Redes Neuronales

**Perceptron:** Se trata de una red que al recibir el vector de entrada calcula la función de decisión lineal y le aplica una función umbral que fuerza a la red a obtener solo dos valores posibles a la salida. La diferencia fundamental está en la forma de calcular el vector de pesos W, que no se hace por cálculos matriciales sino por un método iterativo Ley de aprendizaje: Es una función recursiva que permite obtener el valor apropiado para la Matriz de Coeficientes.

**Leyes de aprendizajes**

1. **Ley de Hebb:** Representa un modelo simplificado del funcionamiento de una neurona biológica. El valor de la Matriz de Coeficientes W se ajusta según la variación dada por esta ley: ∆Wij = lr \* Xi \* Yj → lr = Tasa de aprendizaje

Los parámetros se modifican según: Wij = Wij + ∆Wij , bij = bij + ∆bij .

1. **Ley de aprendizaje del perceptrón:** Esta considera que le producto de Hebb no interviene directamente la salida sino la diferencia entre la salida obtenida al procesar la información y la salida correcta que debería haberse generado, es decir: ∆Wij = Xi (Yj - Yj). Los nuevos pesos se calculan de la siguiente manera:

formula perceptron.png

**El perceptrón y la clasificación no lineal**

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.

**Adaline**

Este método surge a partir de una mejora reemplazando la función de salida abrupta del perceptrón por una función rampa. Esto permite generalizar el perceptrón para e/s continuas, y permite aplicar la técnica del gradiente o descenso iterativo. La ley de aprendizaje que corresponde a este modelo se denomina Regla Delta o Ley de Widrow/Hoff. Ley de Hebb ∩ Perceptrón = ∆Wij = lr \* (Yj - Yj) \* Xi.

Para interpretar el sentido de la búsqueda por iteración descendente del valor de W se considera el cuadrado del error cuadrático medio. Este error se considera para un caso unidimensional, donde la gráfica 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, es decir que hace mínimo el error.

Generalmente se trabaja sobre superficies muy irregulares, por ende hay que conformarse con encontrar un mínimo local “suficientemente bueno”, es decir, que nos permita realizar la clasificación del problema dado. Podemos destacar que los pesos se ajustan simultáneamente, y que un valor alto de la tasa de aprendizaje lr implica saltos oscilatorios, que en general impiden la convergencia hacia una solución, o bien lr pequeña puede demorar demasiado el proceso.

**Retropropagación**

Ley de aprendizaje por retropropagación: implica emplear una red multicapa de perceptrones o adalinas para resolver problemas no lineales. 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.

**Sobreentrenamiento:** Este es un comportamiento propio de la redes de retropropagación cuya manifestación es que luego de un período de entrenamiento normal, el error tiende a aumentar nuevamente.

**Esqueletonización:** Este proceso se aplica para reducir la cantidad de neuronas de una red una vez que esta ha sido ya diseñada y entrenada. Esto se hace para reducir la cantidad de memoria necesaria para almacenar la red. La mayoría de métodos se basan en descartar aquellas neuronas que contribuyen menos a una correcta clasificación. Otro método sencillo, es ir quitando neuronas y ver cuál es el error que produce. En base a esto, luego se va eligiendo que neurona retirar de la red.

**Pasos del algoritmo de retropropagación:**

1. Inicializar los pesos a valores pequeños al azar.
2. Elegir un patrón y aplicarlo a la capa de entrada.
3. Propagar la señal hacia delante de la red.
4. Computar los deltas de la capa de salida.
5. Computar los deltas de las capas previas propagando el error hacia atrás.
6. Actualizar todas las conexiones.
7. Volver al paso 2 y repetir con el patrón siguiente.

**Percepción Visual**

Luego de haber visto el método de convolución de imágenes, decimos que el objetivo de una operación de suavizado es moderar los gradientes de la zona de alto gradiente. Es decir, hacer los bordes menos nítidos, o la imagen más borrosa. El método principal de suavizado es simplemente la función Gausiana, es decir los valores de la máscara de convolución corresponden a valores discretos de la campana de Gauss.

**Pirámide Gausiana:** Debido a que con cada aplicación de este filtro se pierden detalles de la imagen, es frecuente reducir la dimensión de la imagen obtenida, quitando algunos píxels a la imagen que se aplico el suavizado. Este procedimiento consiste en lograr un conjunto de imágenes cada vez menos nítidas y eventualmente de menor tamaño.

**Teoría del espacio escalar:** Según esta teoría es que los sistemas de visión biológicos reconocen los objetos en diferentes escalas. De este modo un sistema puede reconocer “características” que definen un objeto en diferentes escalas y componerlas de modo de reconocer el objeto.

Reconocimiento por Invariantes: Este método se basa en extraer características (en lo posible invariantes a la rotación y otras transformaciones) de un objeto, y almacenarlas asociadas por un índice o etiqueta a ese objeto, conteniendo información acerca de cada característica.

Cuando se examina una imagen, se comparan las características detectadas con las que corresponden a las que describen los objetos almacenados. Se prueba luego la correspondencia de las que corresponden a los objetos más prometedores, y eventualmente se reconocen si aparecen en la imagen.

**Sistemas Expertos**

Un sistema experto es un programa destinado a generar inferencias en un área específica del conocimiento en una forma similar a la que se espera de un experto humano.

Los SE deben resolver problemas por aplicación de conocimiento de un dominio específico.

Algunas características del tipo de problema resulto por un sistema experto son:

* Inexistencia de un algoritmo para hallar la solución
* El problema es resuelto satisfactoriamente por expertos humanos
* Posibilidad de contar con un experto humano para colaborar en el desarrollo
* El conocimiento experto del dominio debe ser relativamente “estático”

Componentes de un SE:

* Símbolos
* Reglas a aplicar sobre los símbolos definidos. Estas se deben obtener del conocimiento de los expertos humanos.
* Los Hechos son los predicados que se suponen verdaderos. Constituyen la base de datos de conocimientos (KDB).

Al expresar el experto humano su conocimiento en lenguaje natural, existe la necesidad de que quien implemente el SE, deba traducir los conocimientos expresados en lenguaje natural por el experto humano a las expresiones lógicas correspondientes, para que estas puedan ser procesadas por el SE.

Se debe recordar que sólo se intenta aplicar los SE a mundos reducidos. Sólo pueden aplicarse prácticamente a mundos de muy baja complejidad, en el sentido de la cantidad de símbolos y predicados que deben manejarse.

Existen dos modelos fundamentales de sistemas expertos:

* SE con encadenamiento hacia adelante
* SE con encadenamiento hacia atrás

Elementos básicos del lenguaje lógico

* Una *proposición* es una sentencia simple que tiene un valor asociado ya sea de verdadero o falso.
* *Átomos*: son proposiciones simples
* *Conectivos lógicos*: Son expresiones que sirven para formar preposiciones compuestas, a partir de preposiciones simples
* *Sentencias*: Átomos o cláusulas formadas por la aplicación de conectivos a Sentencias. Se expresan mediante enunciados y oraciones declarativas. Dichas proposiciones están unidas mediante conectivas lógicas.

**Importancia de las cláusulas de Horn**

Para aplicar la lógica en programas de computador ha sido necesario encontrar algoritmos que determinen si una expresión lógica válida es “satis factible”, es decir si alguna combinación de valores verdaderos o falsos de sus átomos hace verdadera a la expresión. Es por esta razón que se utiliza un algoritmo más rápido si la expresión está en la forma de cláusula de Horn.

Es necesario que no haya dependencia lógica entre los literales para que el método de resultados correctos.

**Reducción a la forma clausal**

1. Eliminar las implicaciones
2. Reducir la aplicación de la negación (por Morgan o doble negación)
3. Eliminar cuantificadores
4. Aplicar la propiedad distributiva y asociativa
5. Que cada cláusula tenga nombre de variable único

**Resolución**

El procedimiento de la resolución es en esencia un proceso iterativo cuya finalidad es evaluar la verdad o falsedad de una premisa.

* Para cláusulas P y –P el resolvente es la cláusula vacía.
* Si un conjunto de sentencias contiene ambos P y –P el conjunto es insatisfactible.
* Si ambos antecedentes son verdaderos luego el resolvente es verdadero.
  + La resolución verifica la refutación, es decir: “Si el conjunto de cláusulas implica a P luego es contradictorio con el mismo conjunto que además contenga a -P”.

**Inferencia y probabilidad. Lógica Difusa.**

La esencia de un sistema fuzzy es mapear un espacio de entrada en uno de salida. El mecanismo primario para lograrlo es la inferencia basada en reglas. Las reglas en lógica difusa se refieren a variables y adjetivos que califican a esas variables.

**Reglas Si – entonces**

Si el servicio es bueno luego la propina es media.

La parte primera de la regla es el antecedente o premisa, mientras que la parte final “y es B” es el consecuente o conclusión.

La interpretación de estas reglas comprende distintas partes: 1) evaluación del antecedente (lo que implica difusión de la entrada y aplicación de operadores difusos) y 2) aplicación del resultado al consecuente (implicación).

Si relajamos las restricciones de la lógica binaria y permitimos que el antecedente sea una sentencia fuzzy, tendremos que considerar como influye en la conclusión.

Tanto el antecedente como el consecuente de una regla pueden estar compuestos por múltiples partes.

**Sumario de reglas**

1. *Difusión de entradas*: Si el antecedente tiene sólo un componente este es el grado de soporte del consecuente.
2. *Aplicación de operadores difusos*: Si existen múltiples partes en el antecedente se aplican los operadores fuzzy y se resuelve el antecedente como un número entre 0 y 1. Esto es el grado de soporte de la regla.
3. *Aplicación de la implicación*: Se emplea el grado de soporte de la regla para conformar el conjunto fuzzy de salida. El consecuente de una regla fuzzy asigna un conjunto fuzzy completo a la salida. Si el antecedente es solo parcialmente cierto, luego el conjunto fuzzy de salida es truncado según el método de implicación.
4. En general, una sola regla no es muy útil. Y por lo tanto se requieren dos o más reglas que interactúen entre ellas. La salida de cada regla es un conjunto fuzzy, pero en general se necesita que la salida sea en definitiva un número. Para obtener este número, primero los conjuntos fuzzy de salida de cada regla se amalgaman en un único conjunto. Luego este conjunto resultante es defuzificado, o resuelto de modo de obtener un número como valor de salida.
5. **Fuzificación**

El primer paso es tomar las entradas y determinar el grado en el cual ellos pertenecen a cada uno de los conjuntos difusos a través de funciones de pertenencia.

1. **Aplicación del operador difuso**

Si el antecedente de una regla dada ha sido dividido 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 o más valores de pertenencia desde las variables de entrada fuzzicadas. La salida es un único valor de entrada.

|  |  |  |
| --- | --- | --- |
| **Operador** | **Método** | **Descripción** |
| P ˄ Q | Mínimo | Mínimo(XP, XQ) |
| P ˄ Q | Producto | XP \* XQ |
| P ˄ Q | Truncamiento | Max(XP + XQ – 1, 0) |
| P ˅ Q | Máximo | Max(XP, XQ) |
| P ˅ Q | Amplificación | XP + XQ \* (1 - XP) |
| P ˅ Q | Adición | Min(XP + QX, 1) |
| ¬P | Complemento | 1 - XP |

1. **Aplicación del método de implicación**

La entrada para la implicación es un número dado por el antecedente, y la salida es un conjunto fuzzy, la que se obtiene para cada regla por separado.

Algunos métodos podrían ser Min() (mínimo) que trunca el conjunto fuzzy, y Prod() (producto) el que amplifica el conjutno de salida fuzzy.

1. **Agregación**

Es el proceso de unificar las salidas para cada regla uniendo los procesos paralelos.

La entrada para la agregación es la lista de salidas truncadas retornadas por el proceso de implicación para cada regla. La salida de este proceso era un conjunto difuso para cada variable de salida.

La agregación es conmutativa, luego el orden en que se ejecutan las reglas no es relevante.

1. **Defuzzificación**

La entrada para el proceso de defuzzificación es el conjunto difuso agregado y la salida es un número.

Un método de defuzzificación que se usa generalmente es el cálculo del centroide, el cual retorna el centro de un área bajo una curva.

La defuzzificación se realiza en dos pasos. Primero las funciones de pertenencia son escaladas de acuerdo a sus posibles valores, luego estas son usadas para calcular el centroide de los conjuntos difusos asociados.

**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.

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.**

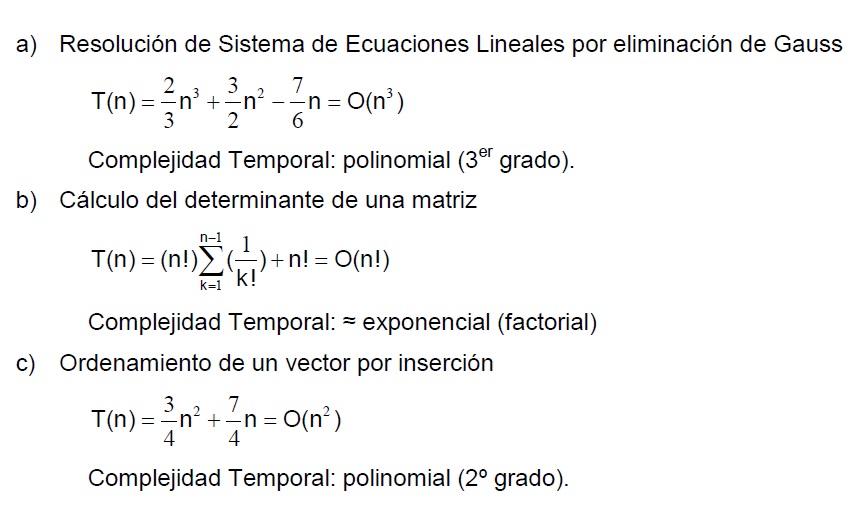
**Definición de complejidad de la máquina de Turing**

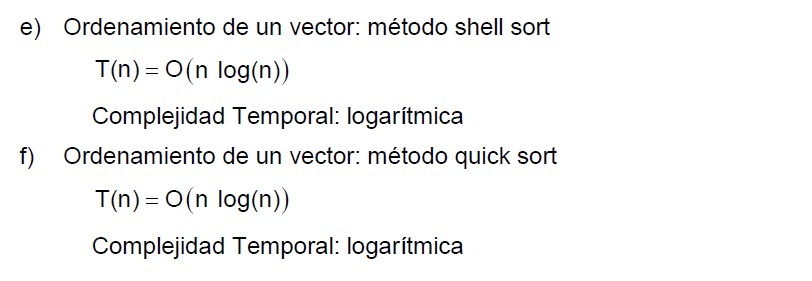
En una máquina de Turing se define como complejidad temporal como el número de pasos requerido para completar el cálculo y la complejidad espacial al número de celdas requeridas de la cinta de E/S.

**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.

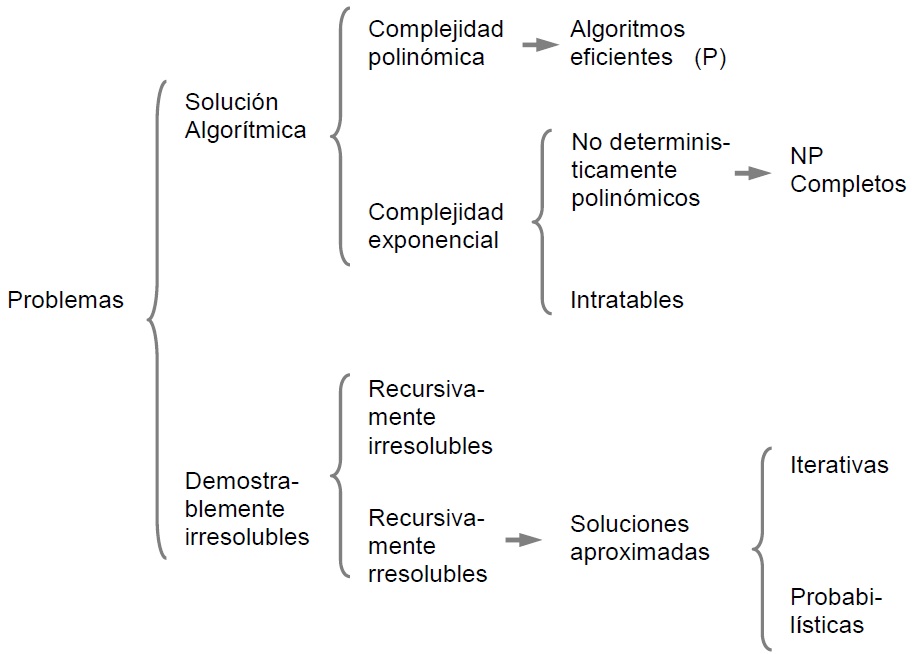




**Clases de problemas**

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 e inteligencia artificial**

Los trabajos de Leung han determinado la complejidad algorítmica del proceso de entrenamiento por backpropagation en ciertas condiciones y esto permite acotar la complejidad temporal del problema con el criterio del costo uniforme.

La complejidad temporal del proceso de entrenamiento de redes multicapa de Perceptrones es de tipo polinómica, de grado entre 3 y 5. Se trata de un problema algorítmico considerado eficiente.

**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”.



La complejidad de todos los métodos tiene un límite exponencial. Más aún, la resolución de este tipo de problemas de la IA suelen ser caracterizado como NP-completo, pero como mínimo es un problema NP. Como consecuencia, se justifica la necesidad de una buena heurística que restrinja en todo lo posible la zona del espacio de estados en que se desarrollará el proceso de búsqueda de la solución.

En efecto, la delimitación de la zona de trabajo en el espacio de estados es el efecto buscado por los algoritmos Primero el Mejor y A\*, con los que se pueden reducirse notablemente los indicadores de complejidad hasta convertirlos en algoritmos eficientes.