APLICACIONES DE LAS REDES NEURONALES EN ROBÓTICA

1.INTRODUCCIÓN :

La programación de robots suele hacerse en relación a las coordenadas cartesianas del espacio de trabajo del robot , por lo tanto caerá en el controlador la tarea de traducir dichas coordenadas en variables articulares o motoras que gobernaran las movimientos del robot. Por lo tanto , el control de robots depende de la disponibilidad de funciones que permitan pasar del espacio físico al espacio de variables articulares o motoras.

Una función básica es la denominada *cinemática inversa* ,que hace corresponder una posición y orientación del elemento terminal a cada vector de valores de las variables articulares. Otra función es la llamada *dinámica inversa* que relaciona la trayectoria del elemento terminal con las fuerzas y pares ejercidos en las distintas articulaciones. También existe el llamado *mapa sensoriomotor* , que relaciona patrones sensoriales (recogidos por algún sensor en el campo de trabajo) con las ordenes motoras necesarias , un caso particular es la evitación de obstáculos por el robot móvil.

Estas funciones tienen el problema de ser altamente no lineales , además , debido a cambios en el entorno y al desgaste del propio robot , estas funciones pueden ser variantes en el tiempo , siendo deseable que la estructura de control se adapte a las variaciones , por todo esto las redes neuronales ofrecen una buena respuesta a este tipo de aplicaciones.

2.REGLAS DE APRENDIZAJE USADAS EN ROBÓTICA :

Las clasificaremos por su tipo de realimentación requerido durante el entrenamiento :

a) reglas correccionales :

No usan ninguna realimentación derivan de la regla hebbiana , usadas en modelos de aprendizaje adaptativo, modelos de resonancia adaptativa , y mapas auto−organizativos de Kohonen.

b) reglas de minimización del error :

Requieren una señal de error exacta que usa como realimentación. Esto quiere decir que precisan conocer las salidas que corresponden a las entradas. Dos reglas de este tipo son la *regla LMS* y la *regla de retropro− pagación del error*, siendo esta última la más conocida y usada, aunque su velocidad de convergencia sea baja y de que (debido a que el error cuadrático medio no permite el aprendizaje incremental) al aprender un nuevo par de e/s , necesariamente se produce un cierto olvido de los pares previamente aprendidos.

c) reglas basadas en una señal de refuerzo :

La señal de refuerzo proporciona información de cómo ha sido de buena

o mala la respuesta a un determinado estímulo , generalmente adopta

valores entre un máximo (premio) y un mínimo (castigo). Lo que se

busca es maximizar el refuerzo total obtenido durante la realización de

una determinada tarea , es decir , que para un estimulo , la respuesta sea

la mejor posible.

3.<u>UTILIZACIÓN DE LAS REDES NEURONALES EN CONTROL</u> :

Un sistema que hay que controlar , generalmente tiene una función de transición *f* y una función de salida *g*. Habrá una señal de control u(t) , que con el estado actual x(t) , determina el estado siguiente x(t + 1) :

x(t + 1) = f (u(t) , x(t))

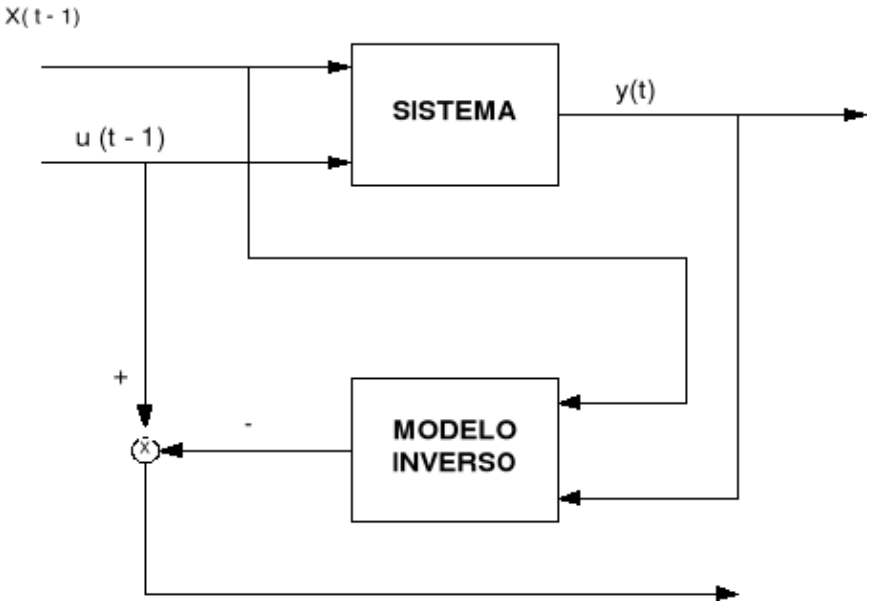
En cada estado x(t) , se produce una salida y(t) : y(t) = g(x(t))

Un controlador equivale al modelo inverso del sistema a controlar puesto que , dada una salida deseada y el estado actual , el controlador generara la señal de control que producirá dicha salida.

Esquemas de control :

a) modelización inversa directa : es el esquema mas simple. Usa el propio sistema para generar pares de entrada−salida y entrena el modelo inverso

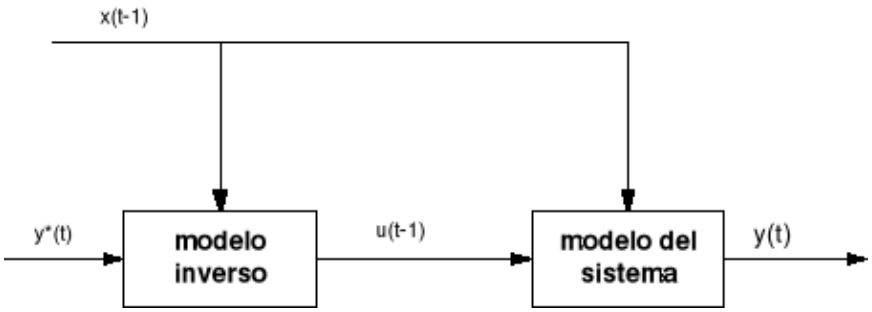
directamente mediante el intercambio de entradas y salidas :



Este esquema solo es aplicable a sistemas con funciones inyectivas ya que

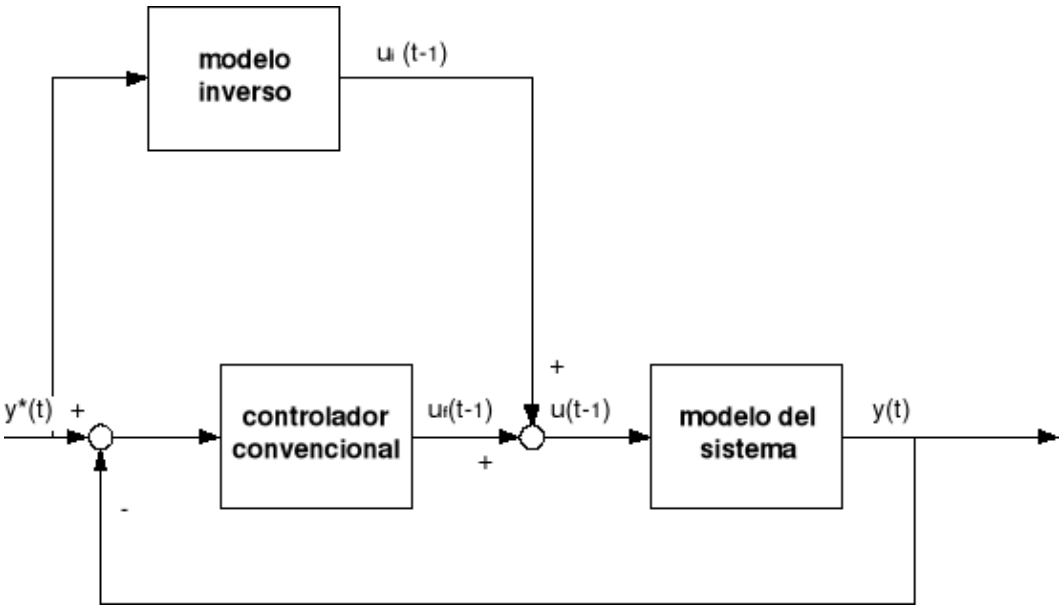
sino la inversa no es una función , y su éxito depende del muestreo.

b) modelización hacia delante : procede en dos etapas. Primero se hace que la red aprenda el modelo del sistema a partir de pares e/s. En la segunda etapa se antepone dicho modelo a otra red neuronal y se entrena el sistema formado por ambas para que aproxime la función identidad , manteniendo los pesos del modelo fijos , ya que serán lo pesos del controlador los que se ajustan.



Siendo y*(t) la salida deseada.

c) aprendizaje basado en la señal f de error realimentada : requiere disponer de un controlador convencional conectado al sistema , siendo la función de este la de hacer que la señal de error tienda a cero. Este enfoque no precisa fase de aprendizaje previa.



4.<u>CINEMÁTICA INVERSA EN UN ROBOT MANIPULADOR</u> :

El control de robots , debe disponer de funciones precisas que hagan corresponder a una posición y orientación cualesquiera del espacio , los valores de las variables articulares que hacen que el elemento terminal del robot alcance la posición y orientación mencionadas. La utilización de redes es de especial interés cuando no se dispone de un modelo preciso de algunas de las articulaciones , o cuando estos por alguna razón no se pueden calibrar para ajustar sus movimientos , ya que la capacidad de aprendizaje brinda la oportunidad

de resolver dichos problemas.

4.1 REDES MULTICAPA CON RETROPOPAGACIÓN DEL ERROR :

La manera mas sencilla de abordar el aprendizaje de la cinemática inversa es aplicar el esquema de *modelización inversa directa* a una red multicapa que incorpora la *regla de retropopagación del error*. La manera de asegurar que la función sea inyectiva, es elegir para el conjunto de aprendizaje solo configuraciones del robot en las cuales las articulaciones del robot permanezcan siempre en el mismo semiespacio , lo que equivale a que no se salgan de un rango determinado.

Jordan y Rumelhart aplicaron en 1992 el *esquema de modelizacion hacia delante* al aprendizaje de la cinemática inversa de un robot manipulador con 3 articulaciones que se mueve en un plano. Aunque este caso de la cinemática inversa no es una función , sino una correspondencia de 2 a 3 variables.

La conclusión después de experimentar con redes multicapa es que puede obtenerse rápidamente una aproximación burda de la cinemática inversa , y sin embargo una representación precisa de la función es muy difícil , esto se debe a que la aproximación a la función se ve influenciada por todos y cada uno de los pesos de la red.

Una manera de resolver este problema es usar representaciones locales , de manera que cada parte de la red sea responsable solo de una pequeña porción del espacio de entrada.

4.2 MAPAS TOPOLOGICOS AUTO−ORGANIZATIVOS :

Se trata de un esquema basado en una *regla correlacional* , que sigue un aprendizaje con la regla de *minimización del error LMS* . El esquema de control seguido es el de *modelización inversa directa* con funcionamiento no supervisado.

Las neuronas están organizadas en una malla tridimensional que , mediante el aprendizaje , evoluciona hacia una representación del espacio de trabajo del robot. Las neuronas de las neuronas serán las coordenadas u(t) del punto que se pretende alcanzar con el elemento terminal. Las salidas , una vez entrenada la red , son las variables articulares t y los jacobianos Ai correspondientes a dicho punto del espacio.

Dada una entrada u , la red determina la neurona k que mejor correlaciona con dicha entrada , según la ecuación :

" wi k ui−−>[Author:ADG] " " wi j ui "j

i i

La salida se calcula entonces mediante la ecuación :

 (x) = k + Ak (u − wk)

Un ciclo de aprendizaje comprende los siguientes cuatro pasos :

1) En primer lugar , se aplica la regla de aprendizaje de Kohonen a los pesos de la red :

wi j (t + 1) = wi j (t) + chk (j) (uj (t) − wi k (t))

donde hk ( ) es una función gaussiana centrada en k , que se usa para modular la adaptación de los pesos de entrada a una neurona en función

de su distancia a la neurona activa.

2) Moviendo el robot a las coordenadas articulares (u) obtenidas mediante la ecuación anteriormente expuesta , el elemento terminal se sitúa en la posición

u' . La diferencia entre la posición deseada u y la realmente alcanzada u' constituye una señal de error , que permite aplicar una regla de minimización del mismo , en este caso la regla LMS :

* = k + = k + Ak (u − u')

3) Aplicando el incremento de corrección Ak (u − u') a las articulaciones del robot , se ajusta la posición del elemento terminal obteniéndose las nuevas coordenadas u". A continuación se aplica la regla LMS al jacobiano, usando

u = (u − u") como señal de error :

A* = Ak + (− Ak u) u / || u ||

4) Finalmente , se utiliza la regla de kohonen para actualizar los valores articulares :

i = i + c'hk (y) (y* − k (t))

y la matriz jacobiana :

Ai = Ai + c'hk' (y) (Ai* − Ak (t))

donde c' es una constante que determina la velocidad de aprendizaje

y h'( ) es también aquí una función gaussiana centrada en k , utilizada

para modular los pasos de aprendizaje en función de la distancia a la neurona activa k.

La amplia experimentación llevada a cabo por los distintos autores muestra que la red se auto−organiza , constituyendo una representación adecuada del espacio de trabajo , en aproximadamente 30.000 iteraciones del ciclo de entrenamiento. Este resultado muestra la gran capacidad de aprendizaje de este esquema , puesto que las condiciones de operación a las que se ha sometido la red son las peores imaginables : completo desconocimiento del robot , inicialización aleatoria de los parámetros y muestreo aleatorio del espacio de trabajo durante el entrenamiento.

5. <u>DINÁMICA INVERSA</u> :

Por ésta se entiende la función que, dad una trayectoria deseada del elemento terminal del robot, proporciona las fuerzas y pares que deben ser aplicados a las distintas articulaciones para que efectivamente se siga dicha trayectoria.

Se suelen aplicar técnicas de control adaptativo, que no requieren conocimiento a priori de esta función, al control dinámico de robots. Sin embargo, el coste computacional de estas técnicas, aumenta rápidamente con el número de variables de estado, haciendo prohibitiva en muchos casos su aplicación en tiempo real.

Recientemente, se han utilizado modelos de Redes Neuronales, para el aprendizaje de la dinámica inversa bien directamente o a través del error proporcionado por un controlador convencional de ganancia fija, como se describirá en los dos subapartados que siguen. Conviene destacar que el caso dinámico difiere del

cinemático en que, para generar los pares de E/S, se precisa algún tipode controlador que guíe al robot desde el principio. A medida que progresa el aprendizaje de la función dinámica inversa , la salida de la R.N deviene más precisa y el efecto del controlador tiende progresivamente a cero.

5.1 RED CMAC.

Adaptaciòn del modelo CMAC (Cerebellar Model Articulation Controller), desarrollado por Abus (1975), al control dinámico de un control industrial con cinco grados de libertad. La adaptación consiste en combinar la rapidez de acceso a tablas proporcionadas por la red CMAC con un procedimiento de corrección de errores similar a la *regla* LMS.

Aquí el CMAC se utiliza para representar el espacio de estados de forma compacta y localizada (tal como los mapas auto−organizativos de Kohonen se utilizaban para modelar el espacio de trabajo del robot) y, mediante el procedimiento de correción de errores, los pesos sinápticos se modifican proporcionalmente al la diferencia entre la salida deseada y la obtenida mediante la red (mientras que la regla LMS se utilizaba para estimar la salida deseada en caso cinemático).

La tarea consiste en enseñar a un robot a seguir una determinada trayectoria. Para ello, se muestran puntos sucesivos de la trayectoria tanto a la red neuronal como al controlador de ganancia fija y se usa la suma de sus respuestas como consigna para guiar el robot. Después de cada ciclo, la consigna junto con el estado del robot se utilizan como par de entrada−salida para entrenar la red neuronal, siguiendo el esquema de *modelización inversa directa.*

A medida que progresa el aprendizaje, la red CMAC aproxima la función dinámica inversa, de modo que la diferencia entre el estado actual y deseado tiende a cero y, por consiguiente, la red neuronal toma el control que antes tenía el controlador de ganancia fija.

La red converge a un error pequeño (entre 1 y 2 unidades de los codificadores de posición) en 10 intentos, suponiendo que se utilicen un número suficiente de pesos. Puesto que el modelo CMAC se basa en el acceso a tablas, los resultados demuestran una gran dependencia respecto del número de pesos utilizados.

5.2 Aprendizaje a partir del error realimentado.

La misma tarea de aprendizaje de una trayectoria que acabamos de describir ha sido abordada bajo el esquema del *aprendizaje basado en la señal de error realimentada.* El robot utilizado tiene tres grados de libertad y la red neuronal esta formada solo por 3 neuronas (una por articulación) cuyas entradas son 13 funciones no−lineales de las velocidades y aceleraciones de las articulaciones. Así pues, 39 pesos son repetidamente modificados utilizando una regla de corrección de errores similar a la *regla LMS*.

El proceso de entrenamiento es : se proporciona la trayectoria deseada tanto a la red neuronal como al controlador de ganancia fija y la suma de sus respuestas se usa como consigna para gobernar el robot.

En este modelo no se generan pares de entrada−salida, sino que usan la salida del controlador de ganancia fija (que, de algún modo mide la desviación del estado actual respecto del deseado en términos de la señal de control necesaria para que el primero se aproxime al segundo) directamente como señal de error. Esto tiene la ventaja de ser directamente accesible en el bucle de control, obviádose así la necesidad de terminar el estado actual del robot. Si se considera este error como una medida cualitativa en lugar de cuantitativa, entonces se convierte en una especie de refuerzo y este esquema puede pensarse como aprendizaje basado en una señal de refuerzo.

Después de entrenar 300 veces al robot a seguir una trayectoria de 6 segundos de duración, el par medio realimentado decrece dos órdenes de magnitud, demostrando que la red neuronal ha tomado el control en

detrimento del controlador de ganancia fija.

6. MAPAS SENSORIOMOTORES.

Funciones que relacionen patrones sensoriales con consignas motoras son necesarias para llevar a cabo operaciones tales como el posicionamiento visual de robots manipuladores, la manipulación fina, y la navegación en el caso de robots móviles.

6.1 Posicionamiento visual de robots.

El paso clave hacia el control basado en visión es el mover una cámara de modo que la imagen captada se corresponda con una imagen de referencia dada. El objetivo a alcanzar no es ya pues una posición en el espacio, sino una sensación óptica.

Este posicionamiento por visión se aborde habitualmente prefijando un conjunto de características en la imagen y calculando a continuación la matriz de interacción que relaciona desplazamientos 2D de dichas características en la imagen con movimientos 3D de la cámara. Las potenciales ventajas derivadas de la aplicación de redes neuronales a esta tarea son: 1. la automatización de la programación del sistema, puesto que la matriz de interacción se obtiene por aprendizaje; y 2. el ahorro de tiempo de ejecución, al obviar la necesidad de calcular la correspondencia entre las características de la imagen captada y la de referencia.

La primera ventaja resulta ya patente en el sistema desarrollado por Hasimoto y col.(1992). El sistema de Hasimoto funciona sobre la base de que es posible detectar cuatro puntos característicos en la imagen. Se utilizan dos redes neuronales para mover la cámara de modo que se capten esos cuatro puntos en una configuración predeterminada. Una controla los movimientos de largo alcance, mientras que la otra guía los movimientos finos. La única diferencia es que la red utiliza la posición de la cámara como entrada, mientras que la segunda funciona independientemente de la posición absoluta.

El entrenamiento de la red se consigue moviendo la cámara desde la posición de referencia a posiciones arbitrarias, y aplicando entonces el algoritmo de retroprogramación del error para aprender la asociación entre los desplazamientos de los cuatro puntos en la imagen y el movimiento realizado.

Los resultados obtenidos mediante simulación muestran que, después de 30000 iteraciones del algoritmo de retroprogramación del error, la posición de los cuarto puntos después de situar la cámara difiere de la de referencia en, como máximo, un pixel. Desde el punto de vista de la precisión, los resultados son pues satisfactorios.

6.2. Manipulación fina.

La posibilidad de usar una red neuronal para aprender la acción que debe aplicarse frente a cada patrón de fuerza (es decir, el apropiado mapa sensoriomotor) resulta muy activa.

Gullapalli y col. (1992) han utilizado un sistema de *aprendizaje asociativo basado en una señal de refuerzo* para aprender la estrategia de control por acomodación activa necesaria para insertar un vástago en un aguero en un entorno 2D. Su sistema toma la posición de vástago junto con los valores de fuerza y pare como entrada, y la discrepancia entre la posición deseada y la posición real del vástago, con un término de penalización que se activa cada vez que las fuerzas medidas sobre el vástago superan un valor máximo predeterminado.

El sistema está formado por una red multicapa con 3 neuronas estocásticas en la capa de salida y neuronas sigmoidales deterministas en las restantes capas. Todas las neuronas toman valores reales.

Durante el entrenamiento, cada intento se inicia con el vástago situado en una posición y una orientación

aleatorias respecto al agujero y finaliza con el vástago satisfactoriamente insertados o bien después de transcurridas 100 unidades de tiempo. Los experimentos realizados con el robot real muestran que, después de 150 intentos, el robot siempre es capaz de completar la inserción. Además, la duración de la inserción decrece continuamente desde 100 a 20 unidades de tiempo a lo largo de 500 intentos.

6.3 Generación de Trayectorias para robots móviles.

Los robots móviles realizan a menudo tareas en entornos no previamente conocidos. Por ello, es de gran importancia que puedan llegar a un determinado objetivo sin colisionar con los obstáculos que encuentren en su camino. Igualmente es de desear que vayan aprendiendo de la experiencia de modo que, después de un cierto tiempo de permanecer en un mismo entorno, sigan trayectorias más eficientes que las generadas inicialmente.

Millán y Torras (1992) han desarrollado un controlador, basado en *aprendizaje por refuerzo*, que permite generar trayectorias progresivamente más cortas y seguras. La entrada del sistema está formada por una fuerza de atracción ejercida por la meta y cuatro fuerzas de repulsión ejercidas por los obstáculos. La salida del sistema codifica una acción del robot en términos de una orientación y una longitud de paso. Todas las señales involucradas están codificadas como números reales, lo cual supone la extensión de los algoritmos de aprendizaje por refuerzo (clásicamente propuestos para señales binarias) a dicho caso.

El sistema se compone de dos redes: un generador de pasos y un módulo crítico. Este último es una red de tres capas que funciona mediante un algoritmo de retropropagación, en tanto que el generador de pasos se construye incrementalmente a partir de una red de dos capas provistas de dos neuronas de salida estocásticas.

Después de 75.000 ciclos de aprendizaje, el generador de pasos ha aprendido un mapa sensoriomotor suficientemente preciso como para generar trayectorias satisfactorias desde prácticamente cualquier posición inicial del robot en el entorno. Obsérvese que, como la retroalimentación es mucho menos informada en el caso del aprendizaje por refuerzo que en el de minimización por error, el primero requiere habitualmente un periodo de entrenamiento considerablemente más largo.

Hay que destacar que el mapa sensoriomotor obtenido es altamente insensible al ruido en las entradas (la generación de trayectorias sigue siendo satisfactoria aún cuando se añada un 20% de ruido blanco), muestra una buena capacidad de generalización frente a cambios en la situación de la meta o en número y distribución de los obstáculos, e incluso funciona adecuadamente cuando la meta y los obstáculos son móviles. La principal limitación es el largo tiempo de entrenamiento.