1. Capitulo 1: Imágenes Digitales

1.1. CONCEPTOS BÁSICOS

El **procesamiento de imágenes** es el área del conocimiento que se ocupa de los algoritmos que permiten realizar transformaciones sobre una imagen digital. Mientras que la **visión computarizada o visión artificial** hace uso de estos algoritmos integrándolos con sistemas apropiados de hardware, introduciendo sus propios algoritmos.

Estos últimos se podrían clasificar en:

- Orientados a aplicaciones prácticas
- Orientados a actividades investigación. Poseen perspectivas de aplicación a corto plazo sujeto a actividades que buscan alcanzar **capacidades de visión avanzadas** (procesamiento de puntos, de áreas, transformaciones geométricas, etc.). Estas ultimas introducen el concepto de **extracción de características**, un sistema de reconocimiento bajo condiciones de entorno o estructuradas, lo que significa que el sistema no posee información a priori a cerca de las características de la posición espacial, iluminación etc. Si debe poseer información a cerca de ciertas propiedades visuales del objeto.

El ojo humano percibe imágenes analógicas (considerada como una función continua f(x,y) en un espacio continuo entre dos dimensiones), pero para que la imagen pueda ser procesada por una computadora primero debe ser digitalizada. El proceso resulta de dos operaciones:

- **Muestreo de intensidades:** correspondiente a una matriz de CxR puntos. Cada ves que se muestrea una señal se da un valor a cada pixel de la imagen, un color determinado, una determinada intensidad.
- **Cuantización de niveles:** se trabajara con diferentes niveles de grises para facilitar el trabajo, entonces cada pixel va a tener un nivel diferente de oscuridad (escala de grises). Se procesan los diferentes niveles de gris por medio de un convertidor A/D, siendo k el número de bits por muestra de la imagen.

La imagen digital queda representada por RxC pixeles cada uno codificado con k bits.

Formato PGM

Es un formato de imagen digital que resulta fácil de comprender y manipular por cualquier lenguaje, se lo emplea con fines didácticos o experimentales. Una imagen PGM representa una imagen digital en escalas de grises.

Podemos considerar que por cada nivel de gris tenemos 8 bits, implica 256 niveles de grises diferentes, según las características de la imagen codificada pueden emplearse una mayor cantidad de valores.

1.1.1. VECINDAD DE UN PIXEL

Para comprender los procedimientos de la visión artificial se requiere transformar imágenes mediante un gran conjunto de métodos que efectúan diversas transformaciones a partir del cálculo de relaciones matemáticas entre un pixel y los que lo rodean.

Desde el campo **biológico** el procesamiento de la imagen se asocia con el término de **campo receptivo** el cual estaría formado por el conjunto de neuronas que procesan una porción de información.

En el caso de **procedimientos computacionales** es preciso conocer el entorno o **vecindad** de un pixel. Se denomina vecindad local o simplemente vecindad a los pixeles adyacentes al mismo.

Tipos de vecindad:

- Vecindad 4 $N_4(P)$: los 4 pixeles adyacentes que tienen un borde común con el pixel central. Forma de cruz. Un pixel en coordenadas (x,y) tiene 4 vecinos en coordenadas horizontales y verticales cuyas coordenadas son: (x+1,y); (x-1,y); (x,y+1); (x,y-1)
- Vecindad 8 $N_8(P)$: los 4 pixeles adyacentes que tienen un borde común con el pixel central y también se consideran los pixeles situados en las diagonales ($N_D(P)$) cuyas coordenadas son (x+1,y+1); (x+1,y-1); (x-1,y+1); (x-1,y-1).

  Representa el *conjunto base* de un pixel (P) $N_B(P)= N_4(P)+ N_D(P)$
- Vecindad 6 $N_6(P)$: los 4 pixeles adyacentes que tienen un borde común con el pixel central y se agregan dos pixeles opuestos sobre las diagonales.

La vecindad adoptada en un caso en particular es determinante para considerar válido el resultado que se espera obtener.

1.1.1.1. ADYACENCIA

Dos sub conjuntos de una imagen S son adyacentes si algún pixel de S1 es adyacente (vecino) de algún pixel de S2.

1.1.1.2. CONECTIVIDAD

Es un concepto que se utiliza para establecer las fronteras de un objeto y las regiones que componen la imagen. Para establecer si dos pixeles están conectados debemos ver si estos son adyacentes o vecinos en algún sentido que cumplan con algún criterio de similaridad (que sean parecidos).

Tipos de conectividad:

- 4 conectividad: 2 pixeles p y q con valores en V están 4conectados si q Э al conjunto $N_4(p)$.
- 8 conectividad: 2 pixeles p y q con valores en V están 8conectados si q Э al conjunto $N_8(p)$.
- n-conectividad: 2 pixeles p y q con valores en V están n-conectados si:
  - q Э al conjunto $N_4(p)$.
  - q Э al conjunto $N_D(p)$ y $N_4(p) \Omega N_4(p)$ es vacío (no pertenezcan al conjunto V).

1.1.1.3. REGIÓN Un set de pixeles que están conectados.

1.1.1.4. CAMINO

Un camino del pixel p, con coordenada (x,y), a q, con coordenadas (s,e) es una sucesión distinta de pixeles con coordenadas (x,y) (x0,y0) (x1,y1)…(xn,yn) (coordenadas de pixeles adyacentes entre si).

La longitud del camino es n; cantidad de pixeles por los que hay que pasar para llegar de un pixel a otro.

1.1.1.5. DISTANCIA

D4(p,q)= |x-s| + |y-e| diferencia absoluta de cada una de las coordenadas.

D8(p,q)= max (|x-s|;|y-e|)

Eucledіana (geometrica) real, que vemos. De(p,q)= $\sqrt{\quad}$

1.1.1.6. *CONTORNO* Sub conjunto de pixeles de una regios que son adyacentes a pixeles que no pertenecen a la región.

1.1.2. HISTOGRAMA

Hay algunos métodos de procesamiento que dependen de las propiedades estadísticas de la imagen. Un **histograma** representa la distribución de probabilidades de las frecuencias de intensidades de una imagen. Si tenemos una imagen digital con niveles de grises en el rango [1,L] su histograma es una función discreta con valores $h(g_i) = n_i / n$ (distribución de frecuencia) i=1,2..L Siendo g_i=el i-esimo nivel de gris, n_i=nro. de píxeles en la imagen con ese nivel de gris, n=nro. total de pixeles de la imagen. La función h representa la probabilidad de que un pixel elegido al azar pertenezca a un determinado nivel de gris.

A partir del histograma podemos obtener también valores etadisticos como el valor medio de niveles de gris $m_g = \sum g x\, h(g)$ y la desviación estándar $\sigma^2 = \sum (g-m_g)^2 x\, h(g)$. Con g =1,2,…L

1.1.2.1. BINARIZACIÓN. UMBRAL OPTIMO

Una imagen binaria es aquella que esta formada solamente por 1 y 0. Existe un histograma especial llamado **bimodal** que representa la distribuicon de probabilidades de este tipo de imagen. Este se caracteriza por tener dos máximos locales claramente separados. Este histograma corresponde a una imagen con aéreas claras y oscuras y muy pocas o ningún nivel de gris. De este tipo de imágenes y de su histograma, se pueden obtener imágenes binarias detectando su umbral óptimo como el valor mínimo entre las dos cimas y agrupando los valores hacia un lado u otro del umbral. (DIBUJO)

1.1.3. FRECUENCIA ESPACIAL

Velocidad con la que cambia la intensidad de los pixeles de un área. La frecuencia determinada al recorrer horizontalmente una secuencia de pixeles. El ancho de pixeles varía en forma abrupta, varia en forma no abrupta, existen diferentes tipos de anchos.

1.1.4. CONTRASTE

Es la sensibilidad del ojo humano frente a los diferentes niveles de intensidades en aéreas de imágenes adyacentes. Se describe de dos maneras:

1.1.4.1. FRECUENCIA ESPACIAL

Considerando la denominada función de sensibilidad al contraste FSC. La cual establece que ante una mayor frecuencia espacial menor es la sensibilidad al contraste (mayor frecuencia menos contraste). (DIBUJO)

1.1.4.2. INTENSIDAD - CONSTANTE DE WEBER

La constante establece que la respuesta del ojo al contraste no es lineal, ante una pequeña variación de intensidad el ojo no la detecta, pero a medida que crece la variación, la sensibilidad del ojo crece proporcionalmente.

El menor cambio discernible en la magnitud de una estimulo es proporcional al estimulo $\Delta x=kx$. Para que la persona se de cuenta del contraste, la intensidad tiene que variar en un 2%. (DIBUJO)

1.1.5. CUANTIZACIÓN

La intensidad de una imagen analógica se cuantifica en niveles de gris, la cuantización es usualmente lineal, pero para resaltar el contraste entre partes oscuras y claras puede ser no lineal.

Para reducir los requerimientos de almacenamiento de la imagen se una manera drástica se van disminuyendo las cantidades de bits. La perdida de información es reconocida a partir de los 5 bits y los bits menos significativos 0, 1 y eventualmente 2 contienen ruido.

1.2. PROCESAMIENTO DE IMAGENES

1.2.1. ECUALIZACIÓN

Es un proceso aplicado a imágenes de bajo contraste, es decir con áreas de nivel medio de grises. Este proceso consiste en transformar los niveles de grises: g g´, se reacomodan las líneas del nuevo histograma con nuevos intervalos Δg de manera de obtener niveles de densidades correspondiente entre todos los valores de g y g´.

Si tenemos g=nivel de gris; h(g)=frecuencia de nivel de gris; Δg=k*h(g) donde k= numero de niveles de gris sobre la sumatoria de h(g) (que por lo general es 1) y g´=g+Δg. el nuevo histograma es h(g´) donde traslado los valores originales de h(g).

Gg	hh(g)	Δg=k*h(g)	Gg'(g)	h(g´)

Igualación del histograma

Supongamos que deseamos encontrar una operación puntual tal que aplicada a una imagen, el histograma de la imagen a la salida tenga el mayor número de pixeles para cada valor de pixel, es decir que tenga un histograma plano. Esta operación puede ser útil para poner a las imágenes en un formato consistente previo a la comparación o a la segmentación. El número total de pixeles en cada nivel será igual a A_0/D_m, donde D_m es el máximo valor del pixel, y A_0 el área de la imagen.

De acuerdo a la ecuación (9-13) el histograma de la imagen resultante es un cociente de dos funciones con el mismo argumento. Si el numerador y el denominador son la misma función escalada por una constante, entonces el histograma será una constante, esto es:

$$f'(D) = \frac{D_m}{A_0} H(D) \quad . \qquad (10\text{-}1)$$

Integrando la ecuación (10-1), obtenemos:

$$f(D) = \frac{D_m}{A_0} \int_0^D H(u)du \qquad (10\text{-}2)$$

Por otro lado la función de densidad de probabilidad de una imagen es su histograma normalizado por su área.

$$\rho(D) = \frac{1}{A_0} H(D) \quad , \qquad (10\text{-}3)$$

donde H(D) es el histograma y A_0 es el área de la imagen. Además la función de distribución acumulada, es el área determinada por un umbral, normalizada por el área de la imagen.

$$P(D) = \int_0^D \rho(u)du = \frac{1}{A_0} \int_0^D H(u)du \quad . \qquad (10\text{-}4)$$

De esta manera, la función de distribución acumulada es la operación puntual que 'aplana' el histograma. Es decir,

$$f(D) = D_m \, P(D), \qquad (10\text{-}5)$$

y la función que iguala el histograma de la figura 10-1 es:

$$B(x, y) = f[A(x, y)] = D_m \, P[A(x, y)] \qquad (10\text{-}6)$$

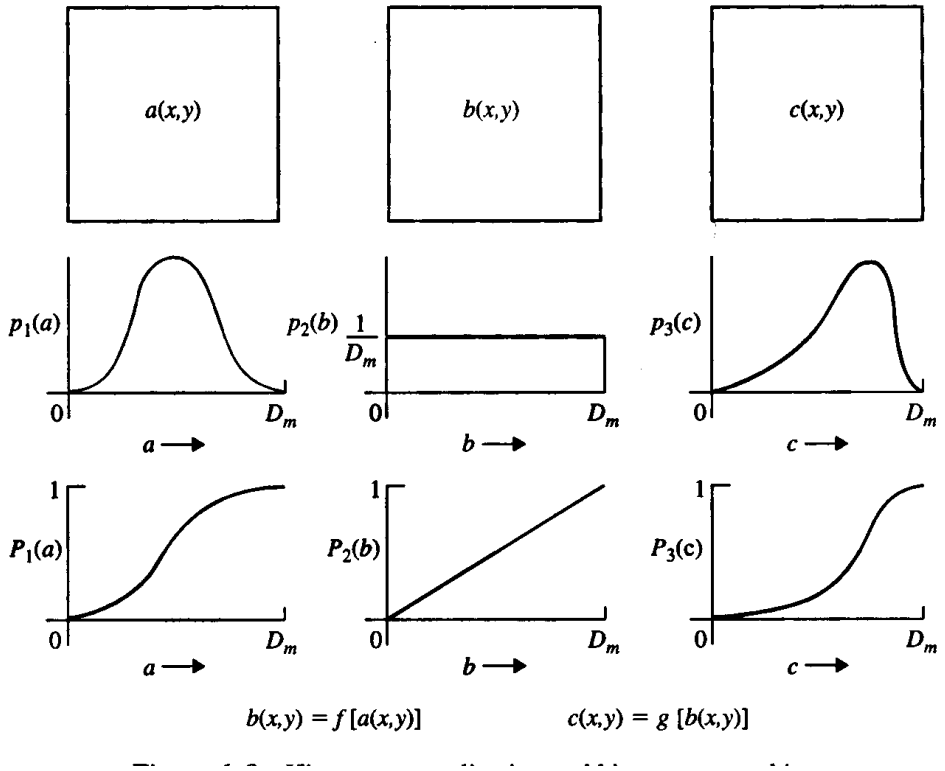


Figura 10-1 Igualación de histograma y ajuste de histograma.

Algunos niveles no son ocupados, algunos otros son altamente poblados. En promedio, el histograma es plano.

Histogram Matching (Correspondencia de histogramas).

Algunas veces es deseable transformar una imagen tal que su histograma corresponda ('matches') al de otra imagen con una cierta forma funcional.

Supongamos que deseamos transformar A(x, y) en C(x, y), con $H_3(D)$ como el histograma especificado.

Esto se puede hacer en dos pasos. Primero usamos f(D) para transformar A(x, y) en B(x, y). B(x, y) tiene un histograma aplanado. Después realizar una segunda operación g(D) a B(x, y), para obtener C(x, y), es decir

$$C(x, y) = g\,[B(x, y)]. \qquad (10\text{-}7)$$

De la ecuación (25) sabemos como obtener B(x, y). Además la operación:

$$B(x, y) = D_m\, P_3[C(x, y)], \qquad (10\text{-}8)$$

transformará a C(x, y) en una imagen con histograma aplanado, que es lo opuesto de lo que buscamos.

Expresando B(x, y) como en la ecuación 27, podemos escribir la segunda operación puntual ecuación (26) como:

$$C(x, y) = g\{D_m P_3[C(x, y)\ ]\}. \qquad (10\text{-}9)$$

Lo cual significa que la secuencia de operaciones $D_m P_3(D)$ seguida por g(D), no produce ningún efecto. Por lo tanto se puede deducir que g(D) es la inversa de $D_m P_3(D)$.

$$g(D) = P^{-1}{}_3(D/D_m) \qquad (10\text{-}10)$$

Si queremos transformar A(x, y) en C(x, y) en un solo paso.

$$C(x, y) = g\{f[A(x, y)]\} = P^{-1}{}_3\{P_1\{A(x, y))]\} \qquad (10\text{-}11)$$

Calibración fotométrica

Históricamente, el uso más común de las operaciones puntuales, es la remoción de la no linealidad fotométrica inducida por el digitalizador.

Supongamos un digitalizador de película que tiene una no linealidad entre la densidad de la película y el valor asignado al pixel. Podemos pensas en un digitalizador ideal seguido por una operación no lineal.

Deseamos diseñar una segunda operación puntual, para restaurar la no linealidad que el digitalizador produce en la imagen. La transformación no lineal del digitalizador, es conocida o se puede medir.

Deseamos seleccionar g(D) tal que el efecto neto de las dos operaciones en cascada es cero; es decir

$$C(x, y) = g\{f[A(x, y)]\} = A(x, y). \qquad (10\text{-}12)$$

Esto se satisface por

$$g(D) = f^{-1}(D) \qquad (10\text{-}13)$$

La curva de transferencia del digitalizador f(D), puede ser medida digitalizando una imagen de prueba. Esta función, normalmente es no decreciente e invertible numéricamente, para producir la escala de transformación.

Un problema puede surgir si el digitalizador se satura y da una pendiente de f(D) igual a cero.

Ejemplo:

$$D_B = f(D_A) = aD^2_A + b \qquad (10\text{-}14)$$

Despejando D_A se obtiene la inversa de f(D), o sea g(D)

$$g(D_B) = \sqrt{\frac{D_B - b}{a}} \qquad (10\text{-}15)$$

1.2.2. *INVARANTES GEOMETRICAS*

Valores que nos ayudan a la toma de decisiones en base a una imagen. Son datos obtenidos de una imagen.

1.2.2.1. ***ÁREA***

Sumatoria de todos los valores de todos los pixeles de una región. $A= \sum p_{ij}$

1.2.2.2. ***BARICENTRO***

Para una superficie definida las coordenadas del eje de gravedad (i,j) se calculan como el promedio de la suma de las coordenadas de cada pixel que pertenece a la región.

Es decir:

; al ser la imagen binaria considero los $p_{ij} = 1$.

1.2.2.3. ***PERÍMETRO***

Suma de los pixeles del contorno, definiendo si se trabaja con los pixeles del contorno o con los externos a la región.

1.2.2.4. ***NUMERO DE EULER***

Dados dos patrones y encontrar en la imagen la cantidad de ocurrencia de los mismos. El número de Euler se calcula por la diferencia entre el total de ocurrencias del primero menos el total de ocurrencias del segundo. Indica cuantos espacios internos tiene la región, cuan sólida es la imagen.

1.2.2.5. ***EJES PRINCIPALES DE INERCIA***

Sirven para determinar la orientación del objeto en el plano.

El eje de inercia esta definido por el momento de inercia que las partes de un objeto ejercen respecto de ese eje. Al considerar una imagen binaria en el *plano*, tenemos que la contribución de cada pixel es el *nivel de gris* (y no masa) y solo consideraremos los pixeles de valor 1 como aquellos que aportan la inercia a la región. Además debemos tener en cuenta que un eje de inercia de cualquier región pasa por el baricentro

$j´=j-j_0$ y $i´=i-i_0$. Para calcular la inercia de un pixel respecto al eje debemos calcular el cuadrado de la distancia del pixel al eje de inercia: $i´\text{sen}\theta - j´\cos(\theta-\rho)$; siendo θ el ángulo de inclinación respecto al eje vertical y ρ la distancia perpendicular el eje desde el origen.

Así todos los pixeles de la región multiplicados por la distancia al eje, sumados estos valores, nos darán la inercia de la región respecto a ese eje: $I=\sum\sum p_{ij}(i´\text{sen}\theta - j´\cos\theta)^2 = a\,\text{sen}^2\theta - 2b\,\text{sen}\theta\cos\theta + c\cos^2\theta$ siendo:

$a=\sum\sum i´^2 p_{ij}$; $c=\sum\sum j´^2 p_{ij}$ y $b=\sum\sum i´j´p_{ij}$

1.2.2.6. ***ORIENTACION***

Es un método más robusto para determinar ángulos de la orientación de un objeto. Su cálculo se hace a partir del momento de inercia calculado: $\tan 2\theta_0 = 2b/(a-c)$

1.2.2.7. ***SPREAD***

Nos dice cuan irregular es el contorno de la región, como es de macizo o disperso. Es el cociente de suma entre los momentos máximos y mínimos y el cuadrado del área.

1.2.3. FILTROS

Sirven para modificar aquellos valores que difieren demasiado de sus vecinos. Calcula un nuevo pixel para la imagen transformada en función de su vecindad. Hay que considerar que además de reducir el ruido, se reducen los detalles pequeños.

1.2.3.1. MEDIANA

Recorremos la imagen de un vector con una ventana de filtro predeterminada. Al tomar una imagen de un vector (ancho impar) vemos el valor que posee le pixel central, ordenamos la secuencia de esa ventana, tomamos el valor central y lo reemplazamos en la imagen original. Es decir reemplazamos en valor central de la imagen original por el valor central de la ventana ordenada.

1.2.3.2. VALOR MEDIO

Este caso es similar al anterior solo que no se reemplaza por el valor central de la ventana ordenada, sino que se reemplaza por el promedio de los valores de la ventana.

1.2.4. DETECCIÓN DE CONTORNO

El contorno son los pixeles que delimitan a un objeto por completo. A partir de un pixel inicial se va detectando el contorno siguiendo a sus vecinos, de acuerdo a las siguientes reglas: giro a la izquierda +90º si el pixel pertenece a la región, giro a la derecha -90º si el pixel no pertenece a la región. De acuerdo a como se considere la región si figura negra con fondo blanco o figura blanca con fondo negro va a ser el contorno detectado. Todos los pixeles que consideramos, pertenecen a la imagen son los que forman parte del contorno de la imagen

1.2.5. CONVOLUCIÓN

Consiste en re calcular el valor de un pixel en base a los pixeles de su vecindad. Dependiendo de la mascara que se utilice es la nueva imagen que vamos a obtener. El método consiste en definir una máscara cuya dimensión sea igual a la dimensión de la vecindad elegida para procesar la imagen. Consideramos a los vecinos de un pixel representados por la matriz y la máscara a aplicar calculamos los nuevos valores del pixel P como la suma de los productos entre cada uno de los elementos de por cada uno de los elementos de y reemplazar el valor de P por el nuevo valor calculado .

1.2.6. DETECCIÓN DE BORDES

Borde es cualquier parte de la imagen que cambie bruscamente de intensidad (mayor frecuencia espacial) y frecuentemente corresponden a partes del contorno de un objeto. La forma en la que se obtienen es por Convolución. Existen mascaras pre definidas para la detección de bordes:

Prewit Sobel Prewit Sobel

Bordes Verticales Bordes Horizontales

Estas máscaras se pueden combinar para obtener diferentes interpretaciones. Existen además otros tipos en lo cuales se pueden detectar bordes en cualquier dirección.

Pos procesamiento de bordes

Muchas veces las líneas de bordes obtenidas deben ser pos procesadas, ej. cuando se quiere detectar los bordes que representan a cierto objeto dado. Los bordes al ser detectados, con un detector de bordes, resultan en un conjunto de bordes aislados y con ramificaciones. Por ese motivo las ramificaciones e interrupciones deben ser eliminadas, y posteriormente puede requerirse un proceso de adelgazamiento para que todos los bordes tengan un pixel de ancho. En las imágenes de gris tienen máximos fluctuantes y poseen interrupciones.

- *Búsqueda en arco del pixel sucesor*

  Una regla simple es elegir como pixel sucesor al vecino con mayor nivel de gris. El problema en este caso es que debido al ruido la línea puede oscilar entre un amplio margen de valores de gris. La búsqueda en arco del pixel sucesor consiste en efectuar una búsqueda en una vecindad ampliada. Se define un espacio de búsqueda como un sector limitado a ambos lados de la dirección principal del borde en su extremo, el pixel sucesor queda determinado por el rayo de búsqueda con el mayor valor medio. Como resultado las fluctuaciones de nivel de gris en el borde son relativamente neutralizadas.

- *Búsqueda con dirección controlada por gradiente*

  El pixel inicial debe ser el pixel del borde. La dirección principal de búsqueda puede calcularse de los gradientes de un detector de bordes local en el pixel investigado. Esta dirección puede describirse por el ángulo relativo al eje x. El pixel sucesor queda determinado por este vector de búsqueda.

1.2.7. SEGMENTACIÓN

Proceso de extraer regiones de interés de una imagen, es decir todos los pixeles que tiene igual valor y que se tocan entre si. Así se obtiene una lista de todas las regiones extraídas y los pixeles correspondientes a las mismas y si la imagen no es binaria también se contara con intensidades de pixeles.

Tipos de segmentación:

- Determinación de Áreas (regiones): Se van incorporando pixeles que posean igual valor de intensidad respecto a los pixeles de la región.
- Determinación por perímetros (líneas): No se obtiene un borde cerrado, sino que se debe realizar un pos proceso para obtener dicho contorno, mediante: seguimiento de contorno, conexión de bordes abiertos, eliminación de bordes sobrantes.

Métodos orientados a regiones imágenes binarias:

La imagen de gris es binarizada y luego dividad en regiones a través de un algoritmo de segmentación. Una etiqueta es asignada a cada píxel de cada nueva región. Algoritmo de segmentación:

1.2.7.1. REGION AGGREGATION

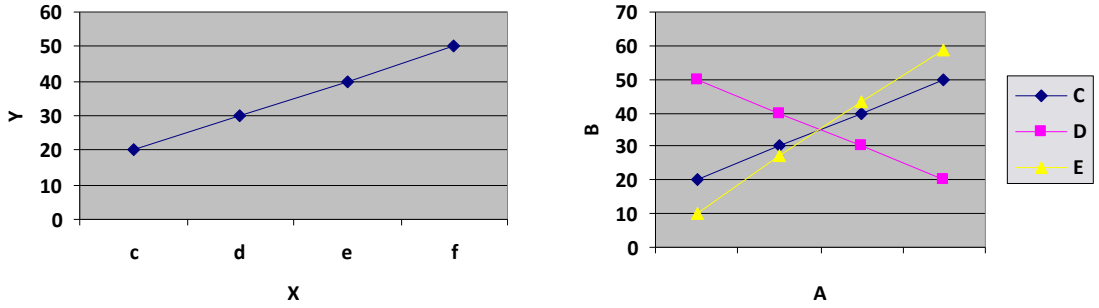
Se trata de elegir un conjunto de semillas para determinar si pertenecen o no a una región. Analizada una semilla, si esta no pertenece a una región, se descarta, en caso contrario se etiqueta con un número de región y posteriormente se inspeccionan los ocho vecinos de este píxel. A su vez, cada vecino se analiza de la misma manera y se va etiquetando de acuerdo al valor de la semilla, y así sucesivamente. Finalmente se obtiene una imagen etiquetada la cual puede verse haciendo corresponder a cada etiqueta diferentes niveles de grises.

Métodos orientados a líneas

1.2.7.2. TRANSFORMADA DE HOUGH

Procedimiento utilizado para detectar los puntos relevantes pertenecientes a una recta o curva obteniendo los parámetros que definen a ese objeto en la imagen. Para la detección de rectas se obtienen los parámetros de la forma norma Hessiana y el número de píxeles dispuestos sobre la recta.

Si consideramos una recta y=ax+b b=-ax+y



Un punto en el plano (x,y) se transforma en una recta en el plano (a,b).

Una recta en el plano (x,y) se transforma en una punto en el plano (a,b)

Así los puntos c, d y e en (x,y) son rectas C,D y E en (a,b).

La robustez del método se evidencia en el hecho de que las líneas rectas son detectadas aun si están interrumpidas o no, todos los píxeles yacen exactamente sobre la línea.

Esta es una técnica que permite descubrir formas en una imagen, más precisamente morflogías simples en los bordes, y presenta una notable robustez con respecto a las distorsiones. Se basa en transformar puntos de la imagen en un espacio de parámetros. La idea es encontrar curvas parametrizabales como rectas, círculos y polinomiales. Se pueden encontrar formas más complejas pero el costo computacional crece rápidamente. Generalmente se realiza detección de bordes a la imagen, y luego se aplica la transformada a esta. De esta forma son menos los puntos que hay que recorrer y por lo tanto más rápido es el algoritmo.

Su modo de operación estadístico, consiste en que para cada punto que se desea averiguar si es parte de una línea se aplica una operación dentro de cierto rango, con lo que se averiguan las posibles líneas de las que puede ser parte el punto. Esto se continúa para todas los puntos en la imagen, al final se determina cuales líneas fueron las que más puntos posibles tuvieron y esas son las líneas en la imagen.

La ecuación básica para la transformada de Hough fue diseñada para detectar líneas rectas y curvas. Esto significa que cualquier línea recta en el espacio de la imagen (x,y) es representada por un solo punto en el espacio de parámetros ρ, θ; y cualquier parte de esta línea recta es transformada en el mismo punto.

$$\rho = xi \cos(\theta) + yi \sin(\theta) \quad [1.1]$$

El rango of θ comprenderá de 0-π; si hacemos esta la distribución K, el cálculo se puede realizar con θ = k π / K (k = 0,1,2, K-1). Si N es el elemento de borde, entonces la ecuación [1.1] para la transformada de Hough debe calcularse NK veces. Si utilizamos k en la ecuación [1.1] en vez de θ; podemos escribir de la manera siguiente:

$$\rho k = xi \cos(k) + yi \sin(k) \quad [1.2]$$

donde ρ es la distancia perpendicular de la línea al origen, y θ es el ángulo entre la normal de la línea y el eje x.

La ventaja de este método es que evita singularidades, como por ejemplo rectas de pendiente infinita. Si se representa φ y ρ en un plano cartesiano, una recta queda determinada mediante un punto con coordenadas (phi(recta),ro(recta)), mientras que un punto, se representa como una función senoidal. Si por ejemplo tenemos dos puntos, tendremos dos senoides desfasadas alfa grados dependiendo de las coordenadas de los puntos. Dichas senoides se irán cruzando cada 180º. La interpretación geométrica de este hecho, es que la función seno de cada punto, representa las infinitas rectas que pasan por cada punto, cuando dos puntos comparten la misma recta, sus representaciones senoidales se cruzan, se obtiene un punto. Cada vez que se da media vuelta (ρ=180º) se vuelve a repetir la misma recta, por lo que volvemos a obtener otro punto, que de hecho es la misma recta.

Un problema de la representación cartesiana de la recta es que tanto la pendiente (a) como la ordenada en el origen (b) tienden a infinito conforme la recta se acerca a posiciones verticales. Para evitar este problema se usa la representación polar (normal) de la recta.

$$\rho = uCos\theta + vSen\theta$$

$0 \leq \rho \leq (u^2_{máx} + v^2_{máx})^{1/2}$, $0 \leq \theta \leq \pi$

2. Capitulo 2: Reconocimiento de Patrones.

2.1. GENERALIDADES

Un patrón es una descripción de un objeto. Podemos distinguir dos categorías de reconocimientos:

- Perceptual: reconocimiento de objetos concretos.
- Conceptual: reconocimiento de objetos abstractos.

El reconocimiento de patrones se basa en el reconocimiento perceptual. A su vez el reconocimiento perceptual, como atributo humano, esta asociado a la inferencia inductiva y asocia una percepción con algunos conceptos generales o pistas derivados de su experiencia pasada. Es la estimación del parecido relativo con que los datos de entrada pueden ser asociados a experiencias pasadas que forman las pistas e información a priori para el reconocimiento.

El problema del reconocimiento puede ser concebido como el hecho de discriminar, clasificar o categorizar la información de entrada, no entre patrones individuales sino entre poblaciones, por medio de la búsqueda de características o atributos invariantes entre los miembros de una población.

Si definimos una **clase** como una categoría determinada por algunos atributos comunes y un **patrón** como una descripción de cualquier miembro de una categoría que representa a una clase de patrones, decimos que la teoría del **Reconocimiento de patrones** se ocupa de buscar la solución al problema general de reconocer miembros de una clase dada en un conjunto que contienen elementos de muchas clases diferentes.

2.1.1. APLICACIONES

Reconocimientos de caracteres, del habla de voz, predicción del clima, diagnósticos médicos, etc.

2.1.2. PROBLEMAS – ETAPAS DEL RECONOCIMIENTO

*2.1.2.1. **SENSADO:***

Identificación y representación de la información obtenida de un objeto a reconocer mediante algún sensor. Cada cantidad medida describe una característica del objeto. Toda la información obtenida esta contenida en un vector de patrones. Cuando el contenido de los vectores patrones son números es fácil representarlos en el plano n-dimensional. En ciertas ocasiones esta información pude agruparse en Cluster, grupos de puntos que poseen características comunes.

*2.1.2.2. **EXTRACCIÓN DE CARACERÍSTICAS:***

Consiste en la reducción de dimensión de un vector de patrones, extracción de las características más importantes y significativas del objeto a reconocer. Los métodos utilizados se denominan “análisis de componentes principales” o “transformación de los ejes principales” lo que se logra es reducir la dimensión de los patrones, pero con la posibilidad de perder un bajo porcentaje de información contenida en ellos.

*2.1.2.3. **CLASIFICACIÓN:***

Es necesario diseñar un mecanismo que dado un conjunto de patrones de entrada, de los cuales no se conoce a priori la pertenencia a la clase, permita determinar a que clase pertenece ese patrón. Un mecanismo útil puede ser la “función de decisión” o “función discriminante”

2.1.3. CLASIFICACIÓN DE LOS METODOS

- **Heurísticos:** basados en la intuición y experiencia.
- **Matemáticos:**
  - **Determinísticos:** Estadística no explicita.
  - **Estadísticos:** Estadística explicita.
- **Sintácticos:** Basados en relaciones entre los objetos.
- Si el sistema de reconocimiento puede auto ajustar ciertos coeficientes internos que definen las funciones discriminantes estaremos en presencia de un **sistema adaptativo** o con capacidad de aprendizaje.

2.1.4. REPRESENTACIÓN GRAFICA

Es importante que las personas podamos representar y visualizar claramente las situaciones, solo en el plano y en el espacio tridimensional. Otro método es un vector como los valores de un histograma.

2.2. FUNCION DE DECISION LINEAL

Al tomar decisiones a cerca de la pertenencia a una clase dada de los patrones con que se conforma un sistema de reconocimientos es necesario establecer algunas reglas en que basar estas decisiones.

Consideramos $d(x)=w_1x_1+ w_2x_2 +w_3=0$ como la ecuación de una línea de separación donde w es el vector de parámetros y x es el vector de las variables de entrada. La función d(x) puede usarse como una función de decisión dado un patrón de clasificación desconocida x, para que podamos determinar si pertenece a una clase w1 si cumple una condición o a una clase w2 en caso contrario.

Esta función de decisión se puede generalizar a n-dimensiones: $d(x)=w_1x_1+ w_2x_2+\ldots+w_nx_n+w_{n+1}=0$, $d(x) =\sum w'_0x + w_{n+1}$ donde el vector w'_0 es llamado vector de pesos de parámetros y el vector x el vector de patrones. Además se suele adicionar 1 como última componente de los vectores de patrones (v.p aumentado).

2.2.1. MATRIZ DE COEFICIENTES

Los vectores conocidos permiten calcular la matriz de coeficientes. Sabiendo que se quiere clasificar patrones que corresponden a la clase w_1 o w_2 y que cada clase contiene dos patrones de dimensión y donde los supra índices indican la clase. Si las clases son linealmente separables entonces se debe encontrar un vector w=(w1,w2,w3) tal que se cumpla:

Para calcular w partimos de la aproximación del error cuadrático medio. Así W es solución del sistema lineal y se calcula: $w = [X^tX]^{-1}X^tY$

2.2.2. EJEMPLO CONCLUSION

2.2.3. VALORES NORMALIZADOS

Consideramos que la salida puede ser 1 o 0, las entradas y las salidas se computan normalizadas de modo que esten siempre entre 0 y 1. Esto implica poder aplicar de forma general un programa que ejecuta algún algoritmo de clasificación a distintos problemas.

2.2.4. PROBLEMAS REALES

Cuando no es posible separar a las clases en una forma sencilla se presentan dos situaciones: **separables en forma no lineal** es una curva y no una línea lo que me divide las clases o **no separable** cuando los patrones de cada una de las clases están, a sus ves, distribuidas en clusters múltiples.

2.2.5. EVALUACION DE RESULTADOS – MATRIZ DE CONFUSION

Fases de un clasificador:

- Entrenamiento: estimación, optimización, adaptación, aprendizaje, etc. De acuerdo al tipo de clasificador. Supervisado No supervisado. Cálculo de los coeficientes W
- Prueba: representa la capacidad de generalización de un clasificador.

2.2.6. FUNCION DE DECISION GEERALIZADA

Generalizando aun mas la funcion de decision se pueden considerar coo patrones de entarada funciones f(x) funciones reales simples del patrón x, representa una infinita variedad de funciones dependiendo de la elección de funciones y el numero de términos usados para la expansión.

2.3. CLASIFICADOR POLINOMIAL

El clasificador lineal es inadecuado para resolver problemas de clasificación de mayor complejidad, por lo que para poder comprender el clasificador no lineal se introduce el concepto de función de decisión generalizada, lo que

significa aplicar una función a cada una de las componentes del vector de entrada, tomándose estas como componentes de un nuevo vector. Si al menos alguna de estas funciones es no lineal se obtiene una función de clasificación no lineal.

Dado un vector $x=(x_1,x_2,x_3,\ldots,x_n)$ podemos transformarlo en $x^*=(f(x_1),f(x_2),f(x_3,)\ldots,f(x_n))$. El caso mas simple se da cuando las funciones son lineales donde $f(x)=x$ donde $d(x)=w_0x+w_{n+1}$. Ahora cuando tenemos funciones cuadráticas se plantea lo siguiente: $x^* = (x_1,x_2)^2=(x_1^2,x_2^2,x_1 x_2, x_1 ,x_2,1)$ obteniendo una $d(x^*) = w_1x_1^2 + w_2x_2^2 + w_3x_1 x_2 + w_4x_1 + w_5x_2 + w_6$. Este caso puede generalizarse:

Donde para cada una de las funciones puede definirse $f(x)=x_p^s x_q^t$ con p y q=1,…n y s y t =1,0 .

2.3.1. EXPRESIÓN MATRICIAL DE LA APROXIMACION POLICNOMICA

En lugar de considerar solo una funcion de clsificacion polinomica d(x) podemos necesitar emplear más de estas. En este caso la funcion de decision pasa a ser una componente de un vector de salida.

2.3.2. GRADOS DE LIBERTAD

Numero de terminos necesrios para definir una funcion de decision polinomial, depende del grado r del polinomio y de la n dimension del patrón.

2.3.3. PROCESAMIENTO EN PARALELO

(dibujo)

2.3.4. ESQUEMA DE COMPUTO

En el aumento de las características de los problemas reales se hace inviable el cálculo de la matriz de coeficiente, por lo que se aplica otro procedimiento. Se introduce un nuevo vector $z = (x_1,x_2,x_3,\ldots,x_n, y_1,y_2,y_3,\ldots,y_m)$ y una nueva matriz M_z.

Para resolver el clasificador polinomial simplemente debemos construir la matriz M y aplicar transformaciones por filas.

2.3.5. ESTRATEGIAS DE PIVOTEO

Al resolver numéricamente un sistema de ecuaciones el error residual va a depender del orden en que se seleccionen los elementos pivote que se utilizan en las transformaciones. El orden adecuado surge también de la elección de la estrategia adecuada:

Pivoteo Máximo (LI)

Implica seleccionar en cada paso el mayor valor de la diagonal principal. Este valor corresponde a la contribución al error final de esa columna. Esto permite minimizar el error final. Además mide la dependencia lineal de esta columna en el sistema, de manera que nos permite eliminar aquellas columnas linealmente independientes.

Pivoteo Mínimo Error (ME)

Analiza el erro que producen el conjunto de los componentes de una columna buscando minimizar el error final.

Ambas implican establecer un orden de prioridad para las coordenadas del vector x en cuanto a la influencia de cada una en el reconocimiento de los patrones que deben confrontar.

2.3.6. OPTIMIZACIÓN RECURSIVA

Se refiere a que ambos términos del calculo del vector $w = \sum[X^tX]^{-1} \sum X^tY$ pueden ser expresados recursivamente como: $\sum[X^tX]^{-1} =(1-\alpha) \sum[X^tX]^{-1}+\alpha[X^tX]^{-1}$ (el otro también). A partir de lo cual pude obtenerse una formula para el cálculo recursivo de w conocida como "*Ley de aprendizaje":*

2.4. CLASIFICADOR DE MARGEN OPTIMO

Todos los resultados obtenidos por los diferentes clasificadores pueden ser mejorados aplicando un método conocido como Maquina de Vectores de Soporte (MVS). Todos los clasificadores se basan en minimizar el erro cuadrático, el MVS permite separar un conjunto de vectores en clases diferentes, pero al mismo tiempo logran una mejora adicional en la determinación del factor de reconocimiento del clasificador y busca reducir la cantidad de patrones a emplear en el cálculo del clasificador de modo de permitir una solución analítica. La idea fundamental es considerar solo aquellos vectores del conjunto de entrenamiento que estén muy próximos a la zona de transición entre una clase y otra.

El principio MVS optimiza la matriz de coeficientes w definiendo dos restricciones:

1 – El híper plano discrimina la clase por minimización del error de clasificación.

2- la línea que representa el híper plano debe corresponder a aquella que maximiza la mínima distancia entre las clases. Encontrar el híper plano con el máximo margen de separación (distancia mínima entre dos puntos de entrenamiento).

2.4.1. FORMULACIÓN LAGRANGIANA

1- Las ecuaciones se pueden asociar a variables denominadas de LaGrange con factores no negativos, simplificando la solución. Uno para cada una de las restricciones vistas.

2- Los patrones de entrenamiento solo aparecerán como producto entre vectores. Lo que permite extender este principio a casos no lineales. Hay un operador de LaGrange por cada punto de entrenamiento.

2.4.2. SOLUCIÓN AL PROBLEMA CUADRATICO

Obtención de los operadores de lagrange a partir de la ecuacion de LaGrange.

2.4.3. CALCULO DE COEFICIENTES

La matriz de coeficientes W puede obtenerse como una suma de algunos vectores (vectores de soporte) de los vectores de aprendizaje.

2.4.4. CLASES NO SEPARABLES

Se relaja la restricción de que la distancia entre los vectores de soporte sea exactamente +-1, por lo que se introducen nuevas variables de relajación.

2.4.5. CLASIFICADORES NO LINEALES

Se aplica la transformacion no lineal al vector x.

2.4.6. MAS DE DOS CLASES

Uno Contra todos

Un problema de k clases se divide en k problemas de clasificación. En cada problema k, una clase es separa da las otras. El resto de las clases se toma como una segunda clase.

Pairwise

Solo dos clases son clasificadas en pasos sucesivos.

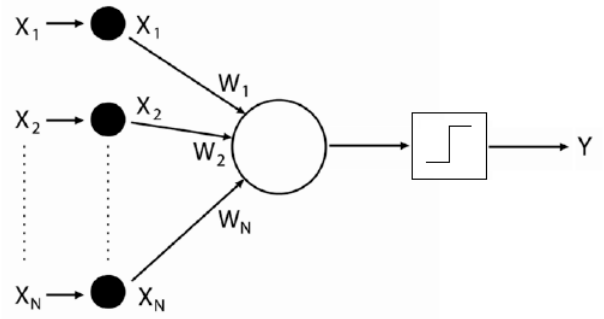
3. Capítulo 3: Redes Neuronales.

3.1. REDES NEURONALES

El costo de resolver un problema de clasificación de forma exacta es muy elevado. La mejor forma de encarar este problema es utilizar un modelo de redes neuronales artificiales, basado en el funcionamiento de neuronas biológicas, en el cual cuando se alcanza un valor umbral a la salida de una neurona se propaga la señal.

3.2. PERCEPTRON

Un esquema basado en redes neuronales que realiza una función de clasificación lineal.



En lugar de expandir el vector x, se adiciona como entrada un valor unitario (siempre). El componente de w que corresponde a ese valor unitario se denomina b.

El funcionamiento del perceptrón es el siguiente: el perceptrón recibe las entradas, calcula una función de decisión lineal y aplica una función umbral que obliga al perceptrón obtener solo dos salidas (0y1 o -1y1 dependiendo de cómo se defina la función F). La diferencia entre el reconocimiento de patrones y el perceptrón es a la hora en que se realizan las funciones de clasificación lineal en el momento de calcular la matriz de coeficientes o el vector de pesos, ya que no se calcula en forma matricial, sino que mediante un proceso iterativo de aproximación.

3.2.1. LEY DE APRENDIZAJE

El algoritmo de Aprendizaje del Perceptron es del tipo Supervisado lo cual requiere que sus resultados sean evaluados y se realicen las modificaciones del sistema si fuera necesario. Los valores de los pesos pueden determinar el funcionamiento de la red; estos valores se pueden fijar o adaptar utilizando diferentes algoritmos de entrenamiento de la Red; de manera que, el Perceptron se expone a un conjunto de patrones de entrada, y los pesos de la red son ajustados de forma que al final del entrenamiento se obtengan las salidas esperadas para cada uno de estos patrones de salida.

Para el **Problema de la función OR**, la red debe ser capaz de devolver, a partir de los cuatro patrones de entrada, a que clase pertenece cada uno. La salida que produce sin tener en cuenta el umbral es : $y = f\left(\sum_i w_i x_i\right)$ donde $\mathbf{x_i}$ son las **entradas** a la neurona; $\mathbf{w_i}$ son los **pesos** entre las neuronas de la capa de entrada y la de la capa de salida; y ***f*** es la **Función de Salida o Transferencia.**

En la etapa de aprendizaje se irán variando los valores de los pesos obteniendo distintas rectas. Lo que se pretende al modificar los pesos de las conexiones es encontrar una recta que divida el plano en dos espacios de las dos clases de valores de entrada.

En el caso de no existir término independiente en la ecuación, el umbral es cero y las posibles rectas pasarán por el Origen de Coordenadas.

Para el **Problema de la función AND**, es imposible un valor de umbral de 0, ya que no existe ninguna recta que pase por el origen de coordenadas y que separe los valores; por lo que es necesario introducir un término independiente para poder realizar esta tarea. Para ello se considera una Entrada de valor fijo 1 a través de una conexión con peso w_0, (el cual permitirá desplazar la recta del origen de coordenadas), que representa el umbral (w_0=-θ) y cuyo valor deberá ser ajustado durante la etapa de aprendizaje.

La función de salida queda como $$y = f\left(\sum_{i=1}^{n} w_i x_i - \theta\right)$$

3.2.1.1. *LEY DE HEBB*

Es la primera ley de aprendizaje y representa un modelo simplificado del funcionamiento de una neurona biológica. Calcula w como sigue: $\Delta w_{ij} = l_r.x_i.y_j$. Donde los coeficientes se modifican según el producto de la entrada por la salida y un factor que se denomina taza de aprendizaje que permite ajustar la variación de esa magnitud. Los parámetros se modifican según: $w_{ij+1} = w_{ij} + \Delta w_{ij}$ y $b_{ij+1} = b_{ij} + \Delta b_{ij}$

3.2.1.2. *LEY DEL PERCEPTRON*

Considera que en el producto de Hebb no interviene directamente en la salida sino que la diferencia entre la salida obtenida al procesar la información y la salida correcta que debería haberse generado $\Delta w_{ij} = x_i.(y'_j - y_j)$ donde y'_j es la salida deseada y y_j la salida realmente obtenida. Resulta como un premio castigo ya que si x pertenece a una clase se modifica el valor de w, de lo contrario no. Los parámetros se modifican según: $w_{ij+1} = w_{ij} + \Delta w_{ij}$ y $b_{ij+1} = b_{ij} + \Delta b_{ij}$

3.2.2. *CLASIFICACIÓN NO LINEAL*

El perceptrón no puede resolver clasificaciones no lineales, para ello se debe utilizar más de un perceptrón obteniendo una red. Esta red puede ir creciendo hasta alcanzar un perceptrón multicapa, el cual se caracteriza por tener tres tipo de capas: la de entrada (1), las ocultas (2) y la de salida (3).

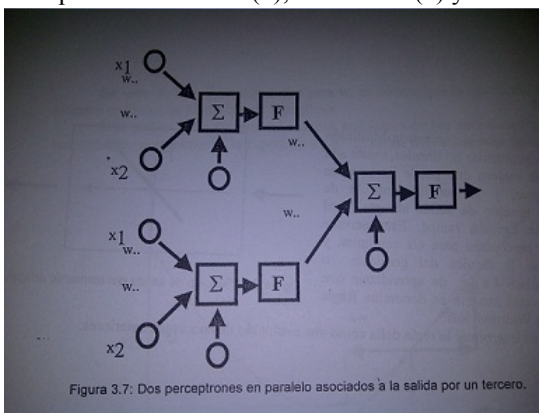


Figura 3.7: Dos perceptrones en paralelo asociados a la salida por un tercero.

Para resolver este tipo de problemas se usa más de un perceptrón, es decir, construyendo una red. Por ejemplo, dos perceptrones resuelven el problema de clasificación no lineal presentado por la función XOR

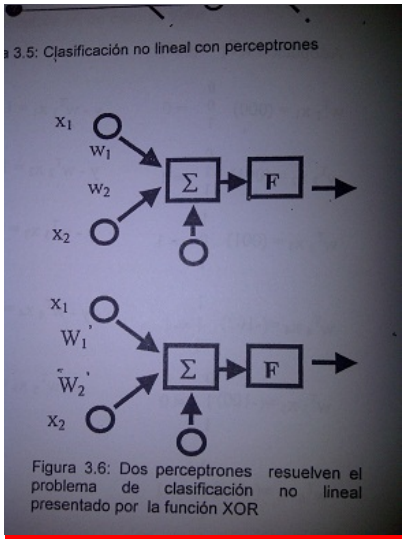


Figura 3.6: Dos perceptrones resuelven el problema de clasificación no lineal presentado por la función XOR

PERCEPTRON MULTICAPA

Un perceptron multinivel o multicapa es una red de tipo Feedforward, compuesta de varias capas de neuronas entre la entrada y la salida de la misma; en éste tipo de red no es necesario conocer la salida de las células de la capa oculta.

Esta red permite establecer regiones de decisión más complejas que la anterior.

Para éste tipo de redes hay que definir el número de neuronas ocultas y el de capas con las que se desea trabajar de acuerdo a la complejidad del Problema. Sin embargo éste número no debería de ser superior a cuatro ya que con un Perceptron de cuatro capas se pueden generar regiones de decisión arbitrariamente complejas. Solo en ciertos problemas se puede simplificar el aprendizaje mediante el aumento de la extensión de la función de activación, en lugar del aumento de la complejidad de la red.

En la práctica un número de neuronas excesivo en cualquier capa puede generar ruido; pero por otro lado, si existe un número de neuronas redundantes se obtiene mayor tolerancia a fallos.

Para el Problema de la XOR, se ha de utilizar una red de tres neuronas, distribuidas en dos capas y se deberán ajustar 6 pesos.

3.3. ADALINE

El **adaline** (de **ADA**ptative **LIN**ear **E**lement) es un tipo de red neuronal artificial. La diferencia entre el Adaline y el perceptrón es que el perceptrón solo tiene capacidad para clasificar, ya que utiliza una función umbral sobre la suma ponderada de las entradas, a diferencia del adaline, que es capaz de estimar una salida real.

Definición

Generalmente se compone de una sola capa de *n* neuronas (por tanto *n* valores de salida)

con *m* entradas con las siguientes características:

Las *m* entradas representan un vector x de entrada que pertenece al espacio R^m.

Por cada neurona, existe un vector w de pesos sinápticos que indican la fuerza de conexión entre los valores de entrada y la neurona. En la práctica representan la ponderación de cada entrada sobre la neurona.

Una constante θ.

La salida y de la neurona se representa por la función de activación, que se define como

$$y = \sum_{i=1}^{n} x_i \cdot w_j + \theta$$

Aprendizaje

A diferencia del perceptrón, a la hora de modificar los pesos durante el entrenamiento el Adaline tiene en cuenta el grado de corrección de la salida estimada respecto a la deseada. Esto se consigue mediante la aplicación de la *regla Delta*, y que se define, para un patrón de entrada x^p con una salida estimada y^p y una salida deseada d^p, como $| d^p - y^p |$.

Dado que el objetivo del Adaline es poder estimar de la manera más exacta la salida (conseguir una salida exacta es prácticamente imposible en la mayoría de los casos), se busca minimizar la desviación de la red para todos los patrones de entrada, eligiendo una medida del error global. Normalmente se utiliza el error cuadrático medio.

$$E = \frac{1}{2}\sum_{p=1}^{m}(d^p - y^p)^2$$

La manera de reducir este error global es ir modificando los valores de los pasos al procesar cada entrada, de forma iterativa, mediante la regla del descenso del gradiente. Suponiendo que tenemos una constante de aprendizaje α:

$$\Delta_p w_j = -\alpha \frac{\partial E^p}{\partial w_j}$$

Si operamos con la derivada, queda:

$$\Delta_p w_j = \alpha(d^p - y^p) \cdot x_j$$

Que será la expresión que utilizaremos por cada entrada para modificar los pesos.

A pesar de que el perceptrón multicapa pueda resolver problemas no lineales, existe aun una importante mejora que puede introducirse sobre el mismo y es sencillamente reemplazar la función de salida abrupta (escalón) por una rampa. Esto posibilita generalizar el perceptrón a entradas – salidas continuas y utilizar la técnica de gradiente o descenso iterativo.

La ley de aprendizaje que corresponde a este modelo se denomina regla delta o ley de Widrow/Hoff:

Perceptrón $\Delta w_{ij} = x_i.(y'_j - y_j)$

Ley de Hebb $\Delta w_{ij} = l_r.x_i.y_j$

Regla delta : $\Delta w_{ij} = l_r.x_i.\ (y'_j - y_j) = l_r.x_i.\delta = \alpha(y-\sigma)x_i$

Las unidades de procesamiento se denominan Adalinas.

Para comprender el sentido de la búsqueda por iteración descendente del valor de W se considera el cuadrado del error cuadrático medio:

$$LMSE = \frac{1}{2L} \times \sum_{k=1}^{L} (y_k - \sigma_k)^2 \quad E=$$

Y además la función para calcular el error de la red (error en una neurona):

Esto se considera cuando es un caso unidimensional donde la grafica del error al cuadrado en función de w es una parábola, en la cual la regla delta permite modificar en cada paso el valor de w hasta alcanzar el óptimo. El gradiente debe ser descendiente.

Lo que se debe hacer es calcular la derivada de la función para que nos oriente en la búsqueda de un mínimo relativo, derivamos el erro con respecto a los distintos pesos:

;

Resolviendo cada una:

Por que la salida es igual a una función lineal de la sumatoria, solo para adaline.

Reemplazando:

En el caso en el que no se un plano, debemos conformarnos con encontrar un mínimo local lo suficientemente bueno, que pueda efectuar la clasificación que exige el problema. Si la taza de aprendizaje es alta difícil de converger al resultado, si la taza es baja se hace mas largo el proceso de resolución.

Ventajas

Con respecto al perceptrón el Adaline posee la ventaja de que su gráfica de error es un hiperparaboloide que posee o bien un único mínimo global, o bien una recta de infinitos mínimos, todos ellos globales. Esto evita la gran cantidad de problemas que da el perceptrón a la hora del entrenamiento debido a que su función de error (también llamada de coste) posee numerosos mínimos locales.

A pesar de que el perceptrón multicapa pueda resolver problemas no lineales, existe aun una importante mejora que puede introducirse sobre el mismo y es sencillamente reemplazar la función de salida abrupta por una rampa. Esto posibilita generalizar el perceptrón a entradas – salidas continuas y utilizar la técnica de gradiente o descenso iterativo.

3.4. RETROPROPAGACION / BACK PROPAGATION

Éste método está basado en la generalización de la Regla Delta.

El **funcionamiento** es como sigue: primero se aplica un patrón entrada como estímulo para la primera capa de las neuronas de la red, se va propagando a través de todas las capas superiores hasta generar una salida, se compara el resultado obtenido en las neuronas de salida con la salida que se desea obtener y se calcula un valor del error para cada neurona de salida. A continuación, estos errores se transmiten hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de la capa intermedia que contribuyan directamente a la salida, recibiendo el porcentaje de error aproximado a la participación de la neurona intermedia en la salida original. Este proceso se repite, capa por capa, hasta que todas las neuronas de la red hayan recibido un error que describa su aportación relativa al error total. Basándose en el valor del error recibido, se ajustan los pesos de conexión a cada neurona, de manera que en la siguiente vez que se presente el mismo patrón, la salida esté más cercana a la deseada; es decir el error disminuya

La **importancia** de la Red **Backpropagation** consiste en su capacidad de autoadaptar los pesos de las neuronas de las capas intermedias para aprender la relación que existe entre un conjunto de patrones dados como ejemplo y sus salidas correspondientes.; para así poder aplicar esa misma relación, después del entrenamiento a nuevos vectores de entrada, dando una salida activa si la nueva entrada es parecida a las presentadas durante el aprendizaje. Esta característica, es la capacidad de Generalización, entendida como la facilidad de dar salidas satisfactorias a entradas que el sistema no ha visto nunca en su fase de entrenamiento.

A diferencia de la Regla Delta en el caso del Perceptron, la técnica backpropagation o Generalización de la Regla Delta, requiere el uso de neuronas cuya función de activación sea continua, y por lo tanto, diferenciable. Generalmente, la función utilizada será del tipo sigmoidal

Aprendizaje de la Red Backpropagation

En una red Backpropagation existe una capa de entrada con n neuronas y una capa de salida con m neuronas y al menos una capa oculta de neuronas internas. Cada neurona de una capa (excepto las de entrada) recibe entradas de todas las neuronas de la capa anterior y envía su salida a todas las neuronas de la capa posterior (excepto las de salida). No hay conexiones hacia atrás ni laterales.

A diferencia de la Regla Delta en el caso del Perceptron, la técnica backpropagation o Generalización de la Regla Delta, requiere el uso de neuronas cuya función de activación sea continua, y por lo tanto, diferenciable. Generalmente, la función utilizada será del tipo sigmoidal.

Consiste en entrenar redes multicapas. La solución viene dada por la posibilidad de calcular el error a la salida de cada capa en función del error de salida de la capa de salida. Para explicar que modo se logra esto, se plantea el siguiente algoritmo de aprendizaje en el que se consideran una capa oculta:

Pasos del algoritmo de retropropagación:

1) Inicializar la taza de aprendizaje y los pesos de toda la red.
2) Ingresar una entrada aleatoria.
3) Propagar los valores de la capa de entrada hacia la capa oculta. (para cada una de las neuronas de la capa ooculta)

$$Net_{ij} = \sum_{i=0}^{A} w_{1ij} x_i \qquad (1)$$ A Cantidad de neuronas de la primera capa

$$h_j = \frac{1}{1 + e^{-Net_{ij}}} \qquad (2)$$

4) Propagar los valores de la capa oculta hacia la capa de salida.

$$Net_{ij} = \sum_{i=0}^{B} w_{2ij} h_i \qquad (3)$$ B cantidad de neuronas de la capa oculta

$$\sigma_j = \frac{1}{1 + e^{-Net_{ij}}} \qquad (4)$$

5) Calcular los errores de las unidades de la capa de salida.

$$\delta_{2j} = \sigma_j(1 - \sigma_j)(y_j - \sigma_j) \qquad (5)$$

6) Calcular los errores de la/s unidades de las capa/s ocultas.

$$\delta_{1j} = h_j(1 - h_j) \sum_{i=1}^{C} (\delta_{2j} w_{2ij}) \qquad (6)$$ C cantidad de neuronas de la capa de salida

7) Ajustar los pesos entre la capa oculta y la capa de salida.

$$W_{2ij(t+1)} = W_{2ij(t)} + \Delta W_{2ij} \qquad (7)$$

$$\Delta W_{2ij} = \alpha \, \delta_{2j} \, h_i \qquad \alpha = 0{,}35$$

8) Ajustar los pesos entre la capa de entrada y la capa oculta.

$$\Delta W_{1ij} = \alpha \, \delta_{1j} \, x_i \qquad (8)$$

9) Repetir desde el paso 2 hasta que el error total sea menor al deseado, o hasta que se cumpla la cantidad total de ciclos.

El error cuadrático medio considerado al final de cada ciclo es:

$$1/2L(\delta_{2j}-y_i)^2 \qquad (9)$$

Donde (y´-y) es el error en la capa de salida.

3.4.1. SOBRE ENTRENAMIENTO

Luego de un periodo de entrenamiento normal, el erro tiende a aumentar nuevamente, en estos casos se debe detener el proceso ya que se ha entrado en una etapa de sobre entrenamiento. Al sobre entrenar la red se desmejora el funcionamiento de la misma.

3.4.2. ESQUELETONIZACIÓN

Se aplica para reducir el número de neuronas de una red una vez que ha sido ya diseñada y entrenada satisfactoriamente. Esto se lleva a cabo para reducir la memoria necesaria en el proceso. Existen diferentes métodos para realizar la poda. Ellos se basan en quitar aquellas neuronas que contribuyen menor al proceso de calificación.

3.4.3. DISEÑO

Estimación del numero de neuronas

Determinar el grado de libertad para una red de tres capas: h=4,51F=(E+1)S1+(S1+1)A. algunos proponen el calculo de las neuronas de la primera capa oculta h=P/(10(m+n))

Variación de parámetros de inicialización

Variaciones en los mismos para obtener mejores resultados debido a la fuerte dependencia.

Programación por lotes

Se emplea un archivo batch para probar diferentes configuraciones con diferentes parámetros iniciales.

Proporcionalidad de los conjuntos de entrenamiento y prueba

Dividir el total de patrones disponibles.

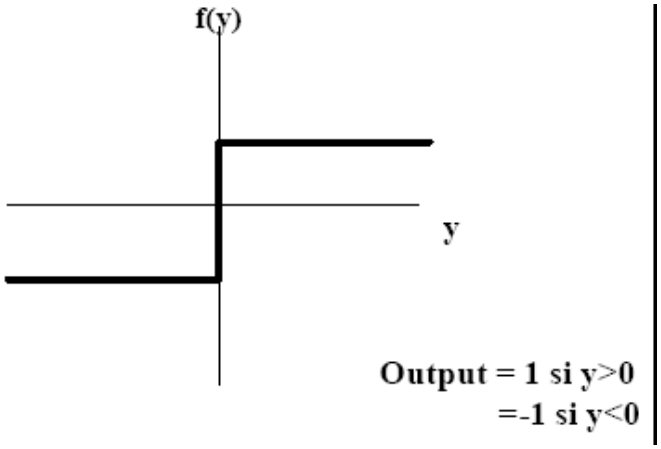
3.5. AGRUPAMIENTO, COMPETTIVDAD Y AUTO ORGANIZACIÓN

Para determinar características subyacentes o bien patrones se pueden emplear un grupo de algoritmos. Uno de ellos es k-means representado como una capa particular de una red neuronal, a través de este análisis es posible determinar el vector de pesos de esta red, cuyas componentes resultan ser claramente valores aproximados a los centros de los cluster que eventualmente forman los datos disponibles. Básicamente se calcula la diferencia entre x-w para cada neurona, se determina la menor diferencia y se actualizan los pesos w de la neurona ganadora en una fracción de esa diferencia.

3.3 Funciones de Activación

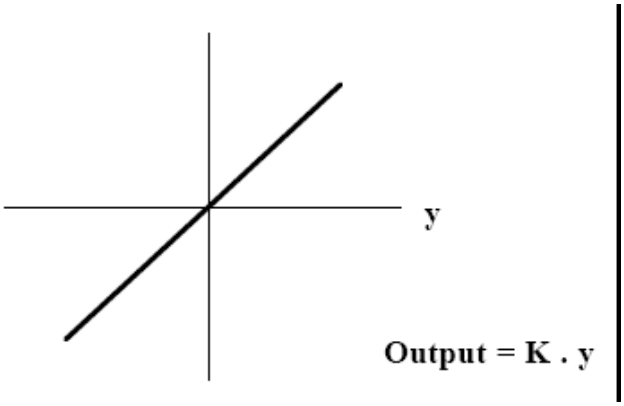
3.3.1 Función Escalón

La función escalón se asocia a neuronas binarias en las cuales cuando la suma de las entradas es mayor o igual que el umbral de la neurona, la activación es 1, si es menor, la activación es 0 (ó –1).



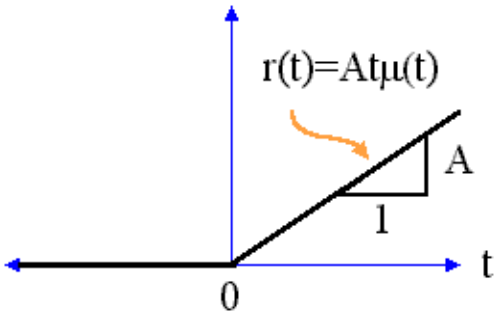
3.3.2 Función Lineal

La función lineal o mixta corresponde a la función F(x) =x. En las neuronas con función mixta si la suma de las señales de entrada es menor que un límite inferior, la activación se define como 0 (ó –1). Si dicha suma es mayor o igual que el límite superior, entonces la activación es 1. Si la suma de entrada está comprendida entre ambos límites, la activación se define como una función lineal de suma de las señales de entrada.



3.3.3 Función Rampa

La función rampa es la integral de la función escalón. Si consideramos que estamos sumando toda el área bajo la función escalón a hasta un tiempo t. Si t < 0 (cero), el valor de la integral será 0 (cero). Si es mayor que 0 (cero), entonces el valor será igual a la integral de 1 desde el tiempo 0 hasta el tiempo t, la cual también tiene el valor t, es decir:

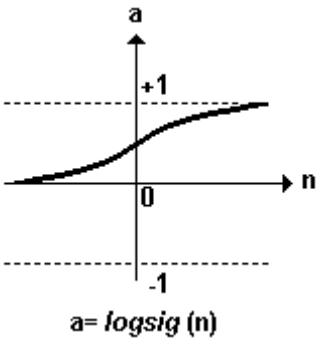


$$r(t) = \begin{cases} t, t > 0 \\ 0, t < 0 \end{cases}$$

3.3.4 Función Logarítmica Sigmoidal

Cualquier función definida en un intervalo de posibles valores de entrada, con un incremento monotónico y que tengan ambos limites superiores e inferiores podrá realizar la función de activación o transferencia.

Con la función sigmoidal, para la mayoría de los valores del estímulo de entrada, el valor dado por la función es cercano a uno de los valores asintóticos. Esto hace posible que en la mayoría de los casos, el valor de salida esté comprendido en la zona alta o baja del sigmoide. De hecho cuando la pendiente es elevada, esta función tiende a la función escalón. La importancia de ésta función es que su derivada es siempre positiva y cercana a cero para los valores grandes positivos o negativos; además toma su valor máximo cuando x es cero. Esto hace que se puedan utilizar las reglas de aprendizaje definidas para la función escalón, con la ventaja respecto a esta función, que la derivada está definida para todo el intervalo. La función escalón no podía definir la derivada en ele punto de transición y esto no ayuda a los métodos de aprendizaje en los cuales se usan derivadas.



$$a = \frac{1}{1+e^{-n}}$$

4. Capitulo 4: Precepción Visual

4.1. TEORIA DEL ESPACIO ESCALAR

4.1.1. SUAVIZADO

El proceso de Convolución permite la aplicación de una operación denominada suavizado. El objetivo del suavizado es la moderación del gradiente de las zonas de alto gradiente, hacer los bordes más nítidos o la imagen más borrosa. Debido a que el suavizado es una operación derivada de la Convolución se debe usar una más cara, a la cual le corresponden valores discretos de la campana de gauss

4.1.2. PIRAMIDE GAUSSIANA

El procedimiento de suavizado corresponde a la aplicación de un filtro especial a una imagen, este procedimiento puede aplicarse repetidas veces sobre la imagen obtenida. De este modo se obtiene una secuencia de imágenes cada vez más difusas. Con cada aplicación de este filtro se pierden detalles de la imagen, debido a esto es frecuente reducir la dimensión de la imagen obtenida. Este procedimiento que consiste en lograr un conjunto de imágenes cada vez menos nítidas y eventualmente de menor tamaño se conoce como pirámide gaussiana.

4.1.3. TEORIA DEL ESPACIO ESCALAR

Plantea que los sistemas de visión biológicos reconocen los objetos en diferentes escalas y que esto puede ser aplicable a la visión artificial, de este modo un sistema puede reconocer características que definen un objeto en diferentes escalas y componerlas de modo de reconocer un objeto.

4.2. RECONOCIMIENTO POR INVARIANTES

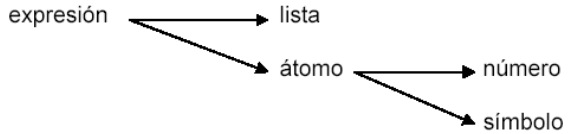
Otra forma de reconocimiento de objetos se denomina indexación, este se basa en extraer características de un objeto y almacenarlas asociadas a un índice o etiqueta a ese objeto, además se podrían almacenar las características invariantes de este objeto. El método consiste en comparar las características detectadas con las que corresponden a las que se describen en los objetos almacenados. Se prueba luego la correspondencia de las que corresponden a los objetos más comprometedores y se reconocen si aparecen en la imagen.

5. Capitulo 5: LISP

5.1. LISP

Programa interprete basado en el lenguaje funcional, no genera daños colaterales, es un hibrido ya que me permite relizar bucles y asignar variables. El nombre de LISP surge de abreviar "List Procesing" o Procesamiento de Listas. Fue propuesto por John Mc Carthy en 1959 en el MIT. En LISP toda la información (inclusive las sentencias) se expresan como listas o como los componentes de las mismas.

Un código de LISP está compuesto de formas o expresiones; el intérprete LISP lee una expresión, la evalúa e imprime el resultado. Este procedimiento se denomina ciclo de lectura, evaluación, impresión (read – eval – print loop).



Una **expresión o forma** es una lista o un átomo (símbolo, entero o cadena).

Un átomo es un símbolo o un número. Si la forma es una lista LISP trata el primer elemento como el nombre de una función, y evalúa los restantes elementos recursivamente, y luego llama a la función con los valores de los elementos restantes.

Un **número** se puede escribir en cualquiera de las notaciones habituales.

Un **símbolo** se representa por un nombre que está formado por caracteres alfanuméricos que no puedan interpretarse como números.

Una **lista** es una sucesión de cero o más expresiones entre paréntesis. El blanco (no la coma) es el separador de expresiones dentro de una lista. Cada una de estas expresiones se denomina elemento de la lista.

Cuando LISP evalúa una expresión, devuelve un valor (que será otra expresión) o bien señala un error.

- Un número se evalúa a sí mismo.
- Un símbolo se evalúa al valor que tiene asignado.
- Una lista se evalúa independientemente una por una del siguiente modo, siempre en función del primer elemento y en función de este considera como emplear los elementos siguientes:

1) **El primer elemento** de la lista debe ser un símbolo "s" cuyo significado funcional "F" esté definido.

2) Se evalúan los restantes elementos de la lista, obteniendo los valores v1,…,vn. Si el número o tipo de los valores obtenidos no es coherente con los argumentos requeridos por F, se produce un error.

3) La lista se evalúa a F(v1,…,vn).

La forma en la que se evalúan las listas corresponde a una notación pre fija, donde primero se escribe el operador y luego los parámetros.

5.2. FUNCIONES

Existen dos símbolos (T y NIL) que se evalúan a si mismos. Estos se utilizan como TRUE y FALSE o como TRUE y NULL en otros lenguajes. Un símbolo "s" puede estar ligado, es decir, contener una referencia o indicación a otra expresión "V". Llamamos ligadura al par (s,V). Un símbolo "s" (salvo T o NIL) se evalúa al valor "V" al que está ligado.

Recordar que forma es una expresión que se puede evaluar.

QUOTE

Es una forma que tiene un solo argumento. (QUOTE expresión)
El valor de (QUOTE expresión) es precisamente "expresión", sin evaluar. La abreviatura de QUOTE es " ' ".
(QUOTE expresión) = ' expresión
Ejemplo: (+ 3 4) 7, '(+ 3 4) (+ 3 4), (1 2 3) error, '(1 2 3) (1 2 3)

SETQ

Asigna el segundo argumento (que es una expresión) al primer elemento (que no es evaluado). (SETQ símbolo expresión)

```
>(setq a 5)        ; almacena el número 5 en la variable a
 5                 ; retorno
 >a                ; para consultar el valor almacenado en a
 5
```

CAR

Un único elemento, que debe ser una lista. Si el argumento no es NIL, el valor devuelto es el primer elemento de la lista argumento. Si el argumento es NIL, el valor devuelto es NIL.

CDR
Un único argumento, que debe ser una lista. Si el argumento no es NIL, el valor devuelto es la lista obtenida quitando el primer elemento de la lista argumento. Si el argumento es NIL, el valor devuelto es NIL. Se pueden hacer combinaciones de CAR y CDR

CONS
Exactamente dos argumentos. El segundo debe ser una lista. El valor devuelto es una lista cuyo primer elemento es el primer argumento, y cuyos restantes elementos son los elementos del segundo argumento.

LIST
Un número indeterminado de argumentos. Si no hay ningún argumento, el valor devuelto es NIL. Si hay algún argumento, el valor devuelto es una lista cuyos elementos son los argumentos.

APPEND
Un número indeterminado de argumentos, que deben ser listas. Si no hay ningún argumento, el valor devuelto es NIL. Si hay algún argumento, el valor devuelto es una lista cuyos elementos son los elementos de los argumentos.

LENGTH
Un único argumento, que debe ser una lista. El valor devuelto es la longitud del argumento, es decir, el número de elementos de la lista.

EVAL
Un único argumento. El valor devuelto es el valor resultante de evaluar el argumento. De esta forma, (EVAL expresión) realiza dos evaluaciones: la primera debido al ciclo normal de evaluación y la segunda debido al uso explícito de EVAL.

DEFUN
Es una macro. Ello quiere decir que no evalúa sus argumentos en la forma explicada. DEFUN toma sus argumentos literalmente. El primer argumento es in símbolo. El segundo argumento es una lista de cero o más símbolos que se denominan parámetros (lista-lambda). El cuerpo está formado por un número cualquiera de expresiones.
(DEFUN símbolo lista-lambda cuerpo)
(DEFUN cuadrado (N) (* N N) CUADRADO
(cuadrado 7) 49

PREDICADOS
Son las funciones que devuelven T o NIL.
(EQ e1 e2) es T cuando e1 y e2 son el mismo símbolo
(EQL e1 e2) es T cuando e1 y e2 son EQ o números iguales de igual tipo
(EQUAL e1 e2) es T cuando e1 y e2 son EQL o listas iguales
(= e1 e2 … en) es T cuando e1, e2, …, en son números todos iguales de cualquier tipo
(NULL e) es T cuando e es NIL
(ATOM e) es T cuando e es un átomo
(SYMBOLP e) es T cuando e es un símbolo
(NUMBERP e) es T cuando e es un número
(LISTP e) es T cuando e es una lista
(ENDP e) es T cuando e es NIL (lista vacía). Si e no es una lista da error
(ZEROP e) es T cuando e es cero
(PLUSP e) es T cuando e > 0
(MINUSP e) es T cuando e < 0

IF
Tiene como argumentos: Una condición. Una o dos expresiones. Si la condición es T, se evalúa la primera condición, en caso contrario se evalúa la segunda.

COND
Tiene un número indeterminado de argumentos. Cada argumento es una cláusula. Una cláusula es una lista formada por una o más expresiones. La primera expresión de cada cláusula es la condición y a las restantes se las puede denominar acciones.

Recursividad
LISP también soporta el concepto de recursividad.

DO / DO*
Son las formas iterativas más generales de Lisp.
(DO (ligadura *)
(test resultado *)
expresión-cuerpo *
)
donde "ligadura" puede ser:
símbolo | (símbolo) | (símbolo expr-inicial) | (símbolo expr-inicial expr-paso)
El significado es el siguiente:
- Se crea un entorno con ligaduras para los símbolos indicados.
- Se evalúa cada exp.-inicial (valor por defecto: NIL) y sus valores son asignados a los correspondientes símbolos. La evaluación es en paralelo para DO y secuencial para DO*
- Se evalúa test.
- Si test es verdadero, se evalúa cada expresión de la sucesión "resultado*", la evaluación de DO acaba y el valor que se devuelve es el de la última (por defecto NIL).
- Si test es falso, la evolución de DO continúa: primero se evalúa secuencialmente (en caso de existir) las expresiones del cuerpo. A continuación se evalúan las exp.-paso (en paralelo para DO y secuencialmente para DO*) y los valores son asignados a los correspondientes símbolos. A continuación se evalúa de nuevo el "test" y se repite todo el proceso.

DOLIST
Evalúa "expr-cuerpo" sucesivamente con "variable" ligado a cada uno de los valores "lista" y finalmente devuelve el resultado de evaluar "expr-final" con "variable" ligada a NIL.

LAMBDA
(LAMBDA lista-lambda [expresión] *) Las expresiones lambda sirven para referirse a funciones sin necesidad de darles nombre. Una expresión lambda da directamente la definición de la función. La "lista-lambda" es la lista de parámetros que serán utilizados en "expresión". Una expresión lambda puede aparecer en los mismos lugares que un símbolo con significado funcional. Por ejemplo, puede aparecer como primer elemento de una lista que se ha de evaluar; se dice entonces que tenemos una forma lambda. Los restantes elementos de la lista son los argumentos que se pasan a la definición lambda. Estos argumentos sustituyen a los parámetros en "expresión" y se devuelve el valor de la última de ellas.

FUNCTION - #'
Hasta ahora hemos considerado en Lisp dos tipos de datos: átomos y listas. Lisp considera también como datos a los objetos-procedimientos. Un objeto procedimiento no debe confundirse con el símbolo que lo tiene ligado.
Existe un objeto-procedimiento (llamémosle Función1) que es el significado funcional del símbolo F1. El objeto-procedimiento Función1 se recuperaría implícitamente al evaluar una lista de la forma: (F1 argumento)
Hasta ahora esta manera de tratar los objetos-procedimiento, propia de los lenguajes imperativos, es la única que hemos considerado. Sin embargo, en Lisp es posible manipular explícitamente un objeto-procedimiento mediante la forma especial FUNCTION

(FUNCTION argumento)
- Un solo argumento, que debe ser un símbolo o una expresión lambda.
- FUNCTION toma literalmente su argumento (no lo evalúa)
- El valor devuelto es:
o Si el argumento es un símbolo, el objeto-procedimiento que es el significado funcional del símbolo.
o Si el argumento es una expresión lambda, el objeto-procedimiento definido por la expresión.

o Otro tipo de argumento es un error.

OPTIONAL

Los parámetros que siguen a &OPTIONAL en una lista lambda son opcionales y de la forma:
(símbolo | símbolo expresión)
Las expresiones para los valores por defecto se evaluarán secuencialmente, por lo tanto es posible emplear en ellas los argumentos anteriores de la lista lambda.

KEY

Los parámetros de una lista lambda que siguen a &KEY son parámetros-clave. Los parámetros-clave son siempre opcionales, pero además de esto, cuando en una lista lambda aparece un parámetro-clave las reglas para asignar los argumentos a los parámetros no son las vistas anteriormente, es decir, los argumentos no se asignan en el mismo orden en que aparecen en la llamada.
Las reglas son las siguientes:
a) Los parámetros requeridos se emparejan con los primeros argumentos de la llamada. Si hay mas o menos, se produce un error.
b) A cada parámetro clave se le hace corresponder una palabra-clave, añadiéndole al principio el signo ":". Las palabras-clave son símbolos que se evalúan a sí mismos.
c) Si en la llamada figuran argumentos para los parámetros-clave, cada uno debe ir precedido por la correspondiente palabra-clave. El emparejamiento entre los parámetros y argumentos se produce independientemente del orden de aparición de los argumentos.

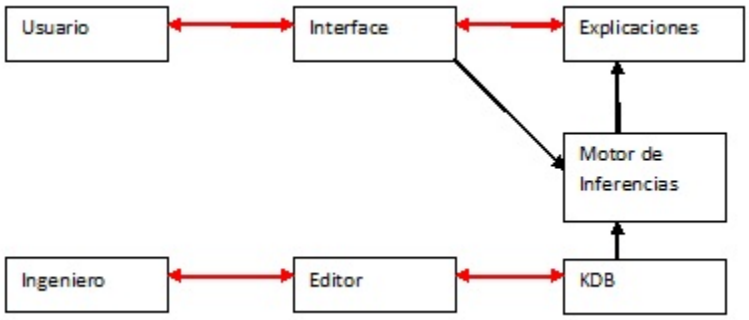
6. Capitulo 6: Sistemas Expertos

6.1. ¿QUE SON LOS SISTEMAS EXPERTOS?

Es un programa destinado a generar inferencias en aun área específica del conocimiento en forma similar a la que se espera de un experto humano. Deben resolver problemas por aplicación de conocimiento en un dominio específico. Este conocimiento es adquirido a través de la intervención de expertos humanos y almacenado en lo que se denomina Base de Datos de Conocimiento. Son aplicaciones a mundos reducidos, ya que no pueden operar en situaciones llamadas de sentido común por ser muy extenso el dominio de conocimiento que debe tener el sistema.

Se utiliza SE cuando:

No existe algún algoritmo para resolver un problema
El problema es resuelto satisfactoriamente por expertos humanos
Existe algún experto humano que pueda colaborar en el desarrollo de un SE
El conocimiento del dominio debe ser relativamente estático



INTERFACE

- Respuesta/aplicaciones/preguntas

MOTOR DE INFERENCIAS

- En el momento de creación del conocimiento conecta al Humano y traduce las Reglas
- Almacena todas la Reglas
- Sacar conclusiones basado en la lógica

KDB

- Base de datos del Conocimiento

EDITOR

- De sentencias lógicas

INGENIERO

- Ingeniero del conocimiento
- Obtiene e incorpora datos en KDB

Componentes de un SE:

Símbolos: Las diferentes definiciones que se van a utilizar en el SE, afirmaciones, elementos, etc.

Reglas: Se aplican sobre los símbolos, se deben obtener del conocimiento de los expertos. Son heurísticas dado que no es posible demostrar su validez general.

Hechos: son todos los predicados que se suponen verdaderos.

6.1.1. FUNDAMENTOS

Los sistemas expertos básicos se basan en la lógica de predicados. La diferencia entre esta y la proposicional es que la lógica de predicados emplea variables y permite expresar una premisa en una sola proposición mientras que en la otra deberíamos usar una proposición por cada una de las variables que estemos analizando. La representación mediante predicados es una forma de representar la estructura del conocimiento.

Para que la lógica de primer orden nos ayude a resolver problemas y garantice inferencias validas, el significado de los símbolos a emplear debe ser preciso no deben existir ambigüedades. En el caso de los SE se pretende aplicar la lógica a mundos que solo conocen en profundidad los expertos humanos, en que ellos expresan su conocimiento en lenguaje natural. Por lo que para que un SE basado en el formalismo lógico requiere definir los símbolos con los cuales se operará y su significado preciso, además de traducir el conocimiento del experto del lenguaje a natural a la expresiones lógicas correspondientes (reglas).

Los SE se basan en la posibilidad de aplicar las nociones de la lógica formal para incrementar una base de conocimientos y solo pueden aplicarse a mundos de muy baja complejidad. El modo mas inmediato consiste en aplicar reglas del tipo si-entonces e instanciando los antecedentes de la misma obtener una instancia valida para el consecuente.

En lugar de instanciar el consecuente para ver si es soportado por sus antecedentes se podría trabajar a la inversa:

Encadenamiento hacia adelante el SE toma una serie de afirmaciones de entrada y ensaya todas las reglas disponibles, una y otra vez, incorporando nuevas afirmaciones, hasta que ninguna de las reglas pueda producir nuevas afirmaciones.

Encadenamiento hacia atrás se comienza con una hipótesis y se trata de verificarla haciendo uso de las afirmaciones disponibles.

6.2. LÓGICA DE PREDICADOS

6.2.1. LÓGICA

La **lógica** permite la posibilidad de utilizar un conjunto de reglas para gestionar inferencias creíbles. Una **inferencia** es la extracción de conclusiones a partir de premisas mas o menos explicitas, puede ser resultante del sentido común o de la aplicación de reglas muy detallistas o cálculos estadísticos. La **epistemología** estudia el origen y características del conocimiento, en especial la forma en como aparece y sus limitaciones.

Los **formalismos** a usar para representar los hechos que ocurren en el mundo, deben ser lo suficientemente adecuados como para representar la información disponible.

El **silogismo** es una formula lógica con premisas seguidas de una conclusión.

La **lógica de primer orden** es uno de los formalismos más utilizados para representar el conocimiento. Cuenta con un lenguaje formal mediante el cual es posible representar formulas llamadas **axiomas** que permiten describir fragmento del conocimiento y consta de un conjunto de **reglas de inferencia** que aplicadas a los axiomas permiten derivar nuevo conocimiento.

El alfabeto

Posee dos símbolos:

- Lógicos: constantes, operadores proposicionales y cuantificación, lógicos y auxiliares.
- No lógicos: variables individuales, funciones n-arias y relaciones n-arias.

A partir de los símbolos se construyen las expresiones:

- Términos: una constante, palabra, letra, variable y una expresión de la forma.
- Formulas: son expresiones de la forma R(t1….tn) donde R es un símbolo de relación y t1…tn son los términos.

Los términos permiten nombrar objetos del universo, mientras que las formulas permiten afirmar o negar propiedades de estos o bien establecen relaciones entre los objetos del universo.

6.2.2. FORMULAS BIEN FORMADAS

La lógica proposicional permite el razonamiento mediante un mecanismo que primero evalúa sentencias simples y luego sentencias complejas mediante el uso de conectivos proposicionales. Una preposición es una sentencia simple con un valor asociado. La lógica proposicional permite la asignación de un valor v o f para la sentencia compleja, no tiene la facilidad para analizar las palabras individuales que componen la sentencia. Una FBF es una proposicion simple o compuesta de significado completo cuyo valor de veracidad puede ser determinado, basado en los valores de veracidad de las proposiciones simples y en la naturaleza de los conectores lógicos involucrados.

El **Silogismo categórico** dice que si es verdadera una instancia cualquiera de las siguientes premisas: Si M es A y B es M entonces B es A. la **resolución** se basa en que una afirmación no puede ser simultáneamente verdadera y falsa.

6.2.3. ELEMENTOS BASICOS

- Átomos: proposiciones simples.
- Conectivos lógicos: expresiones que sirven para formar preposiciones compuestas a partir de preposiciones simples.
- Sentencias: átomos o clausulas formadas por la aplicación de conectivos a sentencias.
- Cuantificadores: universal y existencial Э
- Símbolos de puntuación y delimitación.

Silogismos

Hipotético: A es B y B es C A es C

Disyuntivo: –A B es A B

Función de Skolem

Cuando un cuantificador existencial esta en ámbito de uno universal, la variable cuantificada debe ser reemplazada con una función de skolem: x Ǝ y (x<y) entonces y (y-1<y).

Leyes de Morgan

y

FBF se pueden expresar:

- Forma normal conjuntiva
- Forma normal disyuntiva
- Clausula de Horn: Es una conjunción de disyunciones con no más de un literal positivo (no negado). Una expresión es valida satisfacible si alguna combinación de valores v o f de sus átomos hacen verdadera la expresión.

6.2.4. REDUCCION A LA FORMA CLAUSAL

Existe un algoritmo que permite reducir los predicados a un conjunto de clausulas (afirmaciones) para que puedan ser procesadas por un mecanismo lógico. Este algoritmo parte de las llamadas *formulas bien formadas* (fbf), y reduce estas a la denominada *forma clausal* que es concretamente la expresión original transformada a un conjunto de clausulas simples

Lógica de predicados-------> FBF-------->Logica proposicional (forma clausal)

1. Eliminar implicaciones: x**-->y ≡ -x v y**
2. Reducir la aplicación de negaciones:
   -(-x) ≡ **x**
   -(x v y) ≡ x ∧ y
   -(x ∧ y) ≡ x v –y
3. Eliminar cuantificadores
4. Aplicar propiedad disyuntiva y asociativa:
   P v (Q v R)=(P v Q) v R
5. Que cada clausula tenga nombre de variable único.

6.3. RESOLUCIÓN

La demostración se lleva a cambo mediante la reducción al absurdo. El problema puede surgir con un gran número de combinaciones posibles. Es en esencia un procedimiento iterativo cuya finalidad es evaluar la verdad o falsedad de una premisa. En cada paso dos clausulas padres son resueltas para obtener una tercera denominada resolvente.

6.4. UNIFICACIÓN Y LIGADURA

Permite razonar dentro de un esquema formal sin relación con la codificación de los programas. En el caso de un código, este se encarga de reducir las formas clausales a listas asociadas mas simples de manipular denominadas ligaduras, esta fluye a través de los filtros del programa y al final del proceso pueden ser analizadas para extraer nuevas afirmaciones o demostrar hipótesis.

6.5. CORRESPONDENCIA DE PATRONES

Relacionado con la codificación de SE es la correspondencia de patrones simbólicos. La correspondencia con patrones se basa en identificar si dos vectores son similares es decir se corresponden. La forma simple de entenderlo es calcular la distancia entre los mismos y si esta es lo suficientemente pequeña de acuerdo a valores preestablecidos para un problema dado, entonces puede decirse que ambos vectores se corresponden.

6.5.1. CORRESPONDENCIA SIMBÓLICA

Busca similitudes entre un conjunto de nombres de variables y no entre los valores numéricos de estas. Se consideran dos procedimientos:

Correspondencia: encadenamiento hacia adelante, comparando una expresión común (dato) con un patrón. A diferencia de los datos los patrones pueden contener variables patrón, para facilitar el reconocimiento de estas variables se asocian a un símbolo.

Unificación: encadenamiento hacia atrás. Hace corresponder dos patrones en lugar de un patrón y un dato.

6.5.2. CORRESPONDENCIA SUB SIMBOLICA

Correspondencia numérica.

6.6. ENCADENAMIENTO.

6.6.1. PROGRESIVO

- **Correspondencia**: Busca afirmaciones en la base de datos que correspondan con los antecedentes de una regla.
- **Resolución:** En caso de que se cumplan los antecedentes instanciar el consecuente produciendo una nueva afirmación.

6.6.2. REGRESIVO

- **Correspondencia:** Busca afirmaciones en la base de datos que correspondan a una hipótesis.
- **Unificación:** buscar una regla cuyo consecuente concuerde con la hipótesis. La hipótesis y el consecuente pueden tener variables de patrón.
- **Resolución:** para cada regla cuyo consecuente concuerde con la hipótesis, se intenta verificar de manera recursiva cada antecedente de la regla, considerando a cada una como una hipótesis.

7. Capitulo 7: Lógica difusa (fuzzy)

En los sistemas binarios solo pueden ser evaluados como verdaderos o falsos, pero los razonamientos concernientes a problemas reales muestran que es posible considerar cierto grado de certeza o falsedad para una afirmación y no valores absolutos.

Para incorporar este concepto a los sistemas de inferencia y a los SE se recurre a modelos probabilísticos, es decir se asignan valores de probabilidad a las sentencias y se propagan a través de las reglas. Existen diferentes formas de calcular como influye la probabilidad de los antecedentes sobre los consecuentes.

Los campos de aplicación comunes son: control automático, clasificación, soporte de decisión, SE y visión computarizada.

7.1. LA SOLUCION FUZZY

Ante un problema que presenta diferentes grados de veracidad, seria deseable plantear la solución del problema en términos que permitan capturar la esencia del mismo y eliminar factores arbitrarios. Si damos un significado matemático a las variables lingüísticas obtendremos un sistema de inferencia basado en lógica difusa. Ej. El servicio es pobre la propina es poca, que significa una propina “poca”.

Si la lógica difusa nos permite escribir un código para la solución que resulte mas fácil de interpretar para las personas, entonces estamos diciendo que su contribución radica en el hecho de escribir un código ventajoso.

El sistema difuso esta de algún modo basado en el sentido común, por lo se pueden agregar sentencias al final de la lista que modelaran mas apropiadamente la forma de la función.

El mantenimiento del código será más simple.

7.2. VISIÓN GENERAL

La esencia del sistema radica en mapear un espacio de entrada en uno de salida mediante la inferencia de reglas. Todas las reglas se evalúan en paralelo. Las reglas de la lógica difusa se refieren a variables y adjetivos que califican a esas variables. Antes de discutir la construcción del sistema se definen los términos que se van a emplear y los adjetivos que los describen.

7.3. REGLAS SI-ENTONCES

Para lograr algo útil debemos construir sentencias del tipo si entonces, sentencias completas condicionales. La primera parte es un antecedente y la otra el consecuente. El antecedente es una interpretación que retorna un número entre 0 y 1, mientras que el consecuente es una sentencia que asigna el conjunto fuzzy completo a la variable de salida. Podríamos decir que si el antecedente es verdadero en alguna medida de pertenencia, luego el consecuente también es verdadero en el mismo grado. El antecedente puede estar compuesto por múltiples partes que son calculadas simultáneamente y resueltas como un número simple. El consecuente también puede tener varias partes.

En lógica difusa el consecuente especifica un conjunto difuso para la salida, la función de implicación modifica luego al conjunto difuso en el grado especificado por el antecedente.

7.4. REGLAS

A- Difusión de entradas: se deben resolver todas las sentencias en el antecedente en función de su grado de membrecía entre 0 y 1

B- Aplicación de los operadores difusos: si existen varias partes en el antecedente se aplican los operadores y se resuelve el antecedente como un número entre 0 y 1.

C- Aplicación de la implicación: se emplea el grado de soporte de la regla para conformar el conjunto fuzzy de salida. Si el antecedente es parcialmente cierto el conjunto fuzzy de salida es truncado según el método de implicación.

D- Los conjuntos fuzzy de salida se amalgaman en un único conjunto. Luego este conjunto resultante es defuzificado, o resuelto de modo de obtener un número como valor de salida.

7.5. PASOS

7.5.1. FUZIFICACION

Tomar las entradas y determinar el grado en el cual ellos pertenecen a cada un de los conjuntos difusos a través de funciones de pertenecía. La entrada es siempre un valor numérico limitado y la salida es un grado difuso de pertenecía. Estas funciones de pertenencia o membresía pueden ser vistas como la curva de distribución de probabilidad del adjetivo para el cual han sido definidas.

7.5.2. OPERADOR DIFUSO

Conocemos el grado en el cual cada parte del antecedente ha sido satisfecho para cada regla. Si el antecedente de una regla ha sido dividió en mas de una parte, se debe aplicar un operador difuso para obtener un número que representa el resultado del antecedente para esa regla.

La entrada del operador difuso son dos mas valores de pertenecía desde las variables de entrada fuzzificadas. Existen distintos métodos conjunciones, disyunciones y negación.

- && AND (el resultado es verdadero si ambas expresiones son verdaderas)
- || OR (el resultado es verdadero si alguna expresión es verdadera)
- ! NOT (el resultado invierte la condición de la expresión)

AND y OR trabajan con dos operandos y retornan un valor lógico basadas en las denominadas tablas de verdad. El operador NOT actúa sobre un operando. Estas tablas de verdad son conocidas y usadas en el contexto de la vida diaria, por ejemplo: "si hace sol Y tengo tiempo, iré a la playa", "si NO hace sol, me quedaré en casa", "si llueve O hace viento, iré al cine". Las tablas de verdad de los operadores AND, OR y NOT se muestran en las tablas siguientes

El operador lógico AND

x	Y	resultado
true	true	True
true	false	False
false	true	False
false	false	False

El operador lógico OR

x	Y	resultado
true	true	True
true	false	True
false	true	True
false	false	False

El operador lógico NOT

x	Resultado
true	False
false	True

Los operadores AND y OR combinan expresiones relacionales cuyo resultado viene dado por la última columna de sus tablas de verdad.

7.5.3. MÉTODO DE IMPLICACION

La conformación del consecuente basado en el antecedente. La entrdad para la implicación es un numero dado por el antecedente, y la salida es un conjunto fuzzy que se obtiene para cada regla por separado.

7.5.4. AGREGACIÓN

Unificar las salidas para cada regla uniendo los procesos paralelos. La entrada es la lista de salidas truncadas. La salida de este proceso era un conjunto difuso para cada variable de salida. La agregación es conmutativa, el orden en que se ejecutan las reglas no es relevante.

7.5.5. DEFUZZIFICACION

Ingresa un conjunto difuso agregado y sale un número. Dado un conjunto difuso que engloba un rango de valores de salida se requiere obtener un único número. Un método popular es el cálculo del centroide.

7.6. HEURISTICA

El tipo de funcion de membresia de los adjetivos empleados en las reglas debe ser propuesta por un diseñador del sistema, los valores particulares también, el método de defuzzificacion ha de ser elegido por el diseñador, etc. Todo esto no solo muestra una aplicación particular de la heurística, sino que evidencia su importancia lo que hace que muchas veces se cuestionen los fundamentos para el diseño y contstruccion.

Función de pertenencia o membresía

El concepto de función de membresía en la teoría de los conjuntos difusos es una medida de la pertenencia graduada de un elemento en un conjunto difuso.

Un elemento *u* de *U*.

Puede no pertenecer a *A*: ($\mu_A(u) = 0$),

Pertenecer un poco: ($\mu_A(u) =$ *con un valor cercano a 0*),

Pertenecer moderadamente: ($\mu_A(u) =$ *con un valor no muy cercano a 0 pero tampoco a 1*),

Pertenecer demasiado: ($\mu_A(u) =$ *con un valor muy cercano a 1*),

Pertenecer totalmente a: ($\mu_A(u)=1$).

Debido a que el cambio de la función de membresía de un conjunto a otro es gradual en los conjuntos difusos, dichos conjuntos son agrupamientos de elementos en clases, también llamados etiquetas difusas, las cuales a diferencia de los conjuntos clásicos, no poseen fronteras bien definidas.

¿Cómo se determina la forma exacta de la función de membresía para un conjunto difuso?

Una función de membresía se puede diseñar en tres formas distintas:

(1). Entrevistando a quienes están familiarizados con las conceptos importantes del sistema, y ajustándolos durante el proceso mediante una estrategia de sintonización (hasta los 80s).

(2). Construyéndola directamente a partir de los datos (2 y 3, después de los 80s).

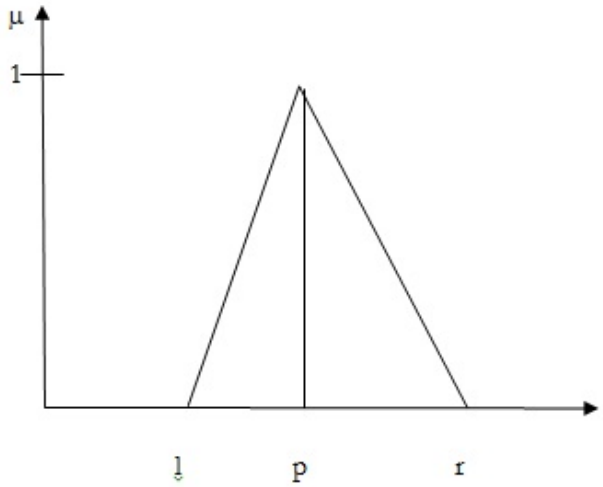
(3). Mediante el aprendizaje basado en la retroalimentación de la ejecución del sistema.

Se han desarrollado muchas técnicas para definir la forma de las funciones de membresía (FM) utilizando técnicas estadísticas, redes neuronales artificiales y algoritmos genéticos.

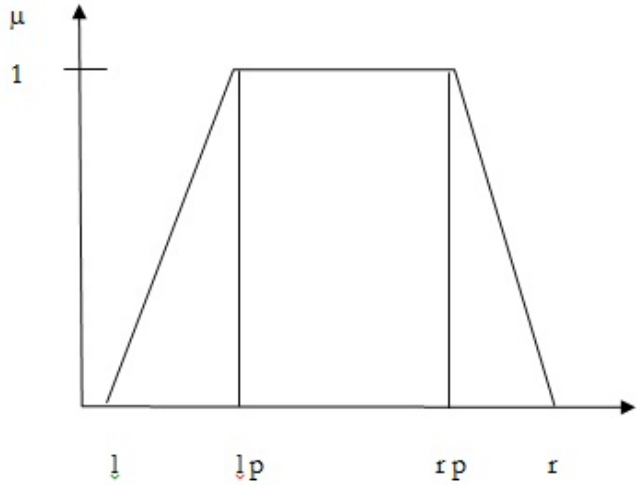
Se debe de tener especial cuidado al diseñar las FMs. Aun que se puede definir una FM de forma arbitraria, se recomienda que se utilicen FM parametrizables que puedan ser definidas por un número pequeño de parámetros.

Funciones de Membresía más utilizadas: Simplicidad

FMs más utilizadas: Simplicidad



- Función de membresía Triangular y sus parámetros.



- Función de membresía trapezoidal y sus parámetros.

Capítulo 8: Búsqueda

7.7. CONCEPTO

Procedimiento cuya finalidad es encontrar datos dentro de un conjunto de estos, pero donde el objetivo del proceso es mas bien encontrar cierto camino recorrido hasta encontrar en dato antes de corroborar la existencia del dato. Esto se debe a que estos procesos se usan en planificación, optimización, resolución de problemas, juegos, etc. Para realizar las búsquedas se utilizan árboles y grafos para conocer la secuencia de estados que deben alcanzarse sucesivamente para lograr un estado objetivo.

7.8. METODOS NO INFORMADOS

7.8.1. PRIMERO EN ANCHURA BPA

Se generan para cada nodo todas las posibles situaciones resultantes de la aplicación de todas las reglas adecuadas. Se continúa:

1. Se crea una pila (abierta) y se le asigna el estado inicial.
2. Mientras la pila no este vacía.
    a. Extraer el primer nodo de la pila llamarlo m.
    b. Expandir m. Se generan las entradas sucesoras del estado actual, para cada operador aplicable hacer:
        i. Aplicar el operador al nodo obteniendo un nuevo estado.
        ii. Si el nuevo estado es el objetivo termina el proceso.
        iii. Incluir el nuevo estado el final de la pila volver a 2ª

Ventajas: No entra en callejones sin salida, si existe una solución garantiza alcanzarla, si existen varias soluciones garantiza alcanzar la optima.

Desventajas: utiliza mucha memoria, es lento.

7.8.2. PRIMERO EN PROFUNDIDAD BPP

Continúa por una sola rama del arbol hasta encontrar la solucion o hasta que se tome la decision de terminar la busqueda por esa dirección, se puede tomar por que: se llego a un callejón sin salida, se produce un estado ya alcanzado o la ruta se alarga mas de lo especificado.

1. Se crea una pila (abierta) y se le asigna el estado inicial.
2. Mientras la pila no este vacía.
    a. Extraer el primer nodo de la pila llamarlo m.
    b. Si la profundidad de m es igual al límite de profundidad regresar a A, en caso contrario continuar.
    c. Expandir m. Se generan las entradas sucesoras del estado actual, para cada operador aplicable hacer:
        i. Aplicar el operador al nodo obteniendo un nuevo estado.
        ii. Si el nuevo estado es el objetivo termina el proceso.
        iii. Incluir el nuevo estado principio de la pila volver a 2ª

Ventajas: Si existe una solución garantiza alcanzarla, no recorre todo el árbol para alcanzar la solución, pero es por azar.

7.8.3. RAMIFICACIÓN Y ACOTACIÓN

Comienza generando rutas completas, manteniéndose la ruta mas corta encontrada hasta ese momento. Deja de explorar tan pronto la distancia total sea mayor a la marcada como la ruta mas corta.

7.9. HEURÍSTICA

Se refiere a un conocimiento que intuitivamente poseemos, que podría parecer experimental debido a la inspiración de quien lo propone. La diferencia entre lo heurístico y lo científico es que este último es un conocimiento debido a la observación, una regla empírica se cumple aunque no sepamos sus causas. En la regla heurística se espera que se cumpla, en casos particulares, y no en todos los casos en que podría aplicarse; no puede ser demostrada teóricamente, ni comprobada experimentalmente para todos los casos no se generaliza; puede resolver perfectamente un problema en particular, pero puede arrojar resultados erróneos con un problema similar.

7.10. METODOS INFORMADOS O DE BUSQUEDA HEURISTICA

La heurística es una técnica que aumenta la eficiencia de un proceso proporcionando una guía para este y posiblemente sacrificando demandas de complejidad. La búsqueda heurística resuelve problemas complicados con eficacia, garantizando una estructura de control que encuentra una buena solución.

Porque usar heurísticas:

- No enredarnos en una explosión combinatoria.
- Buscar una solución adecuada.
- Los peores casos, raramente ocurren.
- Profundizar nuestra comprensión del dominio del problema.

7.10.1. PRIMERO EL MEJOR

Combina las ventajas de BPA y BPP sigue solo un camino a la vez y cambia cuando alguna ruta parezca mas prometedora. Estrategia:

1. Se selecciona el nodo mas prometedor según la función heurística apropiada
2. Se expande el nodo elegido de acuerdo a las reglas de expansión adecuadas
3. Si alguno de los nodos es el optimo se termina, sino:
    a. Se añaden nuevos nodos a la lista
    b. Se vuelve al paso 1

*7.10.2. A**

Se modifica el anterior empleando una función heurística para estimar el costo desde un nodo actual al nodo objetivo. Realiza una estimación de cada uno de los nodos que se van agregando, esto permite examinar los caminos más prometedores.

Utiliza la siguiente función: f´=g+h´. Donde f´=estimación del costo desde el nodo inicial al nodo objetivo, g =nivel del árbol en que se encuentra el nodo, h´=estimación desde el nodo actual al nodo objetivo.

Se emplean dos listas. Abierta: los nodos que se les ha aplicado la función heurística. Cerrada: nodos que ya han sido expandidos.

Se toma el nodo más prometedor y que no haya expandido. Se generan sus sucesores y se le aplica la función heurística y se los añaden a la lista de nodos abiertos

7.10.3. GENERACIÓN Y PRUEBA

La generación de la posible solucion puede realizarse de formas diferentes. Consiste en generar la posible solucion, verficar sis es una solución, si no se hallo la solucion volver a empezar de lo contrario devolverla y terminar.

7.10.4. ESCALADA

Simple: se usa para ayudar al generador a decidirse por cual dirección moverse. La función de prueba se amplia con una función heurística que ayuda a evaluarlos estados y proporciona una estimación de lo cerca que nos encontramos del objetivo. Necesita más tiempo para alcanzar la solución.

Máxima Pendiente: es una variación del anterior, que considera todos los movimientos posibles a partir del estado actual y elige el mejor de ellos como nuevo estado. Tarda más en seleccionar un movimiento. Puede caer en un máximo local (mejor q todos pero no el mejor), meseta (estados vecinos produce el mismo valor, no se sabe a donde moverse) o cresta (no tengo operadores para ubicar la cima)

7.10.5. ENFRIAMIENTO SIMULADO

El objetivo es disminuir la probabilidad de caer en una meseta, máximo local o cresta. Por lo que se realiza una exploración con saltos amplios al principio que se van reduciendo paulatinamente. Se plantea minimizar la función objetivo, para alcanzar un estado final de mínima energía.

7.10.6. MEDIANTE AGENDA

Consideramos a cada nodo como una tarea que un sistema debe realizar. Al expandir una tarea van a existir diversos caminos. Cada camino que recomienda una tarea proporciona una razón por la cual debe ser realizada. Cuantas mas razones haya, aumenta la probabilidad de que la tarea lleve a algo adecuado. Para almacenar las tareas se propone el concepto de agenda, que es en definitiva el conjunto de tareas que el sistema debe realizar.

7.10.7. REDUCCIÓN DE PROBLEMAS

Consiste en dividir el problema en sub problemas hasta llegar al nivel de módulos. En este caso, un nodo no es mejor por si mismo, sino por pertenecer a una ruta mas prometedora.

7.10.8. VERIFICACIÓN DE RESTRICCIONES

Define los estados como un conjunto de restricciones que deben satisfacerse para resolver completamente el problema, de este modo lo que se busca es un estado que satisfaga un conjunto de restricciones impuestas por el problema.

7.10.9. ANÁLISIS DE MEDIOS Y FINES

Permite razonar tanto hacia adelante como para atrás, permite resolver las partes más importantes del problema y después volver atrás y resolver los pequeños problemas. Se centra en la detección de diferencias entre el estado del nodo actual y el nodo objetivo.

1.1.1. ALGORITMO DE RUTEO

No solo busca que existan las rutas a conectar, ni que sean las rutas mas cortas, sino que la ruta encontrada para cada par de puntos sea la ruta que menos interfiera con el menor número de rutas.

8. Capitulo 9: Planificación

8.1. INTRODUCCION

Se refiere al proceso de computar varios pasos de un procedimiento de resolución de un problema antes de ejecutar alguno de ellos.

Cuando nos enfrentamos con el problema de planificar y ejecutar una secuencia de pasos en un mundo que no es completamente predecible nos enfrentamos al problema de planificación en IA.

8.2. CLASIFICACION

Planificación en IA:

- Numérica
- Lógica: Por pila de objetivos, No lineal, Jerárquica
- Reactiva

Planificación en robótica

- Reactiva

- De trayectoria de manipuladores

Planificación industrial: métodos basados en simulación discreta

8.3. EL PROBLEMA

El problema de clasificación se caracteriza porque se puede aplicar a diversos dominios además de que no se pueden explotar todas las opciones.

Entradas: Descripción dele estado del mundo, descripción del objetivo, conjunto de acciones.

Salida: una secuencia de acciones que pueden ser aplicadas al estado, hasta alcanzar la descripción del estado final.

La lógica nos ofrece una manera de reducir los pasos de búsqueda, representando los estados por predicados aplicando reglas lógicas para pasar de uno a otro, dicha aplicación debería modificar el estado anterior definiendo nuevos estados. La regla en si nos dice que esta permitido pasar de un estado a otro pero no efectúa el cambio necesario para generar este paso. Para que se modifique el estado actual es necesario aplicar procedimientos (operadores).

8.4. MUNDO DE LOS BLOQUES

Para poder hacer un estudio sistemático de los métodos y poder compararlos usamos el mundo de los bloques. Se basa en una superficie palana en la que se colocan bloques (de forma cubica únicamente). Los bloques se pueden ubicar unos sobre otros y se pueden llevar a cabo acciones que pueden manipular los bloques (operadores asociados a las reglas).

8.5. CODIFICACION DE LOS OPERADORES

Para poder pasar de un estado a otro modificando predicados usamos operadores, cada operador puede describirse mediante una lista de nuevos predicados que el operador provoca que sean ciertos y una lista de los viejos predicados que el operador provoca que sean falsos.

8.6. PLANIFICACION MEDIANTE UNA PILA DE OBJETIVOS

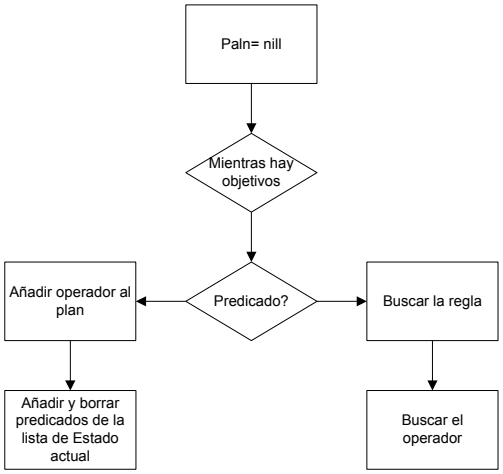
Elementos: Bloques, superficie y brazo operador.

Estados de los elementos: estado del bloque y del brazo.

Operadores: acciones a realizar. Se componen de tres listas: precondición (lo que se debe cumplir para aplicar el operador), Borrado (lo que se debe eliminar de la pila de objetivos), Adición (lo que se agrega al plan de tareas)

Pilas: en todo momento se cuanta con una pila de objetivos y operadores, una de estado actual y la del plan a seguir.

Procedimiento: se tratan los objetivos como sub problemas a resolver. Se pregunta si el primer objetivo se cumple. Si no se cumple se debe aplicar el operador que lo transforme, por lo que se debe reemplazar al predicado objetivo por el operador y luego agregar a la pila las precondiciones que deben cumplirse para ese operador. Para resolver el objetivo que ha quedado arriba de la pila empleamos el operador reemplazando y agregando las precondiciones. Al ir extrayendo los operadores de esa pila, se incorporan a la lista plan y se actualiza el estado actual.



Complejidad

Desde el punto de vista informático, la complejidad se asocia con los recursos de cómputos requeridos para resolver un problema, es decir la complejidad operativa. Una rama de las ciencias de la computación se encarga de estudiar tal complejidad, que hacen referencia a espacio y tiempo.

Complejidad temporal

Uno de los indicadores de complejidad más utilizados es el **tiempo**. Se refiere a la cantidad de intervalos o unidades elementales que demanda en completar la ejecución de un proceso. Es de gran importancia conocer la razón de crecimiento entre el indicador tiempo y la relación de los datos, que representa el parámetro medible. Como resultado se hablará de complejidad temporal lineal, polinómica, exponencial, etc., según la naturaleza de la expresión que las vincula.

El límite de crecimiento de esta medida se denomina complejidad temporal asintótica y es finalmente lo que determina el tamaño del problema que puede ser resuelto por cierto algoritmo.

Complejidad espacial

A medida que aumenta la complejidad de un proceso aumenta también el espacio que demanda y este parámetro es conocido como la **complejidad espacial.**

Medición de la complejidad y computadores reales

Para ello deben especificarse el tiempo que demanda individualmente cada instrucción y el espacio de almacenamiento requerido por cada registro. Una opción es el denominado “criterio de costo uniforme” que asigna a cada instrucción el consumo de una unidad de tiempo y a cada registro utilizado en una unidad de espacio.

Un criterio más realista y específico es el de “costo logarítmico”. Con este criterio se considera la incidencia del largo de los números representados en los operandos en tiempo de ejecución de cada operación.

FUNCIONES BASICAS

a) Resolución de Sistema de Ecuaciones Lineales por eliminación de Gauss

$$T(n) = \frac{2}{3}n^3 + \frac{3}{2}n^2 - \frac{7}{6}n = O(n^3)$$

Complejidad Temporal: polinomial (3er grado).

b) Cálculo del determinante de una matriz

$$T(n) = (n!)\sum_{k=1}^{n-1}(\frac{1}{k!}) + n! = O(n!)$$

Complejidad Temporal: ≈ exponencial (factorial)

c) Ordenamiento de un vector por inserción

$$T(n) = \frac{3}{4}n^2 + \frac{7}{4}n = O(n^2)$$

Complejidad Temporal: polinomial (2º grado).

e) Ordenamiento de un vector: método shell sort

$$T(n) = O(n\ \log(n))$$

Complejidad Temporal: logarítmica

f) Ordenamiento de un vector: método quick sort

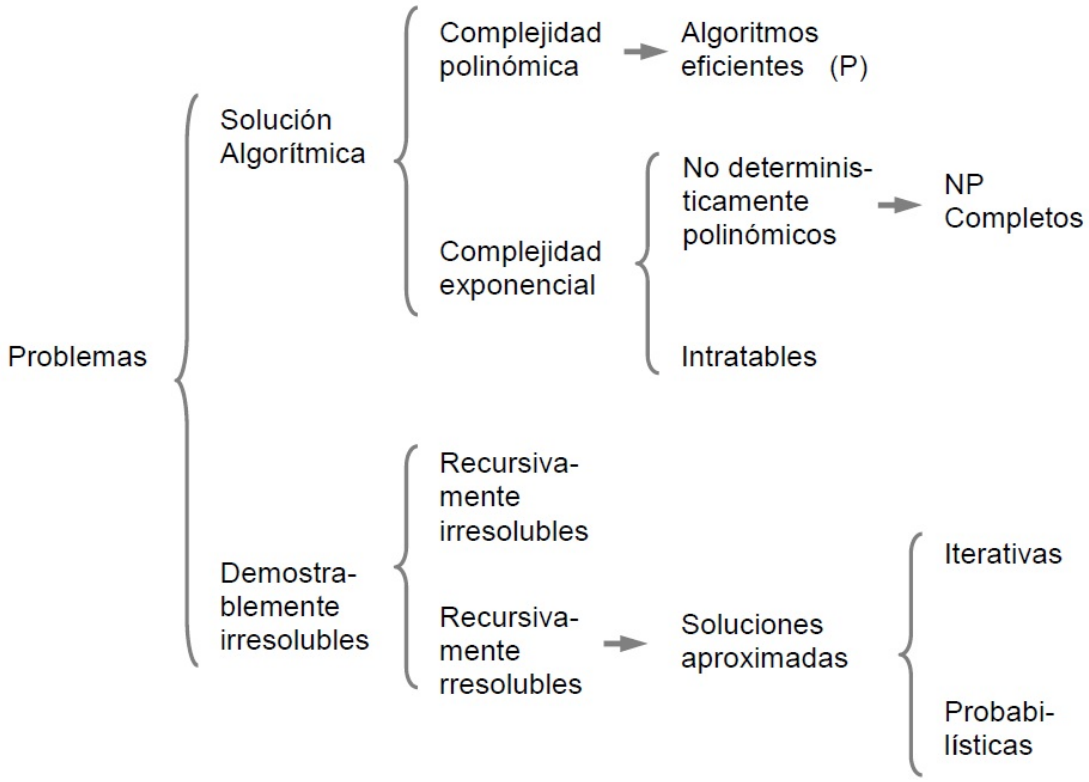
$$T(n) = O(n\ \log(n))$$

Complejidad Temporal: logarítmica

Clases de problemas

Todos los problemas matemáticos pueden ser divididos en 2 grupos: los que admiten un algoritmo para su solución y los demostrablemente irresolubles.

Al grupo de soluciones algorítmicas se los puede dividir a su vez en dos grupos: 1) los problemas de complejidad temporal polinómica y 2) los de complejidad temporal exponencial. Los problemas de complejidad temporal polinómica tienen su tiempo de ejecución acotado por una función de tipo y son considerados “eficientes”. Cuando se trata de problemas de complejidad temporal exponencial, es decir intratables, sólo puede llegarse a una solución cuando la dimensión de datos “n” es pequeña.



La clase de complejidad **P** es el conjunto de los problemas de decisión que pueden ser resueltos en una **máquina determinista** en tiempo polinómico, mientras que la clase de complejidad **NP** es el conjunto de problemas de decisión que pueden ser resuelto por una **máquina no determinista** en tiempo polinómico.

Complejidad en los algoritmos de búsqueda

Se toman aquí como parámetros característicos del problema el factor de ramificación medio “R” y el nivel de profundidad alcanzado en el árbol de búsqueda “P”.

Tabla 5: Orden de Complejidad de los principales métodos de búsqueda

Método	Complejidad Temporal	Complejidad Espacial
Primero en anchura	R^p	R^p
Primero en profundidad	R^p	p.R
Descenso iterativo	R^p	p.R
Máxima pendiente	R^p	p.R
Primero el mejor	$\leq R^p$	$\leq R^p$
A*	$\leq R^p$	$\leq R^p$