

RETROPROPAGACIÓN DEL ERROR

Se utiliza en redes neuronales → Funciones base

Se pueden utilizar diferentes funciones de salida, lo que dará distintos modelos de redes neuronales.

- Unidades sigmoides: $B_i(x, w) = \frac{1}{1 + e^{-\sum_{j=1}^K (w_{j0} + w_{ji} \cdot x_j)}}$ * Para Clasificación Multiclas en salida (US)
- Unidades de Base Radial: $B_i(x_i; (c_i, r_i)) = e^{-\frac{\|x - c_i\|^2}{2r_i^2}}$
- Unidades producto: $B_i(x, w) = \prod_{i=1}^K x^{w_i}$

APRENDIZAJE MEDIANTE ALGORITMOS DE GRADIENTE DESCENDENTE

Arquitectura → Una red feedforward de al menos una capa oculta y con nodos salidas no lineales.

Una función de transferencia diferenciable (sigmoide más común) — Función de transferencia.

Aprendizaje → supervisado, control de error, regla delta generalizada.

Algoritmo retropropagación del error → regla de adaptación de los pesos de la red (aprendizaje mediante un método de gradiente descendente)

Pesos: $w_{2,1}^{(1,0)}$ Peso desde el nodo 1 de capa de entrada (0) a nodo 2 de capa oculta (1)

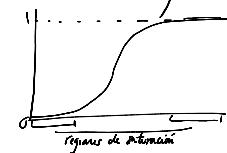
Notaciones →

- P patrones, K salidas, n variables.
- Entrada: (x_1, x_2, \dots, x_n)
- Matrices de entrenamiento: $(x_p, d_p) \quad p = 1, \dots, P$
- Matriz de salida: (o_{p1}, \dots, o_{pn})
- Patrón de entrada: $(x_p, \dots, o_{p,n})$

ALGORITMO DE RETRO PROPAGACIÓN

SIGMOIDE

- Como el perceptrón, la primera parte de una unidad sigmoidal computa una combinación lineal de sus entradas ($o = w \cdot x$)
- Luego computa su salida con la función anterior *
- Esta segunda función a menudo se la reconoce como una función de activación (por pasar de un dominio muy grande) a uno pequeño en la salida
- Tiene la propiedad de que su derivada se expresa con facilidad en su salida, lo que hace la descripción de RE más compacta
- $S'(x) = S(x)(1 - S(x))$
- Cuando $|x|$ es suficientemente grande, se muerde hacia una de las regiones de saturación → umbral o tipo rampa



HACIA DELANTE

Se aplica un vector x de entradas a los nodos de la capa oculta → Si calcula el vector de salida de la capa oculta $-x(1)$ → Si calcula el vector de salida de la capa de salida.

$$x_i^{(1)} = S(\text{net}_i^{(1)}) = S\left(\sum_j w_{ij}^{(1,0)} x_j^{(1)}\right)$$

$$o_k = S(\text{net}_k^{(2)}) = S\left(\sum_j w_{kj}^{(2,1)} x_j^{(1)}\right)$$

Se dice que la red es una transformación de la entrada x a la salida o .

Objetivo del aprendizaje

Reducir la suma de errores al cuadrado para el conjunto de patrones de entrenamiento

HACIA DETRÁS

Objetivo del algoritmo BP

- Cambiar los pesos $w^{(2,1)}$ (capa oculta a salida) usando la regla delta en una sola capa de la red usando el MSE
- Cambiar los pesos $w^{(1,0)}$ (capa oculta a salida) usando la regla delta en otra sola capa de la red usando el MSE.
- La regla delta no se aplica como de salida a salida porque no sabemos los valores deseados de la capa oculta.

La solución: Propagar los errores de capa de salida hacia las nodos de capa oculta implicando los cambios de los pesos $w^{(1,0)}$ → Nuevo Delta

- Cómo calcular los errores en capa oculta
- Puede seguir retropropagándose si la red tiene más de una capa oculta.

REGLA DELTA

Considerando un método de aprendizaje secuencial el error cometido es:

$$E = \sum (d_k - o_k)^2$$

- Se cambian los pesos mediante un algoritmo de gradiente descendente.

• Para los pesos de oculto-salida: $\Delta w_{kj}^{(2,1)} \propto \left(-\frac{\partial E}{\partial w_{kj}^{(2,1)}} \right)$

• Para los pesos de entrada-oculta: $\Delta w_{ji}^{(1,0)} \propto \left(-\frac{\partial E}{\partial w_{ji}^{(1,0)}} \right)$

Derivación mediante los pesos oculto-salida

- Puesto que E es una función de $d_k - o_k$ y $d_k - o_k$ es una función de $\text{net}_k^{(2)}$, y $\text{net}_k^{(2)}$ es una función de $w_{kj}^{(2,1)}$ por la regla de la cadena.

$$\frac{\partial E}{\partial w_{kj}^{(2,1)}} = \frac{\partial E}{\partial (d_k - o_k)} \cdot \frac{\partial (d_k - o_k)}{\partial \text{net}_k^{(2)}} \frac{\partial \text{net}_k^{(2)}}{\partial w_{kj}^{(2,1)}} = -2(d_k - o_k) S'(\text{net}_k^{(2)}) / x_j$$

Derivación mediante los pesos entrada-oculta

- Siendo: $E = \sum (d_k - o_k)^2 \Big| o_k = S(\text{net}_k^{(2)}) \Big| \text{net}_k^{(2)} = \sum_j x_j^{(1)} w_{kj}^{(2)} \Big| x_j^{(1)} = S(\text{net}_j^{(1)}) \Big| \text{net}_j^{(1)} = \sum_i x_i w_{ji}^{(1,0)}$

$$\begin{aligned} \frac{\partial E}{\partial w_{ji}^{(1,0)}} &= \frac{\partial E}{\partial (d_k - o_k)} \frac{\partial (d_k - o_k)}{\partial o_k} \frac{\partial o_k}{\partial x_j^{(1)}} \frac{\partial x_j^{(1)}}{\partial \text{net}_j^{(1)}} \frac{\partial \text{net}_j^{(1)}}{\partial w_{ji}^{(1,0)}} = \\ &= -2(d_k - o_k) (-1) \frac{\partial (S(\text{net}_k^{(2)}))}{\partial \text{net}_k^{(2)}} \frac{\partial (\text{net}_k^{(2)})}{\partial x_j} \frac{\partial x_j^{(1)}}{\partial \text{net}_j^{(1)}} \frac{\partial \text{net}_j^{(1)}}{\partial w_{ji}^{(1,0)}} = \\ &= 2(d_k - o_k) S'(\text{net}_k^{(2)}) w_{kj}^{(2,1)} \cdot S'(\text{net}_j^{(1)}) \cdot x_i \end{aligned}$$

Hay que aplicar la regla de la cadena a o_k respecto a $\text{net}_k^{(2)}$

Adaptación de los pesos

• Para cada valor k (salida) $w_k = w_k + \Delta w_k$

• Oculta-Salida $\rightarrow \Delta w_{kj}^{(2,1)} = \eta \cdot \delta_k \cdot x_j^{(1)} \quad | \text{ Siendo } \delta_k = (d_k - o_k) S'(\text{net}_k^{(2)})$

• Entrada-Oculta $\rightarrow \Delta w_{ji}^{(1,0)} = \eta \cdot \mu_j \cdot x_i^{(0)} \quad | \text{ Siendo } \mu_j = \left(\sum_k \delta_k w_{kj}^{(2,1)} \right) S'(\text{net}_j^{(1)})$

- El error de una determinada neurona se calcula sumando los términos de error de las neuronas influenciadas por ella, ponderados con los pesos que las unen. Este peso señala el grado de "responsabilidad" de la primera sobre el error de la segunda.

CLASIFICACIÓN

- Dos clases: 1 nodo de salida.
- J clases: codificación "1 de J " representada como $(0, \dots, 0, 1, 0, \dots, 0)$
- Con una función sigmoidal en la salida, los nodos de dicha capa nunca darán $1 \circ 0$, sino $1 - \epsilon \circ \epsilon$.
- La reducción del error es más lenta cuando se encuentra en las regiones de saturación (cuando $\epsilon \approx 0$)
- Para un aprendizaje más rápido damos una cota de error ϵ , y definimos el error $b_{jkx} = 0$ si $|d_{jkx} - o_{jkx}| \leq \epsilon$

FORTALEZAS

- Gran poder de representación
 - Algunas funciones L_2 pueden ser representadas mediante una red MLP (base sigmoidal) entornada con BP
 - Muchas de estas funciones se pueden aproximar mediante un algoritmo de aprendizaje BP (de gradiente descendente)
- Gran aplicabilidad del aprendizaje BP
 - Sólo necesita un buen conjunto de patrones de entrenamiento
 - No necesita conocimientos a priori del propio dominio del problema
 - Tolerancia al ruido y a datos perdidos en el conjunto de entrenamiento (lure degeneración)
 - Fácil de implementar el código del algoritmo
- Buen poder de generalización (A menudo)

DEBILIDADES

- A veces tarda mucho en converger
 - El entrenamiento con funciones complejas a menudo necesita muchas épocas y tienen que pasar todos los patrones.
- La red es una caja negra
 - Nos da una transformación no lineal de un patrón de entrada a una determinada salida, pero no el porqué de dicha transformación. (No sabemos qué pasa en el interior)
 - No suele proporcionar una relación causa-efecto. → Se dice a que los nodos ocultos y los pesos aprendidos no tienen una clara semántica.
 - No existe una teoría bien fundamentada que exprese la calidad de BP.
- Problema con la aproximación de gradiente descendiente.
 - Solo garantiza que reduce el error a un mínimo local (predice que $E \neq 0$)
 - No puede escapar del mínimo local
 - No todo función representable puede ser aprendida.
 - Si la función de error presenta muchas valles y límas se quedará en un mínimo local

PROBLEMAS REMEDIOS

- Probar redes con diferentes números de capas y/o nodos ocultos - diferentes funciones de error.
- Probar diferentes pesos iniciales para BP
- Forzar una salida de un mínimo local mediante una perturbación aleatoria - ensamblaje suavecido.

GENERALIZACIÓN

Problema con el sobreentrenamiento.

- Soluciones:
- Más y mejores muestras
 - Redes lo más pequeñas posibles
 - Una cota de error mayor - forzar una terminación más rápida (menos épocas)
 - Introduciendo ruido en las muestras.

VALIDACIÓN CRUZADA

- Tomar aproximadamente el 10% del conjunto de entrenamiento.
- Chequear periódicamente los errores de "generalización" sobre este conjunto
- Detener el entrenamiento cuando el error de validación comienza a aumentar.

PARÁISISIS DE LA RED (SIGMOIDES)

Se produce cuando una entrada entra en la región de saturación porque uno de los pesos de entrada se hace muy grande durante el aprendizaje. \rightarrow Los pesos dejan de cambiar sin importar lo que se haga.

Remedios:

- Usar funciones de activación no saturadas (en la capa de salida)

- Normalizar periódicamente todos los pesos $w_{k,j}^* = w_{k,j} / \|w_k\|_2$

Como $S'(x) = S(x) \cdot (1-S(x))$;
 $S'(x) \rightarrow 0$ cuando $x \rightarrow \pm \infty$

PARÁMETROS DE BD

- El aprendizaje (precisión, velocidad de convergencia, capacidad de generalización) dependen de:
 - Pesos iniciales del algoritmo BP
 - Coeficiente de aprendizaje (η)
 - n' de capas ocultas
 - n' de nodos en capa oculta
 - tipo de función de transferencia.
- Si se usa la validación cruzada con un K-fold y una estructura fija malla donde se van probando diferentes valores de cada uno de los parámetros.
- Se intenta determinar los parámetros a la vez que un buen ajustamiento para una buena generalización.

PESOS INICIALES Y RIGOS

- Asignación aleatoria de los pesos iniciales para todos los pesos de capa oculta a capa de salida.
- Normalizar los pesos con respecto al n' nodos ocultos y n de nodos de entrada

TOPOLOGÍA DE LA RED

- Teóricamente una sola capa oculta (con muchos nodos) debe ser capaz de representar cualquier función L_2

ANÁLISIS DISCRIMINANTE LINEAL

Teoría de decisión Bayesiana: la mejor hipótesis es la más probable

Probabilidades a priori.

- Tenemos el espacio de hipótesis (h) y el de observaciones (D)

$P(h) \rightarrow$ Refleja lo que sabemos sobre las oportunidades de h sin tener ninguna observación (experimento)

$P(D) \rightarrow$ Refleja la probabilidad de obtener la observación D sin considerar la hipótesis

Probabilidades a posteriori

$P(D|h) \rightarrow$ Probabilidad de observar D cuando se da la hipótesis h

$P(h|D) \rightarrow$ Probabilidad de que se cumpla h cuando observamos D .

TEOREMA DE BAYES

- Calcular las probabilidades a posteriori \rightarrow si obtenemos tras realizar observaciones o ejemplos de entrenamiento.

$$P(h_i|D) = \frac{P(h_i \wedge D)}{P(D)} = \frac{P(h_i) \cdot P(D|h_i)}{P(D)} = \frac{P(h_i) \cdot P(D|h_i)}{\sum_{j=1}^J P(h_j) \cdot P(D|h_j)} \quad | \quad P(D) \neq 0$$

Decisor máximo "a posteriori": MAP

$$h_{MAP} = \arg \max_{h \in H} P(h|D) = \arg \max_{h \in H} \frac{P(h) P(D|h)}{P(D)} = \arg \max_{h \in H} P(h) P(D|h)$$

Decisor máxima verosimilitud: ML

Asume todas las probabilidades a priori equiparables: $h_{ML} = \arg \max_{h \in H} P(D|h)$

Ambas comparten las probabilidades a posteriori de una hipótesis dada una observación.
 Se escoge la que más probabilidad tenga en ambas casos.

CLASIFICADOR MAP DE FUERZA BRUTA

Teniendo el decisor máximo MAP y un problema binario:

Función discriminante: $g(D) = P(h_1)P(D|h_1) - P(h_2)P(D|h_2)$

Clasificador:

$$\begin{cases} h_1 \text{ (Clase 1)} & \text{Si } g(x) \geq 0 \\ h_2 \text{ (Clase 2)} & \text{Si } g(x) < 0 \end{cases}$$

Errores de clasificación

Erros de tipo 1 y errores de tipo 2

$$P(E) = P(x \in R_2, C_1) + P(x \in R_1, C_2) = \int_{R_2} P(x|C_1) P(C_1) dx + \int_{R_1} P(x|C_2) P(C_2) dx$$

ANÁLISIS DISCRIMINANTE

Si sabemos las dos distribuciones de x (las clases) de experiencias pasadas, podemos definir una curva que sirva de límite entre ambas distribuciones: $g(x_1, x_2) = 0$

A esta curva se le conoce como función discriminante, ya que divide el espacio muestral en dos regiones y es capaz de clasificar un patrón sin etiqueta de clase. $| g(x_1, x_2) > 0 \text{ ó } g(x_1, x_2) < 0 |$

Lineal

Se trabaja con un conjunto de datos linearmente separable.

Se clasifica mediante una transformación lineal, realizada con un entrenamiento

Encontrar una función discriminante apropiada, estudiando las características de la distribución de x

No lineal

Se trabaja con un conjunto de datos que no son linearmente separables

Se realiza una transformación a una dimensión mayor donde si lo sean (SVM)

OBJETIVO

Proyectar los datos minimizando la distancia entre los objetos de una misma clase y maximizando la distancia entre los objetos de distintas clases.

Tiene en cuenta todos los datos y sus distribuciones.

REGLA DE DECISIÓN DE BAYES

Frontera de decisión: $g(x_1) = g(x_2)$

Dadas dos clases C_1 y C_2 , sus funciones discriminantes serán:

$$g_i(x) = P(C_i) P(x|C_i); i=1,2 \xrightarrow[\text{tomando logaritmos}]{\text{y}} \ln g_i(x) = \ln P(C_i) + \ln P(x|C_i)$$

Regla de decisión

$\left\{ \begin{array}{l} \text{Si } g_1(x) - g_2(x) > 0 \text{ entonces } x \in C_1 \\ \text{Si } g_2(x) - g_1(x) > 0 \text{ entonces } x \in C_2 \end{array} \right.$ Conocido en que	$P(C_1)P(x C_1) - P(C_2)P(x C_2) > 0 \rightarrow \frac{P(x C_1)}{P(x C_2)} - \frac{P(C_2)}{P(C_1)} > 0$ razón de verosimilitudes	 Sopete a cm - ln
--	---	----------------------

Función de entropía

Aplicar $-\ln$ a la razón de verosimilitudes $\rightarrow H(x) = -I(x) = -\ln P(x|C_1) + \ln P(x|C_2)$

Si las probabilidades a priori son iguales (máxima verosimilitud), entonces $\ln \frac{P(C_2)}{P(C_1)} = 0$;

Por lo que $H(x)$ sería la función discriminante

$$\left\{ \begin{array}{l} \text{Si } -\ln P(x|C_1) + \ln P(x|C_2) < 0 \text{ entonces } x \in C_1 \\ \text{Si } -\ln P(x|C_1) + \ln P(x|C_2) > 0 \text{ entonces } x \in C_2 \end{array} \right.$$

DISTRIBUCIONES DISCRETAS

- Cada ejemplo x debe aparecer en C_i un número suficientemente grande de veces como para tener estadísticas significativas
- Si la dimensión de x crece, el número de posibles valores de x lo hace exponencialmente, lo que hace el problema intractable
- Si un suceso a clasificar no se ha dado con anterioridad, no se puede obtener su probabilidad a posteriori

MODELOS DE REDES NEURONALES

BASE RADIAL

- Su salida depende de la distancia del vector de entrada a un vector almacenado dentro (centroide)
- Tienen una capa de entrada, una oculta y una de salida.
- En la oculta los neuronas usan funciones de activación RBF de cambian linealmente para la salida

$$B_j(x; (c_j, r_j)) = e^{-\frac{\|x - c_j\|^2}{2r_j^2}}$$

c_j - centroides
 r_j - radios

• Cada neurona de la capa oculta tiene su campo de atracción.

• Si los regresan la salida es lineal
• Si los regresan la salida es con una función softmax

• Si los patrones de entrenamiento se distribuyen de forma no homogénea, μ_j y σ_j se calcularán realizando un análisis cluster

• Si busca minimizar la suma ponderada de cuadrados de las distancias de cada patrón x_i a su centroide μ_j :

RBFs

- Separar clases mediante hiperesferas
- Usa aprendizaje localizado y aprende más rápido
- Una sola capa oculta, aunque solo necesitan más neuronas "en el dominio"
- Capa oculta es no lineal - salida lineal
- La función de cada neurona RBF calcula la distancia entre el vector de entrada y el centro de esa unidad.

MLPs

- Separa mediante funciones lineales o hiperplanos
- Usa aprendizaje distribuido
- Tiene una o más capas ocultas
- Oculta y salida con funciones no lineales o no son que usamos softmax
- La función de cada neurona de la oculta calcula el producto interno del vector entrada y del vector de pesos sinápticos de dicha neurona.

Clasificación Binaria. Se puede usar una función softmax para ello.

ARQUITECTURA

- Solo se necesita aprender los pesos β_{ij} (oculta - salida).
- Se pueden determinar con cualquier método iterativo, pero al ser la función de aproximación lineal, se puede calcular directamente usando matrices asociadas a métodos lineales de mínimos cuadrados.

Con N patrones y q neuronas RBF, tendríamos Neuronas lineales con N incógnitas

$$y_i = \sum_{k=1}^q \beta_{ik} \cdot \varphi_j(\|x_i - c_k\|) \text{ para } i=1, \dots, N$$

$$(P^T P)^{-1} P^T$$

$$\text{En forma matricial: } P_{N \times q} \cdot \beta_{q \times 1} = y_{N \times 1}, \text{ donde } P = (p_{ik})$$

$$\left\{ \begin{array}{l} N = q \rightarrow \beta = P^{-1} \cdot y \quad (\text{a } P \text{ de rango completo}) \\ N > q \rightarrow \beta = P^+ \cdot y \quad (\text{método Pseudo inversa Moore Penrose}) \\ N < q \rightarrow \text{Demasiadas soluciones. Usar heurísticas.} \end{array} \right.$$

Si N es grande, el cálculo de la pseudo inversa sería inexacto. Si la solución es analítica, se usaría el gradiente descendente.

VENTAJAS

• Solo habrá que ajustar los pesos localmente si hay nuevos patrones train

• Pesos óptimos en tiempo polinómico

• Los regresos no recordados por RBF pueden ser identificados como $\beta_{ik} = 0$

DESVANTAJAS

• El número de neuronas aumenta exponencialmente con la dimensión de la entrada

• No puede extrapolar (No hay control fuera del entorno de los patrones train - RBFs son locales)

REDES DE HOPFIELD

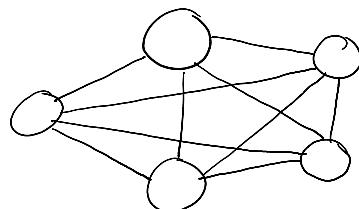
- Buscan simular las conexiones neuronales para resolver un cálculo.

- Es un tipo de red monocapa recurrente, cuyos valores de salida son remitidos a la entrada de una manera no dirigida.
- Aprendizaje no supervisado en el que agrupan patrones por similaridad.
- Conjunto de N neuronas conectadas con pesos simétricos. Sin neuronas especiales de entrada/salida.
- La activación de una neurona es un valor binario decidido por el signo de la suma ponderada de las conexiones a sí misma.
- Un valor umbral de cada neurona que determina si se activa
- Una neurona activa, activa todas las neuronas a las que está conectada con un peso positivo
- La entrada se aplica a todas las neuronas a la vez, cada una con su salida. - Para el cálculo solo se tienen en cuenta las conexiones activas.
- Se repite el proceso hasta que no hay cambios de activación (estado estable)

CARACTERÍSTICAS

Actúan como memorias autoasociativas.

Todas las neuronas están conectadas entre sí.



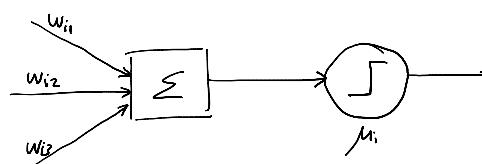
- Pesos simétricos $\rightarrow w_{ij} = w_{ji}$ y $w_{ii} = 0$

- Cambio asincrónico en las unidades

- En cada etapa se elige una unidad al azar

- Sea si el estado de la neurona i : $s_i = 1$ si $\sum_j w_{ij} s_j \geq \Theta_i$ para todo j activo conectado a i

$s_i = -1$ en otro caso



umbral, normalmente 0

| unas entradas binarias y una salida (0-1)
pesos sinápticos w_{ij}
umbral Θ_i

FUNCIÓN DE ENERGÍA

Se define la función de energía a minimizar: $E = -\frac{1}{2} \sum_i \sum_j w_{ij} s_i s_j + \sum_i s_i \Theta_i$

Hay cambio en E . $\left\{ \begin{array}{l} \text{Si } s_i \text{ es inicialmente inactiva } (-1) \text{ y } \sum_j w_{ij} s_j > \Theta_i, \text{ entonces } s_i \text{ se convierte en activa } (+1) \\ \text{Si } s_i \text{ es inicialmente activa } (+1) \text{ y } \sum_j w_{ij} s_j < \Theta_i, \text{ entonces } s_i \text{ se convierte en inactiva } (-1) \end{array} \right.$

CÁLCULO DE LOS PESOS

La red converge a un mínimo local que almacena diferentes patrones.

Almacena M vectores N -dimensionales de patrones, siendo M el n.º de patrones y N el n.º de nodos

$$w_{ij} = \begin{cases} \frac{1}{N} \sum_{m=1}^M x_{mj} x_{mi} & \text{para todo } j \neq i, \text{ donde } i, j = 1, \dots, N \\ 0 & \text{para } j = i \end{cases} \quad \left| \begin{array}{l} W = \frac{1}{N} \sum_{m=1}^M x_m x_m^T \quad (\text{matricial}) \\ \text{traspose} \end{array} \right.$$

COMPUTACIÓN DE PESOS

vectores de dimensión N fijos - señalan las unidades activas e inactivas

$$\text{matriz de pesos} \leftarrow W = \sum_i x_i (x_i)^T - N I_N, \text{ para } i = 1, \dots, M$$

matriz identidad $N \times N$

Cambio sincrónico \rightarrow Comprobar el cambio de todas las neuronas al mult. matricial para ver si es estable o no.

Cambio asincrónico \rightarrow Comprobar el cambio neurona a neurona para ver si es estable ese estado o no (función de energía)

SVMs

Objetivo

Dado un conjunto de datos de datos de entrenamiento, construir un hiperplano "w" como separador de decisión, de tal forma que la separación de las dos clases sea máxima.

(Clasificadores) / polinomial
no lineales } - SVMs

CASIFICADOR POLINOMIAL

La forma del clasificador lineal es:

Tienen una forma común

$$g(x) = w_0 + w^T x = w_0 + w_1 x_1 + w_2 x_2 + \dots + w_d x_d$$

$$g(x) = w_0 + w_1 x_1 + w_2 x_2 + \dots + w_d x_d + w_{11} x_1^2 + w_{12} x_1 x_2 + \dots$$

) Introducimos términos de grado 2.

y se puede seguir hasta cualquier grado, lo que nos lleva a tener un gran número de coeficientes

$g(x) = w_0 \phi_0(x) + w_1 \phi_1(x) + w_2 \phi_2(x)$ Función discriminante lineal generalizada (FDLG)

$g(x) = \sum_{i=0}^d w_i \phi_i(x)$, donde $\phi_i(x) = 1$. $\phi_i(x)$ puede ser cualquier tipo de función. La idea es descomponer una función compleja en una suma de funciones simples.

FDLG (caso del XOR 2 dimensiones a 4)

1º Transformar los datos de entrada con las funciones ϕ a un nuevo espacio de características

2º Aplicar a los datos transformados alguno de los métodos del Análisis Discriminante lineal

TEOREMA DE COVER (Probable que sean linearmente separables en d dimensiones - si tienen algoritmos de entrenamiento para determinar los pesos)

La probabilidad de que dos clases sean linearmente separables se aproxima a 1 cuando la dimensión del espacio de características d tiende a infinito y el número de muestras crece limitado por $2(d+1)$

SVM

CASO LINEAL

- Calcular los pesos usando los puntos de entrenamiento (los incertamientos que existen)
- Dos clases: -1 y +1
 - Función de decisión: $\hat{w}x + w_0 = 0$ - $\begin{cases} \hat{w}x_i + w_0 \geq 0 & \text{para } y_i = +1 \\ \hat{w}x_i + w_0 \leq 0 & \text{para } y_i = -1 \end{cases}$ | Estimar w en los entornamientos
 - Podemos determinar que $\|\hat{w}\|/d = 2$, luego $d = \frac{2}{\|\hat{w}\|}$, donde d es el margen a maximizar.
 - La función a minimizar será: $f(w) = \frac{\|\hat{w}\|^2}{2}$

La distancia de un punto x_i a la frontera lineal $g(x) = \hat{w}^T x + w_0 = 0$ es: $d(x_i, g(x)) = \frac{|y_i(\hat{w}^T x_i + w_0)|}{\sqrt{\hat{w}^T \hat{w}}}$; por lo que para calcular el margen hay que calcular la menor distancia de los puntos del conjunto de entrenamiento a la frontera: $\text{margen}(\hat{w}, w_0) = \min_{i=1 \dots n} \frac{|y_i(\hat{w}^T x_i + w_0)|}{\sqrt{\hat{w}^T \hat{w}}}$: Es un problema max min con restricciones.

Normalización

Elegir de entre todas las fronteras posibles, aquella para la que el menor valor de $|g(\hat{x}_i)| = y_i(\hat{w}^T \hat{x}_i + w_0)$ sea 1

El margen se calculará como: $\text{margen}(\hat{w}, w_0) = \min_{x_i \in S} \frac{|y_i(\hat{w}^T \hat{x}_i + w_0)|}{\sqrt{\hat{w}^T \hat{w}}} = \frac{1}{\sqrt{\hat{w}^T \hat{w}}}$; por lo que para maximizar el margen hay que maximizar el cociente; por lo que el problema queda como un max min con las restricciones:

$$\min_w \frac{1}{2} \hat{w}^T \hat{w}, \text{s.t. } y_i(\hat{w}^T \hat{x}_i + w_0) \geq 1, i = 1 \dots n$$

Problema de programación cuadrática con un solo óptimo y con complejidad computacional polinomial

Buscar el menor valor de $|g(x)|$ - en valor absoluto - y dividir los pesos calculados entre ese valor.

INFERENCIA DUAL

• Un problema de programación lineal tiene la forma: $\min_{x \geq 0} \{ c \cdot x \mid Ax \geq b \}$

• La inferencia dual es: $\max_{P \in P} \{ v \mid A^T v \geq c \}$ | $P \rightarrow$ familia de problemas. Constituye la dominancia por antiteticos (combinaciones lineales) no negativas.

Def 1 - para $v \geq 0$; $v \cdot Ax \geq v \cdot b$ es una antiteticidad de $Ax \geq b$.

Def 2 - $v \cdot Ax \geq v \cdot b$ es una dominancia de $c \cdot x \geq v$, si $v \cdot Ax \geq v \cdot b$ implica que $c \cdot x \geq v$ (para $x \geq 0$); es decir: $v \cdot A \leq c$ y $v \cdot b \geq v$.

$$\max_{v \geq 0} \{ v \mid v \cdot A \leq c, v \cdot b \geq v \} \quad \text{ó} \quad \max_{v \geq 0} \{ v \cdot b \mid v \cdot A \leq c \} \quad \text{cuando } Ax \geq b, x \geq 0 \text{ es factible.}$$

|| \longrightarrow Dualidad fuerte; Máximo valor del problema dual = Mínimo valor del problema primal.

$$\min_{x \geq 0} \{ c \cdot x \mid Ax \geq b \}$$

Ejemplo: Primal

$$\min 4x_1 + 7x_2$$

$$\begin{cases} 2x_1 + 3x_2 \geq 6 \\ 2x_1 + x_2 \geq 4 \\ x_1, x_2 \geq 0 \end{cases}$$

Dual

$$\begin{aligned} &\max 6u_1 + 4u_2 \\ &2u_1 + 2u_2 \leq 4 \\ &3u_1 + u_2 \leq 7 \\ &u_1, u_2 \geq 0 \end{aligned}$$

\rightarrow Buscar la primera covariación del primal con una combinación lineal de estos dos (dominante). Por lo que multiplicaremos la primera covariación sumando u_1 y por la segunda sumando u_2 . El valor de la covariación resultante se fija en los factores min y max.

R solver