
  - Se utilizan los datos para calcular los parámetros de la distribución.
  - Una vez que se conoce la distribución se puede calcular la probabilidad condicional.

En este caso tenemos al atributo temperatura como atributo continuo.

clase	media	stdev	distribución normal x=36.3
yes	37.28	2.38	0.16
no	39.09	0.34	0

Incluso antes de multiplicar las probabilidades vemos que en el atributo «temperatura» la clase «no» tiene cero de probabilidad, por lo que la probabilidad de la clase «yes» será mayor.

$P(Paciente|No) = 0.86 * 0.43 * 0.29 * 0.57 * 0.5 * 0.16 = 0.0048$

$P(Paciente|Yes) = 0.14 * 0.57 * 0.71 * 0.43 * 0.5 * 0 = 0$

$P(Paciente|No)P(No) = 0.0027$

$P(Paciente|Yes)P(Yes) = 0$

Como P(Paciente|No)P(No) es mayor, **la clase para el registro Paciente es «yes»**

Redes Neuronales Artificiales

Las técnicas de clasificación que exploraremos en estos dos capítulos, las *redes neuronales artificiales* y las *máquinas de vectores de soporte* siguen una misma idea: encontrar funciones que definan un

frontera de decisión. Por ejemplo, en el caso de un clasificación binaria tendíamos una función que tome como entrada un vector de características y lo asigne a una de las clases. Para entender mejor esta idea vamos a empezar primero resolviendo un caso sencillo de clasificación. Los conjuntos de datos *linealmente separables* son aquellos que podríamos clasificar simplemente trazando una línea recta y diciendo de este lado es la clase A y del otro la clase B. Claro, lo de la línea recta solo sería posible si los datos estuvieran en un espacio de dos dimensiones, es por eso que de manera general hablamos de *hiperplanos* de $dimenson - (D - 1)$, con esto es 2D el hiperplano es de 1D (la recta) y en 3D un plano y así sucesivamente. Una manera de clasificar este tipo de conjuntos de datos es empleando funciones discriminantes.

Funciones Discriminantes

Un discriminante es una función que toma un vector n como entrada y le asigna una clase. Nos vamos a concentrar solamente en discriminántes lineales, aquellos para los que la superficie de desción es un hiperplano. También por el momento atacaremos problemas de clasificación binarios. Una función discriminante toma como entrada un vector de características:

$$y(\vec{x}) = \vec{w}^T\vec{x} + w_0$$

donde a $\vec{w}$ se le conoce como el *vector de pesos* y a w_0 como el *sesgo* (a $-w_0$ se le conoce también como el *umbral*). Si tenemos dos clases $C = \{c_1, c_2\}$, podemos decir que si $y(\vec{x} \geq 0)$ se le asigna al objeto la clase c_1 y de lo contrario la clase c_1. Veamos a la función discriminante gráficamente:

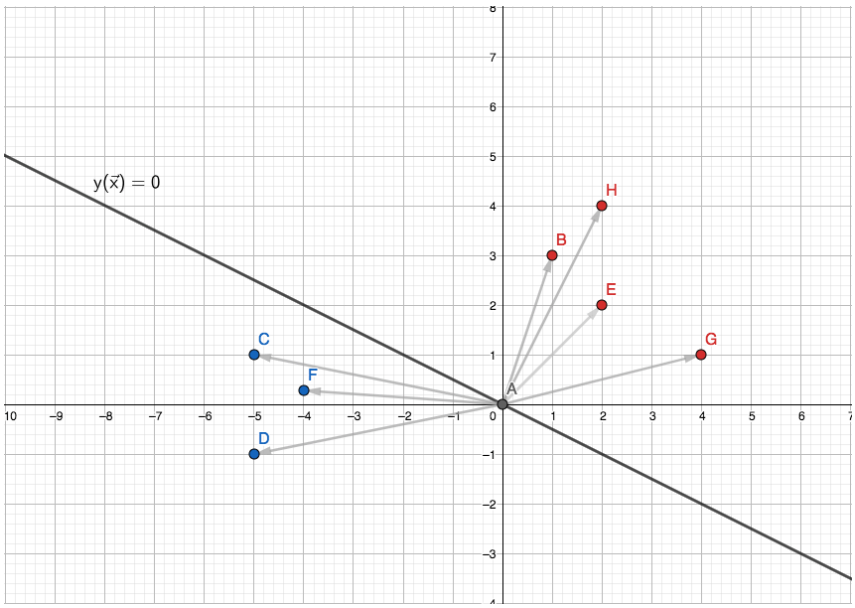


Figura 2: Objetos y función discriminante en 2D

En la gráfica vemos la representación vectorial del conjunto de datos, la clase a la que perteneces esta indicada por el color. Unos de los hiperplanos que discrimina a dichos objetos se muestra como $y(\vec{x}) = 0$. Veamos ahora como se establece este hiperplano en la figura siguiente:

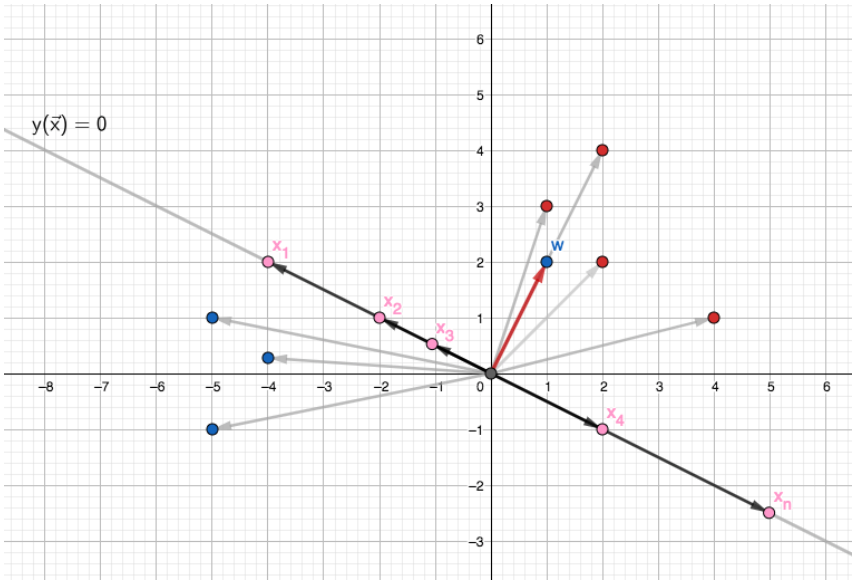


Figura 3: Función discriminante definida por $\vec{w} y vectores ortogonales$

El hiperplano se define por medio del vector de pesos $\vec{w}$ y todos aquellos vectores que son ortogonales

a él. Por ejemplo, en la gráfica $\vec{w} = (1, 2)$, si consideramos el vector de entrada $x_2 = (-2, 1)$, vemos que el producto punto entre ambos es igual a cero ya que son otogonales, lo mismo sería para $x_1, $x_3, …, $x_n y todos los vectores ortogonales a w. Para cambiar la inclinación del hiperplano solamente tendríamos que ajustar a los componentes de $\vec{w}$. Ahora evaluemos el vector $\vec{(x)} = (1, 3)$ en rojo, para esto solo tenemos que calcular de nuevo $\vec{(w)}^T\vec{(x)}$, vemos que el resultado es positivo y para $\vec{(x)} = (-5, 1)$ es negativo. De esta manera sin necesidad de mover del origen al hiperplano hemos establecido una función discriminante con solo encontrar un $\vec{w}$ apropiado. Para aquellos casos en los que el hiperplano debe moverse del origen, esto se ajusta con el sesgo w_0, como se muestra en la figura, donde el hiperplano, mostrado en azul, está fuera del origen.

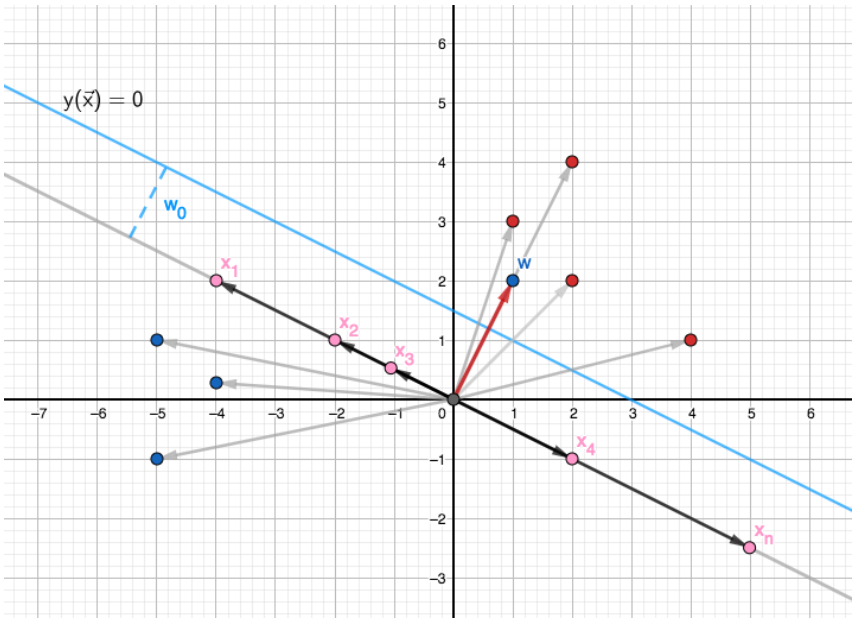


Figura 4: Función discriminante definida por $\vec{w}$, vectores ortogonales más un sesgo w_0, en azul

El objetivo del entrenamiento en este tipo de clasificadores es resolver un problema de optimización: encontrar los parámetros $\vec{w}$ y w_0 que minimicen el error de clasificación del conjunto de datos.

El perceptrón

Vamos a ver ahora un discriminante lineal de gran importancia histórica, el perceptrón de (**???**). El aunque perceptrón solo es capaz de clasificar conjuntos de datos linealmente separables, su desarrollo sentó las bases para las redes neuronales artificiales, tanto en las técnica como en la inspiración en un modelo simplificado de las neuronas biológicas. En la figura podemos ver un esquema de los componentes básicos del perceptrón.

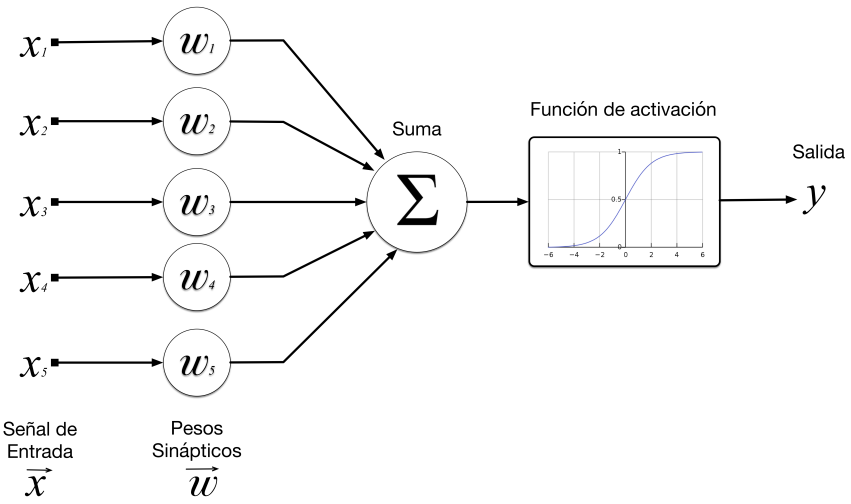


Figura 5: Componentes básicos del perceptrón

Vemos algunos componentes familiares de la función discriminante, la entrada es un vector x, en este caso cada componente se considera una señal, cada componente se multiplica por un peso y los resultados se suman, esto es el producto punto $\vec{x}^T\vec{w}$.El perceptrón agrega un componente adicional, el resultado de la suma, pasa por una función de activación, la cual decide que tanto se dispara el resultado. En el caso básico, la función de activación no lineal esta dada por

$$f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0 \end{cases}$$

Otro componente (**???**), el cual no se considera cuando se presenta el modelo de manera esquemática, el una transformación no lineal fija que se hace al vector de entrada de la forma $\phi(\vec{w})$. Por lo que el modelo de lineal generalizado tiene la forma

$$y(\vec{x}) = f(\vec{w}^T\phi(\vec{w}))$$

Aunque el *sesgo* no se visualiza en el esquema, ni en la función $f()$, normalmente se incluye como $\phi_0(\vec{w}) = 1$. Es importante considerar también la diferencia de la salida de $f()$, ahora toma el valor +1 para la clase c_1 y -1 para la clase c_2. Cuando camparemos el valor objetivo t (del inglés target) este también se debe especificar como $t\ epsilon\{-1, +1\}$. Esto nos permite tener solo una función que satisfacer $\vec{w}^T\phi(\vec{x_n})t_n > 0$, ya que si los signos son iguales el resultado será 1 y si son distintos será -1. Lo que se busca en todos los casos es minimizar el error, por lo que el criterio perceptron de error, es la suma de todos los objetos clasificados incorrectamente para determinado $\vec{w}$:

$$E_P(\vec{w}) = -\sum_{n\epsilon M} \vec{w}^T \phi(\vec{x_n}) t_n$$

Donde M es el conjunto de los objetos clasificados incorrectamente.

Algoritmo de Aprendizaje

Una vez definida la función objetivo podemos evaluar que tan bueno es cierto vector de pesos $\vec{w}$, esto nos sirve también para saber como ajustarlo utilizando alguna técnica optimización. Al preceso de ajustar los pesos le llamamos aprendizaje, ya que un algoritmo ajustará los pesos, (moverá a $\vec{w}$) de acuerdo al desempeño que se tenga con los datos de entrenamiento. El entrenamiento que consiste en clasificar los objetos, si se clasifica un objeto correctamente, pues no ajustamos nada, pero si se comete un error debemos ajustar a $\vec{w}$. Vamos a utilizar la técnica del gradiente descendiente estocástico (SGD) para ir ajustando los pesos de manera iterativa. Esto es posible ya que la función del error $E_P(\vec{w})$ es lineal y diferenciable. La actualización de los pesos estará dada por:

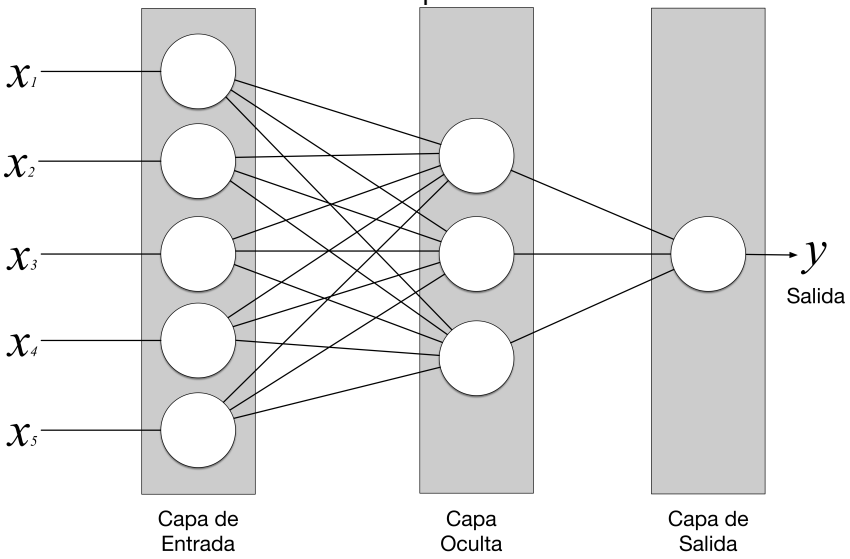
$$\vec{w}^{(\tau+1)} = \vec{w}^{(\tau)} - \eta \nabla E_P(\vec{w}) = \vec{w}^{(\tau)} + \eta \phi(x_n) t_n$$

Utilizamos al entero τ para indicar el avance en la iteración o el tiempo. Con (τ) nos referimos al vector actual y con $(\tau + 1)$ al vector en su nueva posición. La segunda expresión se refiere a la actualización propuesta por (SGD) la cual tiene el componente η que es la tasa de aprendizaje, esto nos indica que tan grande queremos que sea el movimiento del vector, también tenemos al gradiente ∇ que nos indicaría la dirección, y también el error del perceptrón E_P. En este caso la actualización se realiza con la tercera expresión, donde en caso de error de clasificación a $\vec{w}^{(\tau)}$ le sumamos o restamos (dado por t_n) el vector que se clasificó incorrectamente ⊠(x_n). En este caso η puede ser igual a 1. En caso de que los datos sean efectivamente separables linealmente, el algoritmo está garantizado que el algoritmo converge.

Red Neuronal Artificial Multicapa

Un perceptrón por si solo, es incapaz de clasificar datos como los mostrados en la figura, en este caso los datos modelan a una compuerta lógica XOR. Este problema si puede ser modelado con una red conocida como *perceptrón multi capas* (Multi Layer Perceptron MLP). Una MLP consiste normalmente de mínimo tres capas como vemos en la figura. Digamos que iniciamos con un perceptrón pero en lugar tomar la salida, esta alimenta (feeds-forward) a otro conjunto de perceptrones en una nueva *capa oculta*, puede haber más capas ocultas y cada capa oculta puede tener

distinto número de nodos. Las capas ocultas también tienen funciones de activación no lineales.



Al igual que con el perceptrón, el aprendizaje de los pesos es lo que hace la magia. En este caso el aprendizaje es más complicado, pues la función del error no es lineal como antes. El algoritmo de aprendizaje más utilizado para este tipo de redes es *backpropagation* utilizando también una técnica basada en el gradiente descendente.

Support Vector Machines

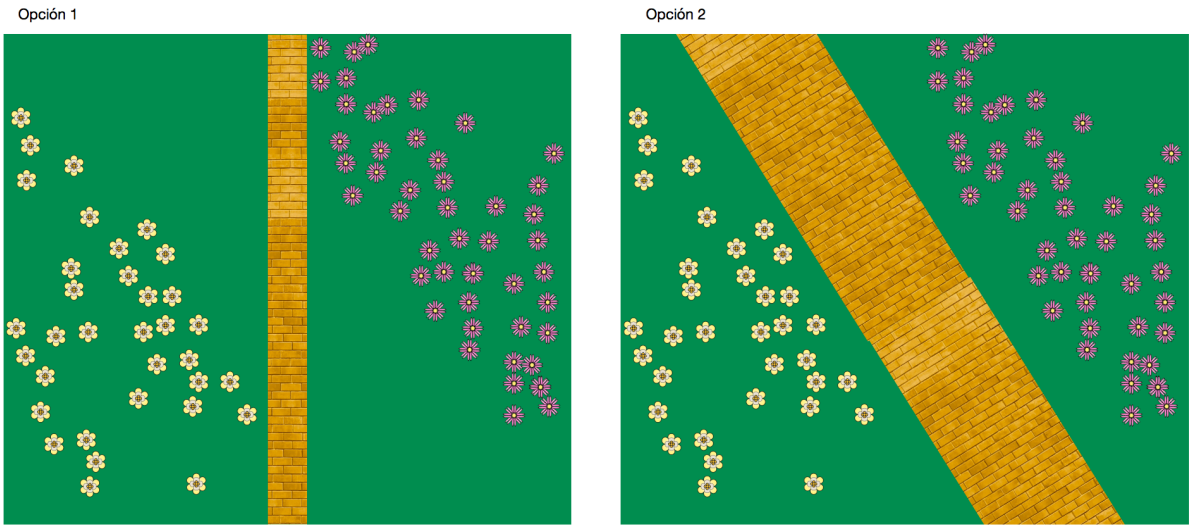


Figura 6: Dos opciones para el camino que pasa por el jardín

El algoritmo de Support Vector Machines (SVM) también es un algoritmo que utiliza una función discriminante lineal para hacer clasificación. Lo que distingue a SVM de otros algoritmos es una restricción adicional que pone a la función discriminante. La función debe estar en una posición tal, que maximice el margen de esta función. El concepto de máximo margen lo podemos entender con un ejemplo. Digamos que yo tengo un jardin dos grupos de flores linealmente separados, como se muestra en la figura. Justo en la separación de ambos grupos y quiero construir un camino de ladrillos amarillos para pasar por el jardín. La única restricción que tengo es que no puedo tocar ninguna de las flores. Me presentan dos opciones y ambas cumplen con lo dicho. ¿Qué diferencia hay entre ellas?

1. La primera opción nos brinda un camino muy angosto aunque cumple su cometido. Digamos que esta opción tiene un margen algo angosto.
2. La segunda opción también cumple, pero como está inclinada y debido a la posición de las flores nos permite ensanchar el camino. Digamos ahora que esta opción tiene un margen mayor.

Si en ambos casos trazo una recta que paste justo a la mitad de los caminos, esta recta sería la función discriminante y el camino en sí sería el margen.

Bueno, SVM sugiere eso, buscar una función en tal posición que maximice dicho margen. Ni la función discriminante, ni el perceptrón se ponen exigentes a momento de ajustar los pesos, se detienen en cuanto clasifican correctamente a los objetos. Ahora, la ventaja de maximizar el margen obliga al algoritmo a explorar más, pero al cumplir con el objetivo tendrá posiblemente un mejor clasificador. La razón tiene que ver con los datos que no se han visto, los datos de prueba. Un margen mayor, es más general, casi por definición no estará sobre ajustado. De hecho en el ejemplo, podemos decir que la opción con mayor margen refleja o aprendió mejor la manera en que crecen las flores en el jardín.

El caso linealmente separable

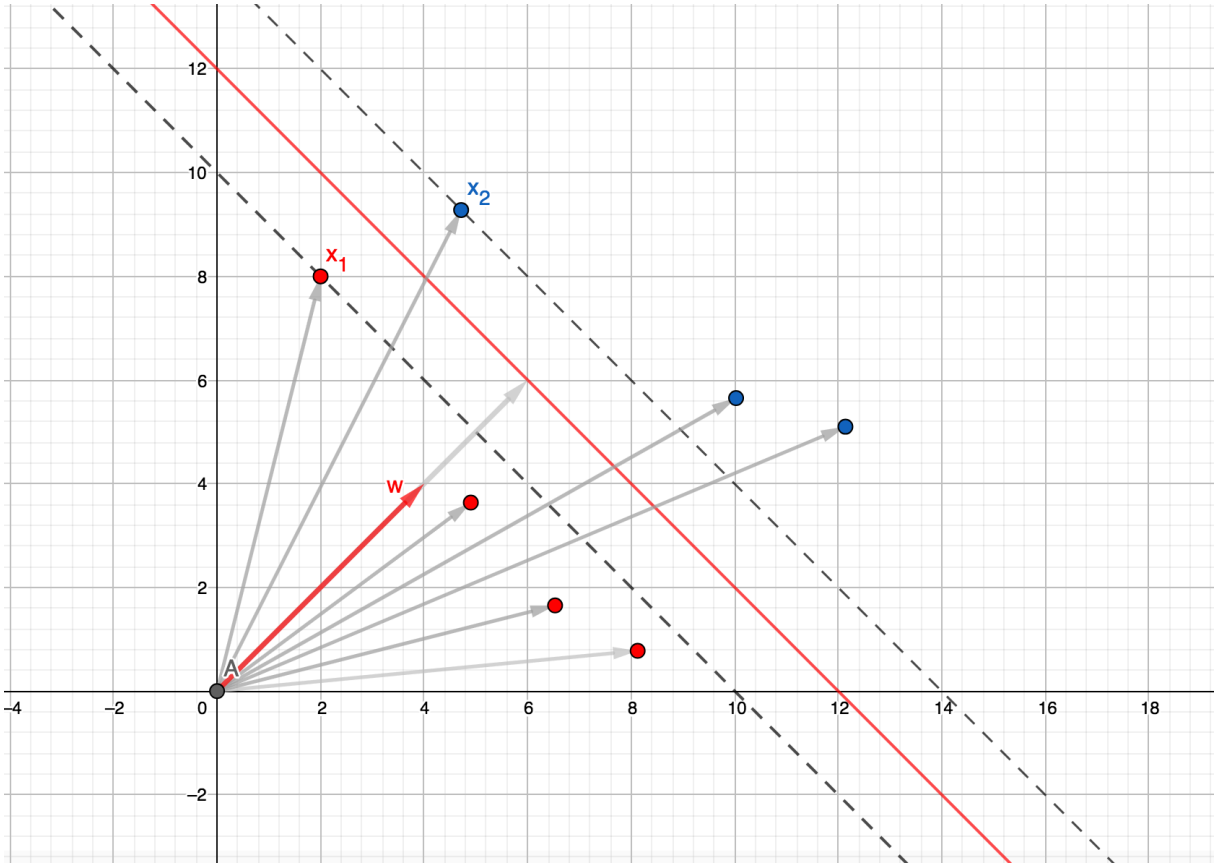


Figura 7: Vectores de soporte

Vamos a considerar una función discriminante lineal, recordemos que se define como:

$$y(\vec{x}) = \vec{w}^T\vec{x} + w_0$$

En la literatura de SVM (**???**) al sesgo se le conoce como b (por sesgo en inglés *bias*), esto puede crear confusión por que también se utiliza un vector $\vec{x_0}$. Para esta sección vamos a renombrar entonces a la función discriminante como (también se asumirá la transpuesta de $\vec{w}$):

$$y(\vec{x}) = \vec{w} \cdot \vec{x} + b$$

Esta frontera se muestra en rojo en la figura, y vemos al vector w ortogonal a la hiperplano $y(\vec{x}) = 0$. Como el hiperplano no está en el origen sabemos que el valor de b no es cero. El conjunto de datos esta representado por vectores de dos clases: rojos y azules. Estos objetos están separados linealmente.

Fíjate como en este caso el hiperplano tiene un margen máximo con respecto a los vectores $\vec{x_1}$ y $\vec{x_2}$, rojo y azul respectivamente. Estos son vectores de los datos de entrenamiento, correctamente clasificados, cercanos a la frontera y nos ayudan a encontrar el hiperplano con máximo margen, si, estos son los vectores de soporte. Nos apoyamos en ellos para encontrar al hiperplano. ¿Cómo nos ayudan?. Bueno, si sabemos que estos dos vectores son los más cercanos a la frontera y están clasificados correctamente, es solo cuestión de encontrar al hiperplano que este más lejos de los dos o el que esté en la mitad es lo mismo. Ocupamos ver como podemos medir la distancia entre ellos y también como expresar a ambos hiperplanos. Los hiperplanos se expresan de ls siguiente manera:

$$\vec{w}^T\vec{x_2} + b = 1\vec{w}^T\vec{x_1} + b = -1$$

Si evaluamos los vectores $\vec{x_1}$ y $\vec{x_2}$, para los valores de w y b actuales, estos tendrán el signo correcto, pero no los valores 1 y -1. ¿Qué pasaría si multiplicamos a w o a b por un escalar?. Se generaría un hiperplano paralelo a $\vec{w}^T\vec{x} + b$. Entonces solo es cuestión de escalar ambos parámetros de la frontera de decisión para que encontrar los hiperplanos paralelos. El margen está dado por:

$$\frac{2}{||w||} = \frac{2}{\sqrt{w \cdot w}}.$$

Decimos entonces que el hiperplano óptimo es aquel que minimiza a $w \cdot w$ bajo la restricción general:

$$y_i(\vec{w}^T\vec{x} + b) \geq 1, i = 1, ..., N$$

Este problema de optimización que se resuelve con técnicas de programación cuadrática. Una vez encontrados w y b el clasificador debe incluir los multiplicadores de Lagrange.

Métodos Ensamble

Así como las decisiones difíciles reúnen a grupos de expertos en cierto padecimiento médico, en ocasiones podemos reunir a varios expertos para ayudarnos a clasificar los datos. Para nuestro caso cada nuevo modelo puede ser visto un experto distinto. A estos métodos se les conoce como métodos ensemble.

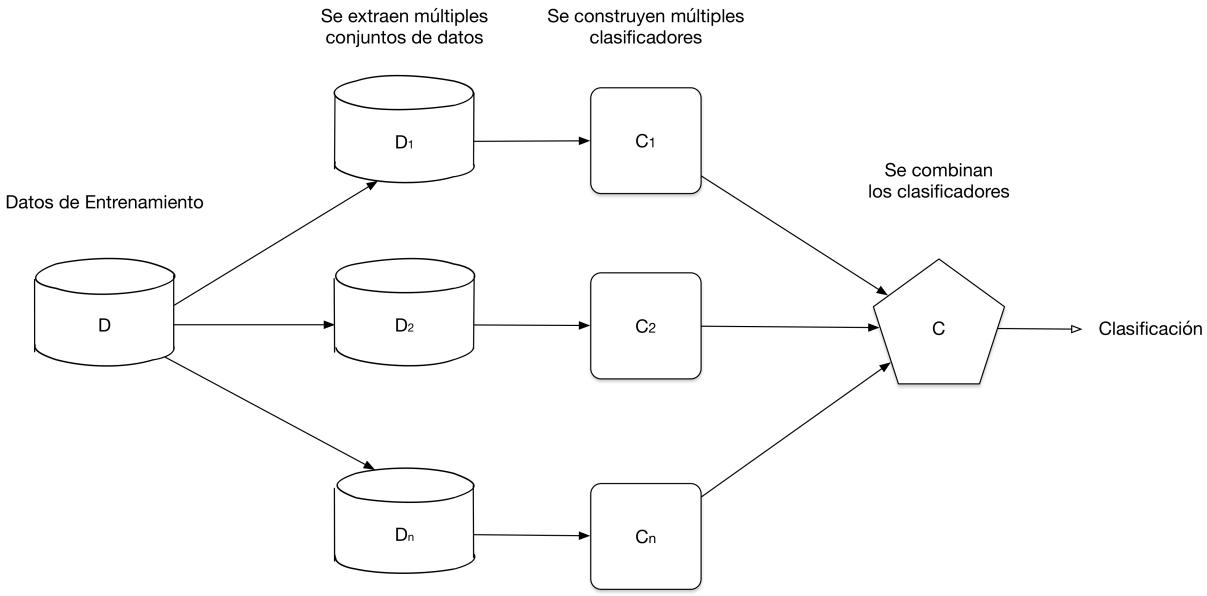


Figura 8: Esquema general del aprendizaje ensemble