Algoritmos Básicos de la Computación Cognitiva para la Programación de Cerebros Artificiales

Ing. Juan Carlos Kuri Pinto [1]
Ph.D. Sixto Ernesto García Aguilar [2]
Facultad de Ingeniería en Electricidad y Computación (FIEC) [1] [2]
Escuela Superior Politécnica del Litoral (ESPOL)
Campus Gustavo Galindo, Km 30.5 vía Perimetral
Apartado 09-01-5863. Guayaquil-Ecuador
jckuri@gmail.com [1]
sgarcia@espol.edu.ec [2]

Resumen

El reduccionismo científico crea soluciones casi perfectas, inflexibles y específicas para una gran cantidad de problemas. Por lo general, estas soluciones son modelos de deducción causal que van desde las causas hacia los efectos. En esta tesis se presenta 3 algoritmos inteligentes de inducción causal que van desde los efectos hacia las causas: El reconstructor de imágenes mentales, el caracterizador evolutivo y el ruteador causal jerárquico. Estos algoritmos usan redes de nodos interrelacionados que exploran el espacio de patrones para encontrar las mejores causas o soluciones que explican los efectos o problemas presentados como evidencia. Para cada algoritmo, se presenta una aplicación. Pero estos algoritmos son muy generales y aplicables a muchos problemas. El reconstructor de imágenes mentales resuelve el juego del buscaminas en pocos segundos. El caracterizador evolutivo infiere la tridimensionalidad de fotos bidimensionales mediante un sistema básico de visión artificial. Y el ruteador causal jerárquico se presenta como una solución teórica para ***el aprendizaje y auto-organización en tiempo real de las geometrías causales*** *de un brazo robótico y de todo tipo de robot. Pero este problema es una frontera de la ciencia. No se lo resuelve pero se sugieren estrategias para resolverlo.*

Palabras Claves: *Holismo versus reduccionismo, inducción causal, el problema inverso de la inteligencia, inferencia probabilística, optimización de parámetros, descenso de gradiente, procesos de decisión markovianos, redes neuronales, neurociencia, inteligencia artificial aplicada a los videojuegos, campo vectorial de la visión, mecánica de Lagrange aplicada a la robótica.*

Abstract

Scientific reductionism finds solutions that are almost perfect, inflexible, and specific for a big variety of problems. In general, these solutions are models of causal deduction that go from causes to effects. This thesis presents 3 intelligent algorithms of causal induction that go from effects to causes: The mental image reconstructor, the evolutionary characterizer, and the hierarchical causal router. These algorithms use networks of interrelated nodes that exploit the pattern space to find the best causes or solutions that explain the effects or problems presented as evidence. For each algorithm, an application is presented. But these algorithms are very general and applicable to many problems. The mental image reconstructor solves the minesweeper videogame in few seconds. The evolutionary characterizer infers the tridimensionality of bidimensional photos through a basic system of artificial vision. And the hierarchical causal router is presented as a theoretical solution for ***learning and self-organizing in real time the causal geometries*** *of a robotic arm and of every type of robot. But this problem is a frontier of science. It is not solved here but some strategies to solve it are suggested.*

Keywords*: Holism versus reductionism, causal induction, the inverse problem of intelligence, probabilistic inference, parameter optimization, gradient descent, Markov decision processes, neural networks, neuroscience, artificial intelligence applied to videogames, vectorial field of vision, Lagrangian mechanics applied to robotics.*

1. Introducción

El reduccionismo trata de abstraer los fenómenos naturales tomando en cuenta sólo los aspectos más relevantes y desechando los detalles que no son importantes para la resolución de un problema en particular. En inteligencia artificial, el reduccionismo ha influenciado el pensamiento de muchos investigadores en la creación de algoritmos que resuelven problemas de manera especializada y que en muchos casos no son capaces de aprender nuevas habilidades en otros dominios. El reduccionismo ha creado grandes teorías e invenciones, incluyendo los computadores que son muy rápidos, precisos y confiables.

Sin embargo, el cerebro no es un conglomerado de módulos separados, innatos, hardwired y reduccionistas como lo es el software actual. El cerebro está conformado de redes holistas [1], sinérgicas, auto-organizativas [2], autónomas, invariantes [3] y capaces de aprender empíricamente. Las soluciones reduccionistas como el software actual, los inventos, las teorías, la ciencia, la programación y

las matemáticas son el producto final de la inteligencia, pero no la inteligencia per se.

Esta tesis explota las redes de nodos abstractos con el fin de que el computador sea capaz de aprender a partir de la experiencia [4]; de resolver ambigüedades explorando el contexto en los niveles superiores de complejidad [1]; de reconocer patrones de manera invariante y flexible [3]; de auto-organizar el conocimiento aprendido y de generar autónomamente comportamientos y percepciones [2].

Estos algoritmos fueron inspirados indirectamente por la neurociencia. Se puede hallar correspondencias analógicas entre los sistemas nerviosos biológicos y las redes de nodos abstractos. Las neuronas biológicas corresponderían a nodos abstractos en el espacio de la mente. Esta tesis no pretende resolver completamente el dilema de cómo funciona la inteligencia. Es sólo un intento básico para superar las limitaciones del software actual. Estos algoritmos no son réplicas exactas de cómo el cerebro funciona; pero son mezclas sinérgicas de matemáticas y algoritmos de inteligencia artificial que son usados actualmente por la comunidad científica. Cada algoritmo será demostrado con una implementación y una aplicación práctica en el lenguaje de programación Haskell [5], lo cual garantiza su validez. Los algoritmos expuestos en la tesis fueron formulados con el propósito de crear un conjunto de herramientas abstractas y versátiles que sean capaces de resolver una cantidad casi ilimitada de problemas. Cada algoritmo tiene un gran rango de aplicabilidad.

2. Algoritmo: Reconstructor de Imágenes Mentales

El reconstructor de imágenes mentales sirve para hacer inducción causal. El nombre de esta red sugiere que a partir de la evidencia presentada en forma de sensaciones, se puede reconstruir causalmente las percepciones o imágenes mentales de lo que ocurre en la realidad. La inducción es uno de los aspectos más importantes de la inteligencia [6]. Por ejemplo, la percepción fluye en el sentido inverso de la causalidad: Va de sensaciones (efectos) a percepciones (causas).

La inducción causal puede hacerse por medio de un algoritmo de búsqueda. Puede ser un análisis combinatorial de fuerza bruta cuyo orden algorítmico es exponencial. Sin embargo, el reconstructor de imágenes mentales es una solución parcialmente exponencial donde se explota la zona de crecimiento suave de la exponencial previniendo las explosiones combinatoriales a través del análisis parcial de los constraints por separado. El algoritmo genera patrones de percepción coherentes basándose en la evidencia de los patrones de sensación y en los constraints. Esta teoría es muy versátil porque la inducción no sólo es usada en la percepción (córtex perceptivo) sino también en la planificación (córtex ejecutivo) gracias al isomorfismo de las columnas corticales [3].

La intuición detrás del algoritmo de reconstrucción de imágenes mentales es muy sencilla. Pero puede ser expresada de múltiples formas dependiendo del tipo de campo causal con el que se trabaja. Se podría definir un campo causal como las múltiples posibles causas que activan a un nodo mental abstracto. Es decir, si se activa un nodo mental, significa que esta activación se debe a una o varias de las múltiples causas conectadas a este. Y si no se activa un nodo mental, significa que las múltiples causas conectadas a este no están presentes. Estas asociaciones causales podrían ser aprendidas o pre-programadas. Y así mismo el tipo de campo causal podría ser exacto o flexible. Continuo o discreto. Jerárquico o plano. Perceptivo o ejecutivo. Excitatorio o inhibitorio. Simple o complejo. Y pueden haber diferentes tipos de clasificaciones y formas de campos causales que todavía están por descubrirse.

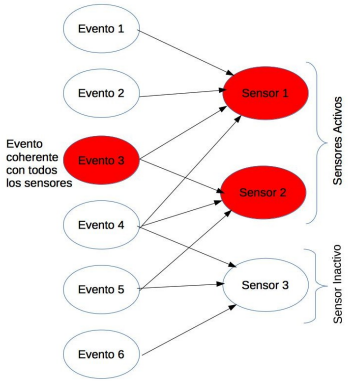

Figura 1. Ejemplo de campos causales traslapados

En el gráfico, a la izquierda están los eventos que son las posibles causas de un fenómeno. A la derecha están los sensores que capturan los efectos producidos por la presencia de los eventos. Los eventos existen en 2 formas: Los eventos de la realidad y la representación mental de los eventos en el cerebro. Las percepciones son las causas que son coherentes con todas las sensaciones.

Intuitivamente, se puede decir que si un nodo mental se activa, una de sus posibles causas asociadas está presente. Pero no se sabe cuál de ellas está presente. Para saberlo y para resolver la ambigüedad, se examina el contexto. Es decir, se examina la activación de sus campos causales vecinos que tienen traslapamiento de causas o, en otras palabras, posibles causas comunes que los vincula. El traslapamiento causal elimina las posibles causas que no son coherentes con ambos campos causales y así sucesivamente a través de las cadenas de traslapamiento causal. Y mientras más traslapamiento causal hay y más evidencia de activaciones es presentada, menos causas son coherentes porque es poco probable que una causa satisfaga muchos campos causales asociados a esta. Es decir, las imágenes mentales se vuelven muy claras y pierden ambigüedad. Además, las percepciones se vuelven indebatibles cuando todos los campos causales están

completamente apoyados por evidencia (hechos) o carecen totalmente de evidencia (incoherencias).

3. Aplicación: Resolución del Juego de Buscaminas en Pocos Segundos

El reconstructor de imágenes mentales permite resolver el juego del buscaminas en tiempos que ningún ser humano puede superar:

Figura 2. Mejores tiempos del reconstructor de imágenes mentales

Para ilustrar cómo funciona el algoritmo, se presentará un ejemplo:

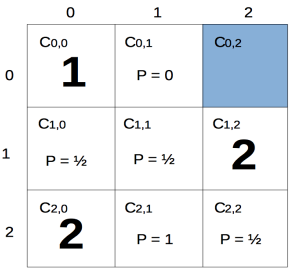

Figura 3. Ejemplo de un campo de minas

En este ejemplo, hay 5 variables o celdas que tienen una probabilidad *P* de tener una mina: $C_{0,1}$, $C_{1,0}$, $C_{1,1}$, $C_{2,1}$ y $C_{2,2}$. Si tienen mina, tendrán un valor de *1*. Y si no tienen mina, un valor de *0*. Y hay 3 campos traslapados de causalidad (3 sensores de minas) que se describen con las siguientes ecuaciones algebraicas:

$$C_{0,1}+C_{1,0}+C_{1,1}=1 \quad (1)$$

$$C_{1,0}+C_{1,1}+C_{2,1}=2 \quad (2)$$

$$C_{0,1}+C_{1,1}+C_{2,1}+C_{2,2}=2 \quad (3)$$

Cada variable *Ci,j* sólo puede tomar 2 valores: *0* ó *1*. Entonces para encontrar los valores que sean coherentes con todos los campos causales se puede hacer una búsqueda exhaustiva de todas las posibles combinaciones de patrones cuyo orden algorítimico es exponencial.

Pero también se puede buscar de una forma más inteligente analizando cada campo causal por separado y luego se buscan las intersecciones entre campos causales, desechando los patrones incoherentes para cada campo causal. De esta forma no se analiza todo el espacio de patrones, sólo los patrones que posiblemente sean coherentes.

El espacio de posibles patrones es muy grande y sufre de la maldición de la dimensionalidad de Bellman [7]. Pero los patrones coherentes son muy pocos en comparación con todo el espacio de patrones. En el ejemplo mostrado anteriormente, sólo 2 patrones son coherentes con todos los campos causales o constraints. Mientras que todo el espacio de patrones tiene 32 patrones (2^5). Los 2 patrones coherentes son:

$$C_{0,1}=0, C_{1,0}=0, C_{1,1}=1, C_{2,1}=1, C_{2,2}=0 \quad (4)$$

$$C_{0,1}=0, C_{1,0}=1, C_{1,1}=0, C_{2,1}=1, C_{2,2}=1 \quad (5)$$

La coherencia es una heurística muy poderosa que ahorra recursos computacionales y es aplicable no sólo para la percepción sino que también es aplicable para la generación de patrones sinérgicos.

4. Algoritmo: Caracterizador Evolutivo

El caracterizador evolutivo es un algoritmo muy versátil y configurable que sirve para hacer optimización de soluciones, inducción causal, resolución de problemas inversos y reconocimiento de patrones. Este algoritmo riega una lluvia de patrones a evolucionar en un espacio multidimensional. Los patrones a evolucionar son como gotas que se acumulan en los mínimos locales y global de la curva multidimensional de error. Las acumulaciones de gotas pueden ser representadas por un solo patrón en vez de varios para ahorrar recursos computacionales. La técnica de descenso de gradiente es usada como una metáfora de la selección natural puesto que la dirección del gradiente es el camino más directo para llegar a los óptimos locales y globales.

Se tiene un campo vectorial *F* con inputs *i*, parámetros a ajustar *p* y outputs *o*:

$$F(i_1, i_2, \ldots, i_{n_i}, p_1, p_2, \ldots, p_{n_p}) = (o_1, o_2, \ldots, o_{n_o}) \quad (6)$$

De todos los *n* ejemplos, se calcula la suma de los errores cuadráticos en el campo vectorial *F*:

$$Error = \sum_{i=1}^{n} \|O_i - F(I_i, P)\|^2 \quad (7)$$

Luego se calcula el gradiente de la curva del error con respecto a los parámetros *P*:

$$\nabla Error = \frac{\partial Error}{\partial P} = \left(\frac{\partial Error}{\partial p_1}, \frac{\partial Error}{\partial p_2}, \ldots, \frac{\partial Error}{\partial p_{n_p}}\right) \quad (8)$$

Para luego hacer evolucionar los parámetros *P* mediante la técnica del descenso del gradiente:

$$P_{n+1} = P_n - factor \cdot \frac{\nabla Error}{\|\nabla Error\|} \quad (9)$$

El factor de evolución puede ser común a todos los parámetros. Pero es mejor tener un *factor* de evolución por cada parámetro a evolucionar:

$$(p_1, p_2, \ldots, p_{n_p})_{n+1} = (p_1, p_2, \ldots, p_{n_p})_n - \frac{1}{\|\nabla Error\|}\left(factor_1 \cdot \frac{\partial Error}{\partial p_1}, factor_2 \cdot \frac{\partial Error}{\partial p_2}, \ldots, factor_{n_p} \cdot \frac{\partial Error}{\partial p_{n_p}}\right) \quad (10)$$

La regresión lineal es una proyección del plano parametrizable con respecto a los hiper-puntos a ajustar con la curva del error [8]. Entonces, si el caracterizador evolutivo es una generalización de la regresión lineal, el caracterizador evolutivo es una proyección de la superficie parametrizable con respecto a los patrones a ajustar con la curva del error. En otras palabras, es una proyección de patrones que es muy flexible (no exacta) y que sería la generalización de las fórmulas matemáticas exactas. Su flexibilidad es perfecta para el campo de la inteligencia artificial y el reconocimiento de patrones. Y de hecho las redes neuronales artificiales demuestran todo su potencial con respecto a la flexibilidad como Douglas Hofstadter lo enfatiza claramente en sus publicaciones: “The entire effort of artificial intelligence is essentially a fight against computers’ rigidity.”

5. Aplicación: Sistema Básico de Visión Artificial

En este sistema básico de visión artificial, el usuario debe tomar una captura de pantalla de un programa que permita ver lo que una webcam filma. El objetivo es capturar una foto de una hoja con puntos de referencia circulares y con alguna imagen en el centro de la hoja. Para luego hacer una reconstrucción tridimensional de la foto bidimensional. Es decir, inferir la posición, ángulos y factores de conversión de la webcam. Y finalmente corregir la perspectiva de la imagen en el centro de la hoja. El sistema calcula distancias con precisión milimétrica. Su error es de aproximadamente 5 mm.

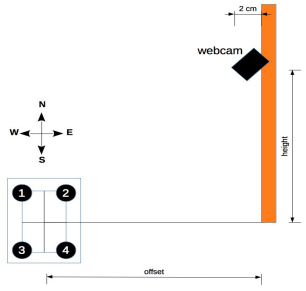

Figura 4. Mediciones del Sistema Básico de Visión Artificial

Para programar este sistema básico de visión artificial, es necesario hacer caracterizaciones parametrizables de los puntos de referencia y del campo vectorial de la visión.

5.1. Caracterización de los Puntos de Referencia

Cuando se proyecta un círculo en el campo vectorial de la visión, este círculo puede aproximarse a una elipse escalada, rotada y trasladada en el plano 2D. Entonces esta es la forma con la cual se caracterizan los puntos de referencia.

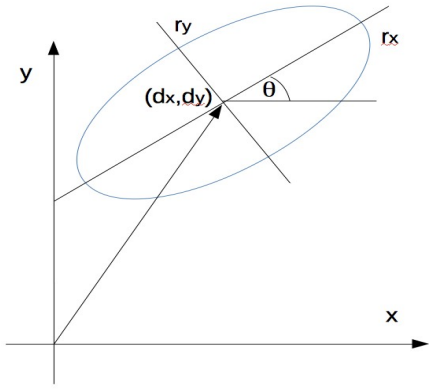

Figura 5. Elipse escalada, rotada y trasladada en el plano 2D

En el gráfico, *rx* y *ry* son los radios de la elipse. *(dx,dy)* es el centro de la elipse. Y θ es el ángulo de rotación de la elipse. La función radio inverso devuelve el radio de la elipse escalada, rotada y trasladada en el que se encuentra ubicado un punto *(x,y)*:

$$RadioInverso(x,y) = \left\| \begin{bmatrix} \frac{1}{rx} & 0 \\ 0 & \frac{1}{ry} \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{pmatrix} x-dx \\ y-dy \end{pmatrix} \right\| \quad (11)$$

La función *color* dice cuál debería ser el color de un pixel *(x,y)* para así caracterizar una imagen que contiene una elipse negra en un fondo blanco. *blanco* y *negro* son los tonos de color. Y *sigmoide* es una función logística.

$$color(x,y) = negro + (blanco - negro) \cdot sigmoide(15 \cdot (RadioInverso(x,y) - 1)) \quad (12)$$

5.2. Caracterización del Campo Vectorial de la Visión

Para caracterizarlo, se debe rotar una base ortonormal [9] en el espacio 3D. Luego se proyecta los vectores relativos al origen de la cámara con respecto a la base ortonormal. Estas proyecciones tridimensionales y lineales se deben proyectar hacia el

plano 2D de la pinhole camera por medio de la relación de triángulos.

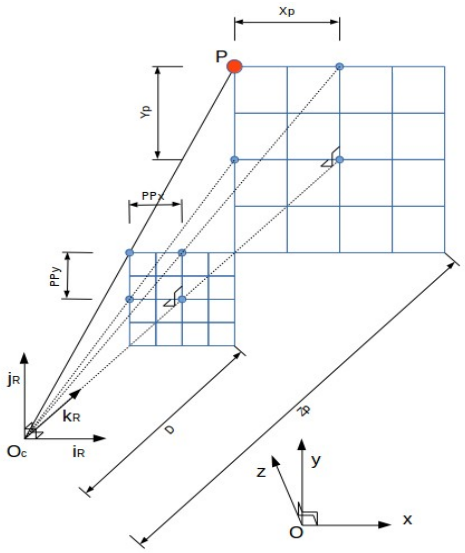

Figura 6. Campo vectorial de la visión

Oc es el centro de la cámara. i_R, j_R y k_R son la base ortonormal de la cámara. *Xp*, *Yp* y *Zp* son un punto proyectado con respecto a la base ortonormal. *D* es la distancia al plano de proyección. *PPx* y *PPy* son las proyecciones 2D en el plano de la cámara. A continuación, se presenta la base ortonormal en el espacio 3D. *A*, *B* y *C* son los ángulos de rotación.

$$i_R=\begin{pmatrix}\sin A\cdot\sin B\cdot\sin C+\cos A\cdot\cos B\\ \cos B\cdot\sin C\\ \cos A\cdot\sin B\cdot\sin C-\sin A\cdot\cos B\end{pmatrix} \quad (13)$$

$$j_R=\begin{pmatrix}\sin A\cdot\sin B\cdot\cos C-\cos A\cdot\sin C\\ \cos B\cdot\cos C\\ \cos A\cdot\sin B\cdot\cos C+\sin A\cdot\sin C\end{pmatrix} \quad (14)$$

$$k_R=\begin{pmatrix}\sin A\cdot\cos B\\ -\sin B\\ \cos A\cdot\cos B\end{pmatrix} \quad (15)$$

Con la siguiente ecuación se calculan los vectores relativos *v* con respecto al centro de la cámara o_c. *p* es el punto a proyectar.

$$v=p-o_c \quad (16)$$

Las siguientes ecuaciones son la base ortonormal rotada en el espacio 3D y sus proyecciones tridimensionales y lineales [9]:

$$H=\{i_R,j_R,k_R\} \quad (17)$$

$$proy_H v=(Xp,Yp,Zp)=(v\cdot i_R)i_R+(v\cdot j_R)j_R+(v\cdot k_R)k_R \quad (18)$$

A continuación, se presenta la proyección de $proy_H v$ hacia el plano 2D de la pinhole camera por medio de la relación de triángulos. *factor2D* sirve para agrandar las proyecciones 2D dentro de la pinhole camera que son pequeñas y presentarlas en la pantalla que es grande.

$$PP=\left(\frac{Xp\cdot D\cdot \text{factor2D}}{Zp},\frac{Yp\cdot D\cdot \text{factor2D}}{Zp}\right) \quad (19)$$

6. Algoritmo: Ruteador Causal Jerárquico

Este algoritmo es básicamente una versión PAC (Probably Approximately Correct) de los MDP (procesos de decisión markovianos) [4] que aprende y auto-organiza geometrías causales [10] y desarrolla reflejos condicionados similares a los generados por el subconsciente. En el gráfico, *S* son los estados, *a* son las acciones, *p* son las probabilidades y *C* son los costos.

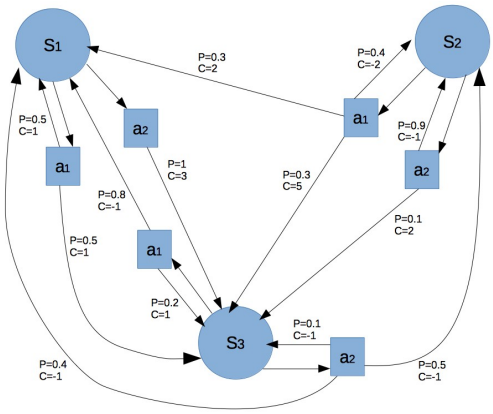

Figura 7. Ejemplo simple de un proceso de decisión markoviano

La versión PAC de los MDP de esta tesis sólo toma en cuenta los caminos causales más probables y menos costosos de tal modo que el cálculo de los costos es lineal y no arborícora. Siendo lineal, es más fácil calcular sus costos y verificar que los caminos causales todavía sigan vigentes. En la fórmula, *S* es el estado actual, *G* es el objetivo, *S'* es el estado siguiente, *a* es la acción, *P* es la probabilidad y *T* es el costo de transición.

$$Cost(S,G)=\min_a\left[\max_{S'}P(S,S',a)\cdot[T(S,S',a)+Cost(S',G)]\right] \quad (20)$$

Si se representa gráficamente la conectividad en memoria generada por el algoritmo de Bellman-Ford, la red generada se parece un poco a la capa 1 de las columnas corticales [11]. Puede ser una coincidencia, pero una forma de reconocer, generar y representar patrones de forma invariante [3] es a través de los algoritmos de búsqueda. La programación dinámica [7] es un algoritmo de búsqueda aplicable a reinforcement learning (ejecución) y a minimum-edit distance [12] (percepción). Ambos funcionan en redes de estados conectadas por acciones y

transformaciones. Es probable que el cerebro funcione de una forma similar. Aunque es un tanto especulativo afirmarlo.

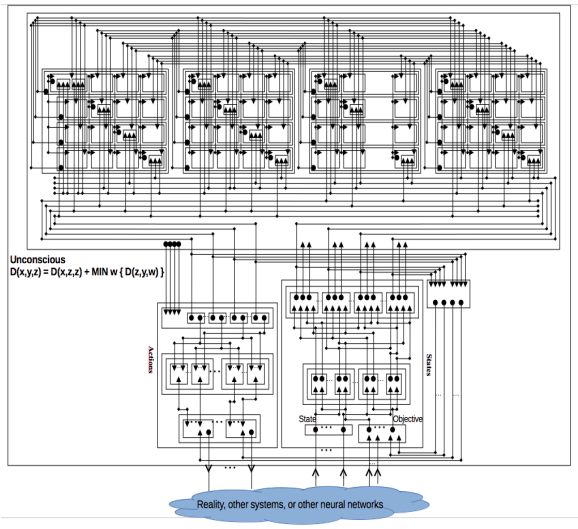

Figura 8. Representación gráfica de la conectividad en memoria generada por el algoritmo del cerebro artificial

Este algoritmo corrige el problema de la falta de propagación de malas noticias del algoritmo de Bellman-Ford [7]. En la demostración, se pueden abrir y cerrar los caminos causales y el algoritmo propagará las buenas y las malas noticias sin problemas y en tiempo real:

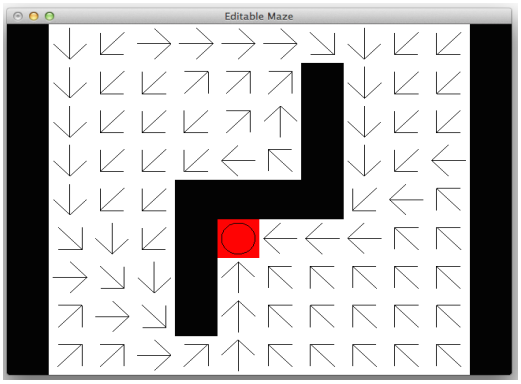

Figura 9. Programa ejemplo que encuentra los reflejos de una geometría causal

Una buena noticia puede ser que un animal desarrolle un reflejo condicionado o un patrón de sinergias musculares. Una mala noticia puede ser que un animal sufra de una fractura o amputación de una extremidad y deba rutear sus movimientos a través de otros mecanismos.

7. Problema no Resuelto: Brazo Robótico que Escribe en una Pizarra Virtual

Un modelo matemático bastante bueno de un brazo robótico puede ser un doble péndulo invertido controlado por 2 motores que generan torques [13]. En la figura, τ son los torques, *m* son las masas, *o* son los ejes de giro, *c* son los centros de masa, *l* son las longitudes, *lg* son las longitudes hasta el centro de masa, θ son los ángulos de rotación y $\dot{\theta}$ son las velocidades angulares.

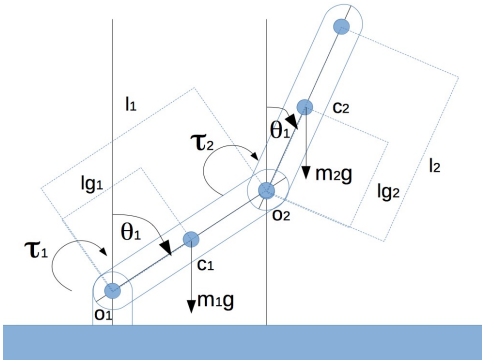

Figura 10. Modelo de un brazo robótico virtual

A continuación, se presenta el Lagrangiano del brazo robótico [14]:

$$L=\frac{1}{2}\cdot\dot{\theta_1}^2\cdot[I_1+m_1\cdot l_{g1}^2]+\frac{1}{2}\cdot I_2\cdot\dot{\theta_2}^2-m_1\cdot g\cdot(O_{1y}+l_{g1}\cdot\cos\theta_1)-m_2\cdot g\cdot(O_{1y}+l_1\cdot\cos\theta_1+l_{g2}\cdot\cos\theta_2)$$
$$+\frac{1}{2}\cdot m_2\cdot[l_1^2\cdot\dot{\theta_1}^2+2\cdot l_1\cdot l_{g2}\cdot\dot{\theta_1}\cdot\dot{\theta_2}\cdot\cos(\theta_1-\theta_2)+l_{g2}^2\cdot\dot{\theta_2}^2] \quad (21)$$

Se aplica la mecánica de Lagrange y se crea un videojuego cuyo objetivo es dibujar letras en una pizarra virtual.

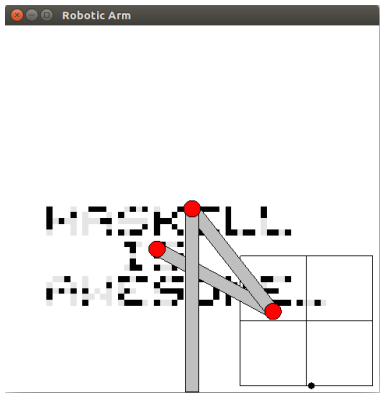

Figura 11. El videojuego del brazo robótico que escribe en una pizarra virtual

El brazo robótico es muy incontrolable y caótico porque es un doble péndulo invertido controlado por 2 torques. Ningún ser humano puede controlarlo. Nadie puede encontrar y recordar los patrones motrices que generen equilibrios estáticos y dinámicos (transiciones) para cada uno de los pixeles de la pizarra virtual. Los algoritmos tradicionales de reinforcement learning tampoco pueden controlar el brazo robótico por las explosiones combinatoriales generadas por la maldición de la dimensionalidad de Bellman.

Pero hay una esperanza: DeepMind Technologies [15] creó algoritmos de Deep Q-Learning. Es

simplemente un algoritmo de Q-Learning cuyos valores son estimados por una red neuronal que abstrae análogamente estados similares en el espacio de patrones. Con este algoritmo, las máquinas aprenden a jugar videojuegos a partir de pixels, acciones y scores, con un nivel que supera a los humanos. Y recientemente un algoritmo similar venció 4 veces al campeón mundial de Go (ajedrez japonés). A grandes rasgos, los videojuegos de Atari, el juego Go y el problema general de la robótica pertenecen a la misma categoría de problemas: Espacios enormes de estados conectados por acciones, probabilidades y costos. En futuras publicaciones, se resolverá este problema que casi se resolvió en esta tesis. Pero sería interesante resolverlo para que la red aprenda en tiempo real con proyecciones simbólicas que no sean métodos iterativos [16] de tal modo que no sea necesario usar demasiados procesadores para resolver el problema. O también buscando rápida y eficazmente la topología de red que se ajuste mejor a la geometría causal del problema por medio del principio de minimum description length (MDL) y el análisis dimensional en el dominio de las frecuencias [17].

8. Resultados

El reconstructor de imágenes mentales es un modelo simple pero captura bien lo que hacen las neuronas (nervios aferentes y confluencia de inervaciones causales [11]) y lo que hacen las redes de neuronas: Inducción causal desambiguando campos causales traslapados. La inducción causal está presente en todos los aspectos de la inteligencia. Esta red tiene una gran futuro y muchas aplicaciones.

El caracterizador evolutivo es una gran herramienta para el aprendizaje supervisado. Además flexibiliza las matemáticas tradicionales y su dinámica de inferencia probabilística funciona con atractores caóticos lo cual permite hacer reconocimiento invariante de patrones. Es tolerante a ruidos en las señales y permite identificar causas superpuestas por medio de sumatorias de patrones en la caracterización matemática.

El ruteador causal es un intento por resolver el gran y ambicioso dilema de cómo funciona el aprendizaje no-supervisado. Esta red no sólo sirve para hacer reinforcement learning sino que también mapea de manera no-supervisada espacios de patrones complejos. Hace inducción causal al explorar, inferir y aprender geometrías causales complejas.

9. Conclusiones

A pesar de la naturaleza multidisciplinaria de los algoritmos y aplicaciones presentadas en esta tesis, es claro que el factor común es la búsqueda de patrones causales en redes cognitivas artificiales. La inducción causal es el tema central de la tesis y uno de los aspectos más importantes de la inteligencia. Si se analiza los aspectos comunes de los algoritmos inteligentes y se expresan como principios de inteligencia, será posible crear un algoritmo general de aprendizaje y razonamiento que podrá ser aplicable a cualquier dominio del conocimiento. La existencia de los cerebros biológicos es tanto la prueba feaciente de que es posible crear al algoritmo maestro [15] como una inspiración para aquellos que se dedican a programar inteligencia artificial.

10. Agradecimientos

Al Universo, a la Naturaleza y a mis padres, el Dr. José Kuri y la Dra. Raquel Pinto, por darme la vida. A Sixto García, a Jorge Flores Herrera, a Monica Anderson y a Eray Özkural por su ayuda. Al equipo de Haskell por hacer posible esta tesis. A la ESPOL, a Coursera y a Udacity por educarme.

11. Referencias

[1] Artificial Intuition. 2007. Available at http://artificial-intuition.com/
[2] Kelso, S., *Dynamic Patterns: The Self-Organization of Brain and Behavior*, A Bradford Book, 1995.
[3] Hawkins, J., and Blakeslee, S., *On Intelligence*, Times Books, 2004.
[4] Sutton, R., and Barto, A., *Reinforcement Learning: An Introduction*, A Bradford Book, 1998.
[5] The Haskell Programming Language. 2013. Available at http://www.haskell.org/haskellwiki/Haskell
[6] Legg, S., *Machine Super Intelligence*, Lulu, 2008.
[7] Bellman, R., *Dynamic Programming*, Dover Publications, 2003.
[8] Klein, P., *Coding the Matrix: Linear Algebra through Computer Science Applications*, Newtonian Press, 2013.
[9] Grossman, S., *Álgebra Lineal*, McGraw-Hill, 1996.
[10] Bizarre Domains. 2007. Available at http://artificial-intuition.com/bizarre.html
[11] Fuster, J., *Cortex and Mind: Unifying Cognition*, Oxford University Press, 2003.
[12] Structural and Syntactic Pattern Recognition. 2015. Available at http://www.cs.bilkent.edu.tr/~saksoy/courses/cs551/slides/cs551_structural.pdf
[13] Resnick, R., Halliday, D., y Krane, K., *Física*, CECSA, 1996.
[14] Lagrangian mechanics. 2014. Available at http://en.wikipedia.org/wiki/Lagrangian_mechanics
[15] Google DeepMind. 2016. Available at https://deepmind.com/
[16] Saxe, A., McClelland, J., Ganguli, and S., “Exact solutions to the nonlinear dynamics of learning in deep linear neural networks”, Cornell University Library, 2014.
[17] Compressed Network Search Finds Complex Neural Controllers with a Million Weights. 2013. Available at http://people.idsia.ch/~juergen/compressednetworksearch.html