ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería Eléctrica y Computación

“ALGORITMOS BÁSICOS DE LA COMPUTACIÓN COGNITIVA PARA LA PROGRAMACIÓN DE CEREBROS ARTIFICIALES”

EXAMEN COMPLEXIVO

Previa la obtención del Título de:

INGENIERO EN CIENCIAS COMPUTACIONALES

Presentada por:

JUAN CARLOS KURI PINTO

GUAYAQUIL — ECUADOR

2015

AGRADECIMIENTO

Al Universo, a la Naturaleza y a mis padres 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.

DEDICATORIA

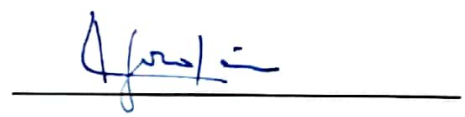
A mi madre la Dra. Raquel Pinto Gálvez por todo lo que me ha dado y apoyado. Se lo debo todo a ella.

A mi padre el Dr. José Kuri González por haberme involucrado en las neurociencias.

TRIBUNAL DE SUSTENTACIÓN

Ph.D. Sixto García

EVALUADOR



M.Sc. Carlos Jordán

EVALUADOR

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este Examen Complexivo, me corresponden exclusivamente; y el patrimonio intelectual del mismo a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL."

(Reglamento de Graduación de la ESPOL.)

Juan Carlos Kuri Pinto

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 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.

ÍNDICE GENERAL

ABREVIATURAS

3D	Tridimensional.
1.x	Dimensión fractálica entre 1 y 2.
2.x	Dimensión fractálica entre 2 y 3.
2D	Bidimensional.
AGI	Artificial General Intelligence.
AI	Artificial Intelligence.
alt	Tecla alt para acceder a los caracteres alternativos.
App	Application o aplicación.
cm	Centímetros.
cognits	Columnas corticales.
DNA	Deoxyribonucleic acid. Genes.
E	East.
FIFO	First In, First Out.
FP Complete	Functional Programming Complete. Un sitio web que ayuda a programar en Haskell.
GPU	Graphics Processing Unit.
GTK	GNOME Toolkit. Librería gráfica de Linux.
HHMM	Hierarchical Hidden Markov Models.

HMDP	Hierarchical Markov Decision Process.
L1	Layer 1. Capa 1 de la corteza cortical.
MDP	Markov Decision Process.
mm	Milímetros.
MOOC	Massive Online Open Course.
N	North.
NN	Neural Network.
OS	Operative System.
PAC learning	Probably Approximately Correct learning.
PAC MDP	Probably Approximately Correct Markov Decision Process.
PixBuf	Pixel Buffer. Una librería para leer y escribir imágenes.
POSMDP	Partially Observable Semi Markov Decision Process.
RL	Reinforcement Learning.
RNN	Recurrent Neural Network.
S	South.
STM	Software Transactional Memory.
TCGA	Thymine, Cytosine, Adenine, Guanine. Las 4 moléculas del DNA.
W	West.
webcam	Cámara web.

X11	X Window System. Sistema de ventanas para Linux.
Xcode	Entorno de desarrollo para Mac.
Xtest	Librería de Unix para simulación de eventos de mouse.

SIMBOLOGÍA

O(.)	Orden algorítmico.
Pi,j	Probabilidad de la celda (i,j).
Ci,j	Celda (i,j) del buscaminas.
i	Índice de las filas.
j	Índice de las columnas.
1C	La suma de los campos causales es 1 y es coherente.
2C	La suma de los campos causales es 2 y es coherente.
3C	La suma de los campos causales es 3 y es coherente.
R^n	Espacio vectorial de n números reales.
P(Xi)	Probabilidad de que ocurra el evento Xi.
H(x)	Entropía de x.
O(NP)	El orden del algoritmo es no polinómico.
O(P)	El orden del algoritmo es polinómico.
P(ganar)	Probabilidad de ganar el juego del buscaminas.
Pi	Riesgo tomado en el intento i.
N	Número de veces.
M	Número de veces en otra dimensión.

F	Función multidimensional para utilizar con el caracterizador evolutivo.
i	Vector multidimensional para representar el input.
o	Vector multidimensional para representar el output.
p	Vector multidimensional para representar los parámetros.
n_i	Número de dimensiones para el input.
n_p	Número de dimensiones para los parámetros.
n_o	Número de dimensiones para el output.
$pair_i$	Par de vectores multidimensionales de (input,output).
∇	Gradiente.
∂	Derivada parcial.
ΔP	Variación infinitesimal para calcular las derivadas parciales.
P_n	Parámetros en la iteración n.
P_{n+1}	Parámetros en la iteración n+1.
r_x	Radio x de la elipse rotada.
r_y	Radio y de la elipse rotada.
θ	Ángulo de rotación de la elipse.
dx	Desplazamiento x de la elipse rotada.
dy	Desplazamiento y de la elipse rotada.
i	Eje i de la base ortonormal.

j	Eje j de la base ortonormal.
k	Eje k de la base ortonormal.
A	Rotación de la base ortonormal sobre el eje j.
B	Rotación de la base ortonormal sobre el eje i.
C	Rotación de la base ortonormal sobre el eje k.
p'	Punto rotado sobre la recta RR(t).
θ	Ángulo de rotación sobre la recta RR(t).
n	Punto de la recta de rotación RR(t).
r	Dirección de la recta de rotación RR(t).
p	Punto a rotar sobre la recta RR(t).
RR(t)	Recta de rotación con parámetro t.
i_R	Eje i de la base ortonormal rotada.
j_R	Eje j de la base ortonormal rotada.
k_R	Eje k de la base ortonormal rotada.
V	Punto de un objeto en coordenadas relativas a la cámara.
P	Punto de un objeto en coordenadas absolutas.
O_c	Punto de origen de la cámara.
H	Base ortonormal rotada.
$proy_H V$	Proyección de vector relativo V con respecto a la base H.
PP	Punto proyectado.

x_p	Coordenada x del punto P.
y_p	Coordenada y del punto P.
z_p	Coordenada z del punto P.
D	Distancia al plano de proyección de la pinhole camera.
factor2D	Factor de conversión de 3D a 2D.
F	F = D * factor2D
PP_x	Coordenada x del punto proyectado PP.
PP_y	Coordenada y del punto proyectado PP.
$P_{0,0}$	Punto (0,0) del cuadrilátero generalizado.
$P_{0,1}$	Punto (0,1) del cuadrilátero generalizado.
$P_{1,0}$	Punto (1,0) del cuadrilátero generalizado.
$P_{1,1}$	Punto (1,1) del cuadrilátero generalizado.
t_i	Parámetro de la recta L_i.
t_j	Parámetro de la recta L_j.
L_i	Recta i del cuadrilátero generalizado.
L_j	Recta j del cuadrilátero generalizado.
$V_{i,0}$	Vector (i,0) del cuadrilátero generalizado.
$V_{i,1}$	Vector (i,1) del cuadrilátero generalizado.
$V_{j,0}$	Vector (j,0) del cuadrilátero generalizado.
$V_{j,1}$	Vector (j,1) del cuadrilátero generalizado.

P4	Variable auxiliar de la transformación inversa del cuadrilátero.
VX	Variable auxiliar de la transformación inversa del cuadrilátero.
VY	Variable auxiliar de la transformación inversa del cuadrilátero.
LO	Variable auxiliar de la transformación inversa del cuadrilátero.
S	Estado del MDP.
a	Acción del MDP.
P	Probabilidad de una transición en el MDP.
C	Costo de una transición en el MDP.
P(x1,x2)	Función de pertenencia piramidal de la lógica difusa.
T(x)	Función de pertenencia triangular de la lógica difusa.
N	Número de eventos para el muestreo de una transición.
Cost(S,G,a)	Costo de rutear desde el estado S hacia el objetivo G por medio de la acción a.
P(S,S',a)	Probabilidad de ir desde el estado S hacia el estado S' por medio de la acción a.
T(S,S',a)	Costo de ir desde el estado S hacia el estado S' por medio de la acción a.
γ	Factor de desvanecimiento de los estados lejanos del MDP.
$\Pi(S,G)$	Política o acción refleja para ir desde S hasta G.
Cost(S,G)	Costo de ir desde el estado S hasta el objetivo G.

$W_{i,j}$	Peso de la sinapsis (i,j)
X_i	Vector de entrada de la red neuronal.
Y_j	Vector intermedio de la red neuronal.
Z_k	Vector de salida de la red neuronal.
τ	Torque.
l	Longitud de una sección del brazo robótico.
lg	Longitud hasta el centro de gravedad de una sección.
θ	Ángulo de una sección del brazo robótico.
ω	Velocidad angular de una sección del brazo robótico.
O	Inicio de una sección del brazo robótico.
C	Centro de una sección del brazo robótico.
m	Masa de una sección del brazo robótico.
$\dot{C}$	Derivada del centro C con respecto al tiempo.
$\dot{\theta}$	Derivada del ángulo con respecto al tiempo.
$\dot{x}$	Derivada de x con respecto al tiempo.
$\dot{y}$	Derivada de y con respecto al tiempo.
h	Height o altura de una sección.
w	Width o ancho de una sección.
I_e	Momento de inercia de una placa rectangular.
I_p	Momento de inercia generalizado.

$\rho(r)$	Densidad en función del radio de giro.
dV	Diferencial volumétrico que sirve para integrar la inercia.
L	Lagrangiano.
g	Gravedad
V_i	Velocidad de la sección i.
Q_j	Sumatoria de fuerzas en la mecánica de Lagrange.
F_i	Fuerza aplicada al sistema.
kf_i	Coeficiente de fricción aplicado al sistema.
$\ddot{\theta}$	Aceleración angular de una sección del brazo robótico.

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Esta tesis tiene 3 partes. Se presenta 3 algoritmos inteligentes de inducción causal con sus respectivas aplicaciones:

La parte 1 es el reconstructor de imágenes mentales (capítulo 2) que resuelve el videojuego del buscaminas en pocos segundos (capítulo 3).

La parte 2 es el caracterizador evolutivo (capítulo 4) que infiere la tridimensionalidad de fotos bidimensionales mediante un sistema básico de visión artificial (capítulo 5).

La parte 3 es el ruteador causal jerárquico (capítulo 6) que se presenta como una solución teórica para el aprendizaje y auto-organización en tiempo real de las geometrías causales de un brazo robótico (capítulo 7).

CAPÍTULO 1

1. ANTECEDENTES Y JUSTIFICACIÓN

1.1. Descripción del Problema

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 [1]. 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 [2]. 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 [3], sinérgicas [4], auto-organizativas [5], autónomas [2], invariantes [6] y capaces de aprender empíricamente [7]. 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 [3].

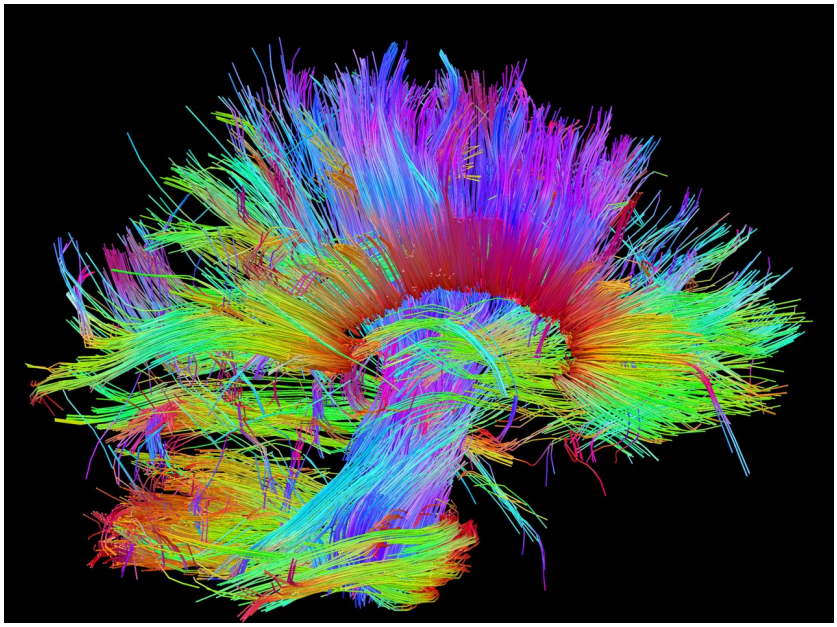


Figura 1.1: El conectoma [8]

Esta tesis explota las redes de nodos abstractos con el fin de que el computador sea capaz de aprender a partir de la experiencia [7]; de resolver ambigüedades explorando el contexto en los niveles superiores de complejidad [3]; de reconocer patrones de manera invariante y flexible [6,9]; de auto-organizar el conocimiento aprendido [5] y de generar autónomamente comportamientos y percepciones [10].

La computación cognitiva es una rama de las ciencias computacionales que estudia los sistemas cognitivos que tratan de

imitar los aspectos más relevantes de la cognición (por ejemplo: sensar, percibir, inferir, predecir, buscar, planificar, crear, aprender, pensar, etc.) con el fin de flexibilizar las habilidades de los computadores, extender su dominio de aplicabilidad en el mundo real e interactuar más naturalmente con los seres humanos [11].

En esta tesis se presenta 3 algoritmos básicos de la computación cognitiva: El reconstructor de imágenes mentales, el caracterizador evolutivo y el ruteador causal jerárquico. Estos algoritmos fueron implementados y programados por el autor de la tesis basándose en varias ideas, conceptos, principios, teorías y otros algoritmos de la neurociencia, la inteligencia artificial, la programación y las matemáticas.

1.2. Justificación

La gran mayoría del software actual no aprende nuevas habilidades a partir de la experiencia, ni resuelve las ambigüedades del mundo real, ni reconoce patrones de manera invariante y flexible, ni auto-organiza

el conocimiento aprendido, ni genera autónomamente comportamientos y percepciones.

El presente trabajo, sin pretender resolver completamente el dilema de cómo funciona la inteligencia, es un intento básico para superar las limitaciones del software actual con el objeto de que los computadores sean cada vez más capaces de ayudarnos a progresar como civilización y de resolver los problemas de nuestra sociedad.

1.3. Solución Propuesta

Para superar las limitaciones del software actual, se propone 3 algoritmos inteligentes de inducción causal que van desde los efectos hacia las causas [12]. Estos algoritmos tienen un gran rango de aplicabilidad porque usan redes de nodos interrelacionados que exploran el espacio de patrones para encontrar las soluciones.

1.4. Objetivos

El presente trabajo tiene varios objetivos: Hacer ingeniería inversa de ciertos aspectos básicos de la cognición [13]. Entender estos 3 algoritmos básicos de la computación cognitiva no sólo a nivel de teoría científica que descansa en un papel inerte sino también a nivel de tecnología que funciona. Demostrar que los algoritmos funcionan con aplicaciones prácticas. Y crear un conjunto de herramientas de programación en Haskell que puedan usarse en proyectos más ambiciosos [9].

1.5. Metodología

Estos algoritmos fueron inspirados indirectamente por la neurociencia [14]. 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 [15]. 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, lo cual garantiza su validez [9]. 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.

Se eligió Haskell para la programación de los algoritmos y las aplicaciones de esta tesis porque es un lenguaje puramente funcional y avanzado cuyas funciones tienen transparencia referencial (funciones 100% determinísticas y sin estado cambiante) y carecen de estados compartidos por otras funciones [9]. Los estados compartidos son peligrosos porque hacen que las funciones se comporten de manera no determinística y generan colisiones de acceso en sistemas multi-hilo de múltiple concurrencia. La carencia de estados compartidos a nivel de funciones permite la delegación de procesos en múltiples procesadores (paralelismo masivo) [16]. La transparencia referencial ayuda a razonar el comportamiento del programa, a probar su coherencia, a simplificar algoritmos y a

optimizar el código a través de memorización, eliminación de subexpresiones redundantes, evaluación vaga y paralelización masiva. Haskell es perfecto para programar todo tipo de sistema, especialmente para solucionar problemas difíciles y complejos como los sistemas de inteligencia artificial. Es impresionante como los programas casi siempre funcionan muy bien a la primera ejecución. Gracias a su sistema de tipos expresivo, inferencial, fuerte y estático, todos los errores se detectan al momento de la compilación, no en la ejecución [17]. Y si hay errores en la ejecución, estos son errores lógicos del programador que no expresó bien sus ideas. En otras palabras, no es culpa de Haskell.

Figura 1.2: Logotipo de Haskell [18]

1.6. Resultados Esperados

Se pretende explicar bien estos algoritmos básicos de la computación cognitiva y potenciarlos a través de la gran velocidad, confiabilidad y precisión de los computadores actuales. Se espera que para ciertas funciones cognitivas específicas, el rendimiento de los algoritmos superará el rendimiento humano.

También se quiere enfatizar la importancia del holismo en la inteligencia artificial y sus 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 [3,19].

CAPÍTULO 2

2. ALGORITMO: RECONSTRUCTOR DE IMÁGENES MENTALES

2.1. Introducción

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 [20]. La inducción es uno de los aspectos más

importantes de la inteligencia [12]. Por ejemplo, la percepción fluye en el sentido inverso de la causalidad: Va de sensaciones (efectos) a percepciones (causas). En el curso online “Visual Perception and the Brain”, el neurocientífico Dale Purves analizó varias modalidades de la percepción y explicó los mecanismos de la fenomenología en base al problema inverso de la inteligencia. Es decir, todo el curso de neurociencias lo dedicó a la inducción causal.

El reconstructor de imágenes mentales es una red causal conformada por un conjunto de campos de causalidad traslapados que al trabajar juntos pueden reconstruir una percepción de la realidad a partir de la evidencia presentada como sensaciones. Según esta idea, las redes de neuronas son vistas como dispositivos de causalidad donde los árboles dendríticos son los campos causales que capturan efectos (sensaciones) y las partes más profundas de la red sugieren causas (percepciones). El método de máxima entropía informacional podría resolver esta red causal de manera más eficiente que un análisis combinatorial de fuerza bruta cuyo orden algorítmico es exponencial [21]. En la tesis, se presenta 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 [22] 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.

2.2. Explicación Intuitiva del Algoritmo

El cerebro percibe la realidad a través de los campos causales traslapados de las neuronas [14]. Estos campos causales tienen una semántica asociada que sirve para inferir una imagen mental coherente de la realidad. Esta inferencia se realiza en las regiones cerebrales de mayor nivel y así sucesivamente las jerarquías van creciendo en nivel de complejidad. Los campos causales inicialmente son sensoriales o motrices, luego asociativos o multi-modales y luego conceptuales o de orden mayor [23].

Por el momento, la neurociencia no ha revelado el mecanismo exacto que explique precisamente cómo el cerebro infiere causas a partir de efectos. Pero existen muchas especulaciones con sus respectivos

modelos matemáticos [24]. En este capítulo se sugiere un algoritmo especulativo que funciona muy bien para ciertos tipos de inducción causal.

La programación tradicional es deductiva que es lo contrario a la inducción causal. Los programas tradicionales generalmente son un conjunto de causas que generan una gran cantidad de comportamientos o efectos. Por ejemplo: Los gráficos por computador (efectos) son generados por un conjunto de patrones y fórmulas matemáticas (causas) [25]. La inteligencia artificial general en cambio es inductiva porque esta observa los efectos generados por la realidad y trata de inducir o inferir el conjunto de causas que generaron tales efectos [12]. Por ejemplo: La visión por computador observa los fotones (efectos) que rebotan en los objetos (causas) y los algoritmos de inteligencia artificial infieren cómo los objetos (causas) están distribuidos en el entorno y cómo estos se interrelacionan entre sí (causas de orden mayor) [26]. Una vez finalizado el proceso de inducción causal, se puede decir que el computador entiende lo que observa.

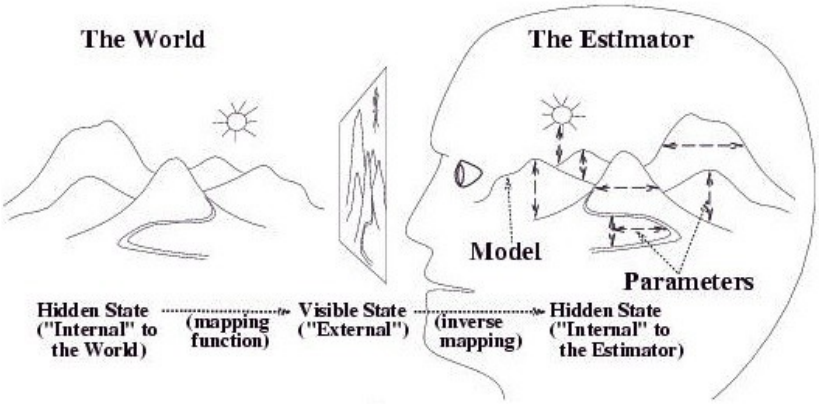


Figura 2.1: Cómo el cerebro reconstruye causalmente una imagen del entorno [27]

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 [3]. Y así mismo el tipo de campo causal podría ser exacto o flexible [28]. También podría ser continuo o discreto. O también puede ser jerárquico o plano [23]. O incluso puede ser

perceptivo o ejecutivo [23]. Y pueden haber diferentes tipos de clasificaciones de campos causales que todavía están por descubrirse.

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 [3]. 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 [22] 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).

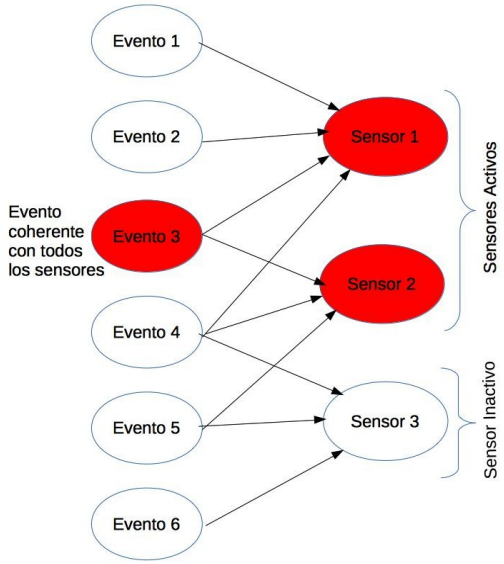


Figura 2.2: Ejemplo de campos causales traslapados

En la figura anterior se puede ver un ejemplo muy explicativo y sencillo de un campo causal traslapado de sensores. El sensor 1 está conectado causalmente con los eventos 1, 2, 3 y 4. El sensor 2 está conectado causalmente con los sensores 3, 4 y 5. Y el sensor 3 está conectado causalmente con los eventos 4, 5 y 6. Los sensores 1 y 2 están traslapados causalmente porque tienen en común los eventos 3 y 4. Los sensores 2 y 3 están traslapados causalmente porque tienen en común los eventos 4 y 5. Los sensores 1 y 3 están traslapados causalmente porque tienen en común el evento 4. Si se analiza los campos causales de los sensores 1 y 2, estos están activados y

quiere decir que los eventos coherentes que causaron sus activaciones son los eventos 3 y 4. Si se analiza los campos causales de los sensores 2 y 3, el sensor 2 está activo y el sensor 3 está inactivo. Esto quiere decir que de los eventos coherentes con el campo causal del sensor 2 hay que restar la intersección de eventos traslapados con el campo causal del sensor 3. En este caso, ya se había analizado previamente los campos causales de los sensores 1 y 2, como resultado se obtuvo que los eventos coherentes fueron los eventos 3 y 4. Si a estos eventos coherentes se resta la intersección de eventos traslapados con el campo causal del sensor 3, sólo quedaría 1 sólo sensor coherente con todos los campos causales: El evento 3.

El problema se vuelve más complejo a medida que se incrementan las posibles causas y sus respectivos campos causales. Las redes causales complejas pueden tener poco o mucho traslapamiento causal [19]. También pueden tener pocas o muchas causas [19]. O también pueden tener pocos o muchos campos causales [19]. O también pueden percibir el entorno o actuar sobre el entorno [23]. Incluso, en modelos causales muy complejos, los campos causales podrían tener múltiples jerarquías, anidamiento y composición [23].

Para simplificar la tesis, se explicará minuciosamente un sólo tipo de campo causal que es pre-programado (no aprendido), exacto (no flexible), discreto (no continuo), plano (sin jerarquías) y perceptivo (no ejecutivo). Es el tipo de campo causal que encontramos en el juego del buscaminas. Se recomienda jugar mucho el juego del buscaminas y volverse un experto en este para entender el siguiente capítulo de la tesis.

CAPÍTULO 3

3. APLICACIÓN: RESOLUCIÓN DEL JUEGO DE BUSCAMINAS EN POCOS SEGUNDOS

3.1. Introducción

El videojuego del buscaminas es tan famoso como Microsoft Windows y en realidad no necesita introducción. Pero para todos aquellos que no lo conocen, aquí va una pequeña explicación:

Básicamente, es un tablero con celdas en las cuales pueden haber

minas que explotan. Inicialmente todas las celdas están cubiertas o, en otras palabras, no se sabe si hay minas en ellas. Pero a medida que se va explorando el tablero, donde no hay minas alrededor aparecen espacios vacíos (correspondientes al número cero que se omite) y donde hay un cierto número de minas alrededor aparece un número entre el 1 y 8 como en la figura siguiente:

Figura 3.1: Juego del Buscaminas

Los números que aparecen son los campos causales del buscaminas. Los números significa que un sensor imaginario en el videojuego

indica la presencia de una cantidad de minas alrededor de la celda que contiene el número. Por lo general, cada celda tiene 3, 5 u 8 celdas vecinas. De estas celdas vecinas, un número N entre 1 y 8 indica cuántas de ellas son minas. Es decir, las minas son la causa y los sensores detectan el efecto; por lo tanto, es un campo causal. El objetivo del juego es inferir dónde están las minas (causas) en base a la evidencia de los sensores (efectos). Es un videojuego de inducción causal.

El juego termina de manera satisfactoria cuando se marcan correctamente la ubicación de las minas y se descubren todas las celdas que no tienen mina. Con el botón derecho del ratón se marcan las minas. Y con el botón izquierdo del ratón se descubren las celdas. En caso de descubrir una celda que contiene una mina, esta celda explota y se pierde el juego como en la figura siguiente:

Figura 3.2: Juego perdido del Buscaminas

Este es un juego probabilístico que su resultado depende parcialmente de la inteligencia del jugador, sea este un jugador humano o un algoritmo de inteligencia artificial. Hay juegos muy fáciles cuyas redes causales son muy claras y casi nada ambiguas. Estos juegos se resuelven muy fácilmente. Y también hay juegos cuyas redes causales son muy complejas y muy ambiguas. Tanto así que si el mejor jugador humano del mundo o el mejor algoritmo de inteligencia artificial tratan de resolver esos juegos muy complejos y muy ambiguos, podrían perder sin importar que tan inteligentes sean.

En esta tesis, se ha programado a un agente virtual que es mejor que cualquier ser humano que se atreva a intentar superarlo. En las pruebas de la tesis, los mejores tiempos de este agente virtual que juega automáticamente al buscaminas son los siguientes:

Figura 3.3: Mejores tiempos del reconstructor de imágenes mentales

Esta captura de pantalla no se trata de ningún efecto de Photoshop y ni de ningún tipo de trampa. En la sustentación de la tesis se demostrará que el sistema de inteligencia artificial puede lograr

tiempos similares que varían dependiendo que tan fácil o difícil sea el juego aleatorio en ese momento. Estos mejores tiempos corresponden a juegos cuyas redes causales fueron muy fáciles. Como se dijo previamente, el sistema de inteligencia artificial también puede perder debido a la naturaleza probabilística del videojuego, sin importar que el algoritmo de inducción causal sea matemáticamente óptimo.

3.2. Campos Causales del Buscaminas

Se analizará en detalle el algoritmo de inducción causal con un ejemplo muy sencillo:

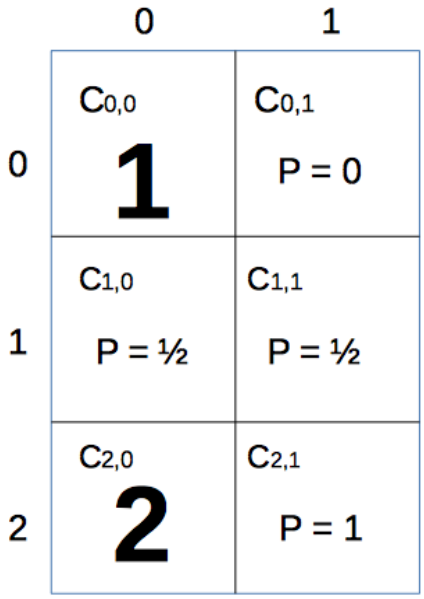


Figura 3.4: Primer ejemplo de un campo de minas

En la figura anterior se puede observar 2 campos causales generados por las celdas $C_{0,0}$ y $C_{2,0}$. Algebraicamente se pueden expresar de la siguiente manera:

$$C_{0,1}+C_{1,0}+C_{1,1}=1 \tag{3.1}$$

$$C_{1,0}+C_{1,1}+C_{2,1}=2 \tag{3.2}$$

Los valores de $C_{i,j}$ pueden ser uno o cero, dependiendo si hay una mina en la celda $C_{i,j}$ o no hay mina, respectivamente. Existen varios métodos para obtener las probabilidades y los patrones de minas que

son coherentes con los campos causales del problema. Por el momento, se explicará el análisis exhaustivo y combinatorial de fuerza bruta.

3.3. Análisis Exhaustivo y Combinatorial de Fuerza Bruta

El análisis exhaustivo y combinatorial de fuerza bruta es el método más sencillo de entender pero el más costoso en términos computacionales porque literalmente genera explosiones combinatoriales que demoran mucho en calcularse y consumen mucha memoria [2].

Tabla 1: Análisis exhaustivo y combinatorial de fuerza bruta

$C_{0,1}$	$C_{1,0}$	$C_{1,1}$	$C_{2,1}$	Coherente con $C_{0,1}+C_{1,0}+C_{1,1}=1$	Coherente con $C_{1,0}+C_{1,1}+C_{2,1}=2$
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1 C	1
0	0	1	1	1 C	2 C
0	1	0	0	1 C	1
0	1	0	1	1 C	2 C
0	1	1	0	2	2 C
0	1	1	1	2	3
1	0	0	0	1 C	0
1	0	0	1	1 C	1
1	0	1	0	2	1
1	0	1	1	2	2 C
1	1	0	0	2	1
1	1	0	1	2	2 C
1	1	1	0	3	2 C
1	1	1	1	3	3

Entonces, de todas las 16 (2^4) posibles combinaciones de patrones para las celdas $C_{0,1}$, $C_{1,0}$, $C_{1,1}$ y $C_{2,1}$, sólo 2 patrones son coherentes:

$$C_{0,1}=0, C_{1,0}=0, C_{1,1}=1, C_{2,1}=1 \tag{3.3}$$

$$C_{0,1}=0, C_{1,0}=1, C_{1,1}=0, C_{2,1}=1 \tag{3.4}$$

Para calcular las probabilidades de que las celdas contengan minas, simplemente se suman las minas por celda en cada patrón coherente y se las divide para el número de patrones coherentes. De esta forma, se obtiene la frecuencia relativa de los patrones coherentes:

$$P(C_{0,1})=0, P(C_{1,0})=\frac{1}{2}, P(C_{1,1})=\frac{1}{2}, P(C_{2,1})=1 \quad (3.5)$$

El número 16 (2^4) tiene un significado: Son 2 posibles valores multiplicados 4 veces. Las 4 veces corresponden al número de variables. Es decir, 16 = 2^4. La fórmula siguiente se puede extrapolar para más variables y más valores:

$$O(valores^{variables}) \quad (3.6)$$

Es decir, este algoritmo explota exponencialmente con el número de variables. Hay que buscar un algoritmo más eficiente. Si se dan cuenta, el número de patrones coherentes es mucho menor al número de posibles combinaciones de patrones. Y esto no es una simple coincidencia porque siempre ocurre lo mismo. Es decir, existe un algoritmo más eficiente: El análisis combinatorial con eliminación de patrones incoherentes que se explicará más adelante.

Si reemplazamos los patrones coherentes de (3.3) dentro de los campos causales de (3.1) y (3.2), obtenemos las siguientes ecuaciones:

$$(C_{0,1}=0)+(C_{1,0}=0)+(C_{1,1}=1)=1 \tag{3.7}$$

$$(C_{1,0}=0)+(C_{1,1}=1)+(C_{2,1}=1)=2 \tag{3.8}$$

Si reemplazamos los patrones coherentes de (3.4) dentro de los campos causales de (3.1) y (3.2), obtenemos las siguientes ecuaciones:

$$(C_{0,1}=0)+(C_{1,0}=1)+(C_{1,1}=0)=1 \tag{3.9}$$

$$(C_{1,0}=1)+(C_{1,1}=0)+(C_{2,1}=1)=2 \tag{3.10}$$

Si se suma (3.7) y (3.9), y al resultado se lo divide para el número de patrones coherentes que es 2, se obtiene las probabilidades (frecuencia relativa) de los patrones coherentes:

$$P(C_{0,1})+P(C_{1,0})+P(C_{1,1})=1 \tag{3.11}$$

Si se suma (3.8) y (3.10), y al resultado se lo divide para el número de patrones coherentes que es 2, se obtiene las probabilidades (frecuencia relativa) de los patrones coherentes:

$$P(C_{1,0})+P(C_{1,1})+P(C_{2,1})=2 \qquad (3.12)$$

Para los escépticos, simplemente comparen el procedimiento que generó (3.11) y (3.12) con el procedimiento que generó (3.5). Así de simple.

Al observar (3.11) y (3.12), uno se da cuenta que hay 4 variables y 2 ecuaciones. Esto crea un subespacio vectorial de R^n. Y si agregamos las desigualdades asociadas a cada probabilidad ($0 \leqslant P(X_i) \leqslant 1$), obtenemos una porción de un subespacio vectorial de R^n [29].

Para resolver esta ambigüedad se podría aplicar el teorema de máxima entropía informacional (la entropía de Shannon en (3.13)) que distribuye probabilidades similares de manera equitativa [21,30]. De hecho, las probabilidades de las celdas $C_{1,0}$ y $C_{1,1}$ son iguales. Y lo mismo ocurre con probabilidades similares en redes causales más complejas. Pero este método genera ecuaciones logarítmicas que se maximizan con el método de multiplicadores de Lagrange y cuya resolución es un tanto complicada de automatizar porque no se generan fórmulas exactas sino más bien es necesario analizar

desigualdades dado que las probabilidades correspondientes a la máxima entropía caen fuera del rango permitido de las probabilidades: $0 \leqslant P(X_i) \leqslant 1$

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i) \tag{3.13}$$

Entonces hay que utilizar otros métodos más sencillos como el análisis exhaustivo y combinatorial de fuerza bruta y finalmente el análisis combinatorial con eliminación de patrones incoherentes. También se podría resolver el problema con estadística bayesiana [31].

3.4. Análisis Combinatorial con Eliminación de Patrones Incoherentes

Para el análisis combinatorial con eliminación de patrones incoherentes, se usará un ejemplo no tan simple de un campo de minas:

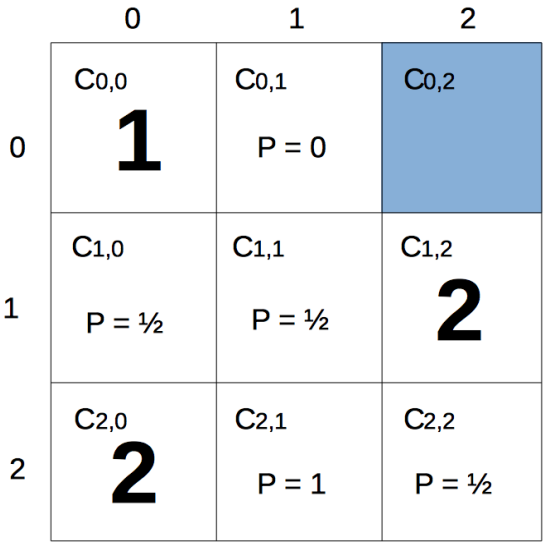


Figura 3.5: Segundo ejemplo de un campo de minas

En este ejemplo, hay 5 variables: $C_{0,1}$, $C_{1,0}$, $C_{1,1}$, $C_{2,1}$ y $C_{2,2}$. Y hay 3 campos traslapados de causalidad (3 sensores de minas) que se describen con las siguientes ecuaciones algebraicas:

$$C_{0,1}+C_{1,0}+C_{1,1}=1 \tag{3.14}$$

$$C_{1,0}+C_{1,1}+C_{2,1}=2 \tag{3.15}$$

$$C_{0,1}+C_{1,1}+C_{2,1}+C_{2,2}=2 \tag{3.16}$$

En el análisis combinatorial con eliminación de patrones incoherentes, no se analiza simultáneamente todos los campos causales, sino más

bien uno por uno.

Tabla 2: Análisis combinatorial del primer campo causal

$C_{0,1}$	$C_{1,0}$	$C_{1,1}$	**Coherente con $C_{0,1}+C_{1,0}+C_{1,1}=1$**
0	0	0	0
0	0	1	1 C
0	1	0	1 C
0	1	1	2
1	0	0	1 C
1	0	1	2
1	1	0	2
1	1	1	3

Luego se analiza el segundo campo causal pero esta vez sólo se toma en cuenta a los 3 patrones coherentes del análisis anterior. Existen 2 variables en común entre los campos de causalidad traslapados: $C_{1,0}$ y $C_{1,1}$. Para el segundo análisis combinatorial, estas variables ya fueron combinadas y no es necesario combinarlas nuevamente. Pero hay una variable nueva $C_{2,1}$ en el segundo campo causal con la cual es necesario encontrar sus posibles combinaciones con los patrones coherentes anteriores:

Tabla 3: Análisis combinatorial del segundo campo causal sin los patrones incoherentes del análisis anterior

$C_{2,1}$	$C_{0,1}$	$C_{1,0}$	$C_{1,1}$	**Coherente con $C_{0,1}+C_{1,0}+C_{1,1}=1$**	**Coherente con $C_{1,0}+C_{1,1}+C_{2,1}=2$**
0	0	0	1	1 C	1
0	0	1	0	1 C	1
0	1	0	0	1 C	0
1	0	0	1	1 C	2 C
1	0	1	0	1 C	2 C
1	1	0	0	1 C	1

Luego se analiza el tercer campo causal pero esta vez sólo se toma en cuenta a los 2 patrones coherentes del análisis anterior. Existen 3 variables en común entre el tercer campo causal y las variables previamente combinadas: $C_{0,1}$, $C_{1,1}$ y $C_{2,1}$. Para el tercer análisis combinatorial, estas variables ya fueron combinadas y no es necesario combinarlas nuevamente. Pero hay una variable nueva $C_{2,2}$ en el tercer campo causal con la cual es necesario encontrar sus posibles combinaciones con los patrones coherentes anteriores:

Tabla 4: Análisis combinatorial del tercer campo causal sin los patrones incoherentes del análisis anterior

$C_{2,2}$	$C_{2,1}$	$C_{0,1}$	$C_{1,0}$	$C_{1,1}$	Coherente con $C_{0,1}+C_{1,0}+C_{1,1}=1$	Coherente con $C_{1,0}+C_{1,1}+C_{2,1}=2$	Coherente con $C_{0,1}+C_{1,1}+C_{2,1}+C_{2,2}=2$
0	1	0	0	1	1 C	2 C	2 C
0	1	0	1	0	1 C	2 C	1
1	1	0	0	1	1 C	2 C	3
1	1	0	1	0	1 C	2 C	2 C

Entonces, de todas las 32 (2^5) posibles combinaciones de patrones para las celdas $C_{0,1}$, $C_{1,0}$, $C_{1,1}$, $C_{2,1}$ y $C_{2,2}$, se han examinado 8+6+4=18 patrones y de estos, sólo 2 patrones son coherentes:

$$C_{0,1}=0, C_{1,0}=0, C_{1,1}=1, C_{2,1}=1, C_{2,2}=0 \tag{3.17}$$

$$C_{0,1}=0, C_{1,0}=1, C_{1,1}=0, C_{2,1}=1, C_{2,2}=1 \tag{3.18}$$

Para calcular las probabilidades de que las celdas contengan minas, simplemente se suman las minas por celda en cada patrón coherente y se las divide para el número de patrones coherentes. De esta forma, se obtiene:

$$P(C_{0,1})=0, P(C_{1,0})=\frac{1}{2}, P(C_{1,1})=\frac{1}{2}, P(C_{2,1})=1, P(C_{2,2})=\frac{1}{2} \tag{3.19}$$

Una vez obtenidas las probabilidades de la frontera en la búsqueda

de minas, simplemente se les da preferencia a las probabilidades deterministas (P(X) = 0 ó P(X) = 1). Se descubren las celdas que no tienen minas (P(X) = 0) y se marcan las celdas que tienen minas (P(X) = 1). Si no existen probabilidades deterministas, se descubre sólo una celda: La celda con la mínima probabilidad de que tenga mina. La elección se realiza entre las celdas de la frontera y las celdas cubiertas que no están en la frontera. La probabilidad de que una celda que no está en la frontera tenga mina es la siguiente [32]:

$$P(celda\ fuera\ de\ frontera) = \frac{nmsm - spf}{ncff} \tag{3.20}$$

Donde:

nmsm = número de minas sin marcar

spf = suma de probabilidades en la frontera

ncff = número de celdas fuera de la frontera

Las fronteras pueden ser muy complicadas y tener muchas celdas. Lo cual dificulta el cálculo de probabilidades. Para esto, es necesario hacer un análisis de independencia causal que separe la frontera de probabilidades en clusters o agrupaciones que son más fáciles de analizar. No es necesario calcular toda la frontera de probabilidades

en conjunto si existe independencias causales que particionan la frontera en varios clusters independientes.

Por cierto, el orden de complejidad del algoritmo es el siguiente:

$$O\left(2^{npvcc} + \left(cc-1\right)\cdot nppccc\cdot 2^{npvcc-npvyecc}\right) \quad (3.21)$$

Donde:

cc = número de campos causales

nppccc = número promedio de patrones coherentes por campo causal

npvcc = número promedio de variables por campo causal

npvyecc = número promedio de variables ya examinadas por campo causal

El algoritmo parece exponencial pero los exponentes son pequeños y NO crecen mucho con el número de variables en la frontera. Si la frontera crece, el valor que crece más es (cc-1)*nppccc. Esta es una demostración de cómo un algoritmo con O(NP) se puede transformar en un algoritmo con O(P). En ciertos algoritmos de planeación, búsqueda y creatividad, también se puede aplicar un método similar. Pero ese análisis es tema de otra tesis.

(3.21) tiene 2 términos. El primer término se calcula en base a la primera búsqueda de patrones coherentes en el primer campo causal a examinar. Y el segundo término se calcula en base a las (cc-1) búsquedas de patrones coherentes en los demás campos causales. Cada búsqueda examina a los patrones coherentes anteriores multiplicados por todas las posibles combinaciones de variables por examinar.

El algoritmo tiene un muy buen tiempo, pero cuando las fronteras son muy grandes, el minesweeper-solver puede demorar mucho en resolver la frontera:

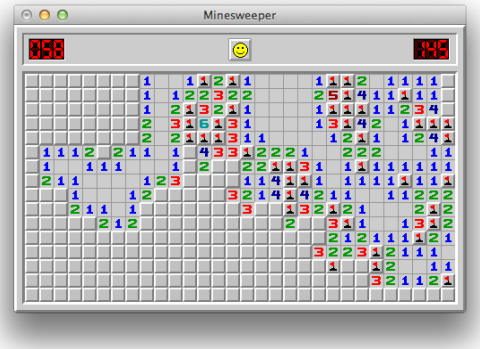


Figura 3.6: El algoritmo tarda mucho en resolver fronteras muy grandes

En fronteras tan grandes como esta, lo mejor que se puede hacer es particionar la frontera arbitrariamente en clusters más pequeños. Se pierde algo de precisión en el cálculo de las probabilidades. Pero es muy probable que en una frontera tan grande hayan varias probabilidades determinísticas (P(Ci)=0 ó P(Ci)=1); lo cual hace innecesario que se calcule las probabilidades no determinísticas de manera exacta.

3.5. Riesgo Total de cada Juego

Para calcular el riesgo total de cada juego, es necesario observar este modelo gráfico probabilístico [15] donde las X representan los juegos perdidos o explosiones:

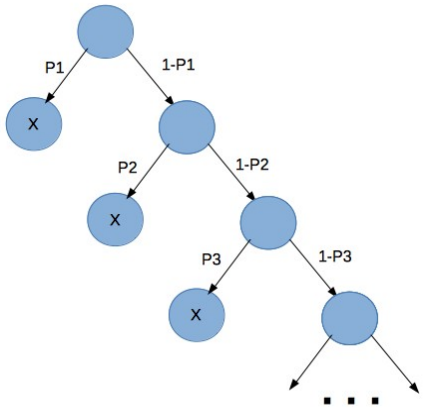


Figura 3.7: Modelo gráfico probabilístico para calcular el riesgo total de cada juego

Observando bien la figura anterior, el riesgo total que se toma en cada juego se calcula con las siguientes fórmulas:

$$P(ganar) = \prod_{i=2}^{n} (1 - P_i) \tag{3.22}$$

$$Riesgo\,Total = 1 - P(ganar) \tag{3.23}$$

Donde:

P_i = Riesgo tomado en el intento i

El riesgo en el primer click no se toma en cuenta porque el juego del buscaminas fue programado en Haskell para que el primer click

siempre tenga cero minas alrededor y así se abra una buena porción de frontera. Por esto, el índice del multiplicatorio comienza con i=2 y no con i=1.

3.6. Uso y Demostración de la Aplicación

Primero se abre un terminal. Se entra al directorio de la tesis. Luego se ejecuta el script minesweeper.sh. Inicialmente, la dificultad del buscaminas es el nivel experto. Pero uno puede cambiar la dificultad con el botón intermedio del ratón. Para emular el botón intermedio del ratón en Mac, se presiona la tecla “alt” más click izquierdo. Para emular el botón intermedio del ratón en Ubuntu, se hace click con los 2 botones del ratón a la vez. Una vez presionado o emulado el botón intermedio del ratón, aparece un menú popup para seleccionar el nivel de dificultad: Principiante, intermedio y avanzado.

Posteriormente, se abre otra ventana de terminal. Se entra al directorio de la tesis. Se verifica si el juego del buscaminas es nuevo y si no lo es, se presiona el botón de la carita amarilla para renovarlo. Es importante que el juego sea nuevo porque el reconocimiento de la ventana del buscaminas sólo es posible en este estado. La ventana

del juego debe estar completamente visible para su posterior captura. Hay que evitar que la sombra de las otras ventanas caiga sobre la ventana del buscaminas. Si una sombra apenas cubre al buscaminas, el programa que resuelve el buscaminas no podrá ver bien al juego del buscaminas y habrán errores de visión. Luego se ejecuta el script minesweeper-solver-for-<OS>.sh donde <OS> es el sistema operativo con el que se trabaja. Sólo hay soporte para 2 sistemas operativos: Mac OS X y Ubuntu.

Al ejecutarse el minesweeper-solver, este pregunta cuántos juegos se desea resolver. Luego este resuelve el juego del buscaminas automáticamente. Lee muy rápido la captura de pantalla, calcula las probabilidades de la frontera y automáticamente hace clicks muy rápidos en los lugares que corresponden para intentar ganar. En cada sesión, se juega N veces y al final aparecen las estadísticas de los tiempos y resultados. Además, en el directorio “screen-capture-for-minesweeper” se guarda en subdirectorios con fecha y hora las capturas de pantalla de los resultados de los juegos.

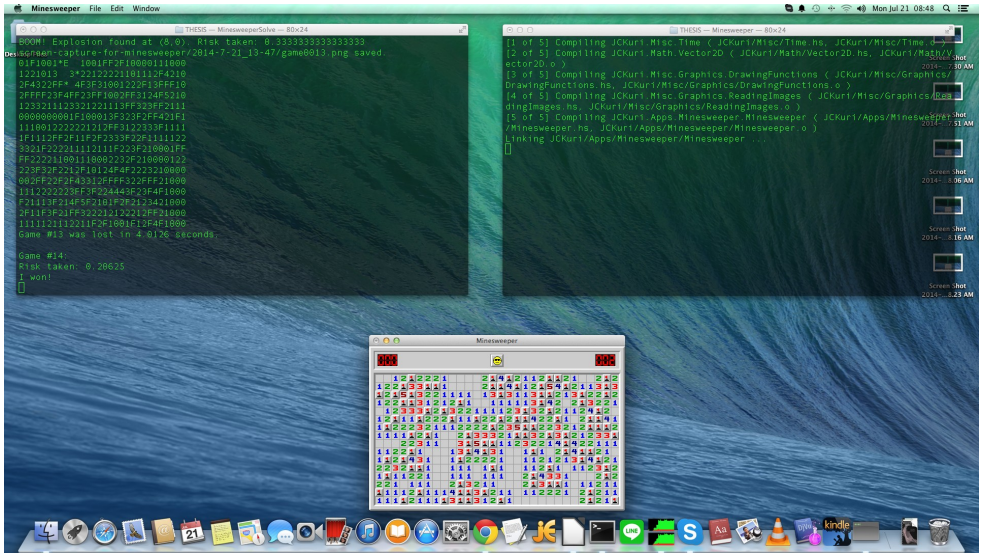


Figura 3.8: Demostración del Minesweeper y el Minesweeper Solver en acción

3.7. Estadísticas del Buscaminas

El sistema de inteligencia artificial resolvió el juego del buscaminas 600 veces: 6 series de 100 juegos cada una. Aquí están los resultados:

Tabla 5: Estadísticas del Buscaminas

Serie de Juegos:	Nivel:	Tipo de Juegos:	Número de Juegos:	Tiempo en Segundos:		Riesgo Total:		Número de Clicks:	
				Promedio:	Desviación Estándar:	Promedio:	Desviación Estándar:	Promedio:	Desviación Estándar:
1	Experto	Perdidos	70	4.1808	2.9564	0.5048	0.2002	267.6571	105.2146
1	Experto	Ganados	30	4.3142	1.4130	0.2774	0.3097	333.9667	16.4725
1	Experto	Todos	100	4.2208	2.5846	0.4366	0.2588	287.55	93.4222
2	Experto	Perdidos	62	4.1429	5.3462	0.5084	0.2293	240.9516	107.9836
2	Experto	Ganados	38	4.4261	1.4608	0.3925	0.3390	324.5526	25.4396
2	Experto	Todos	100	4.2505	4.2928	0.4643	0.2802	272.72	95.3408
3	Intermedio	Perdidos	17	1.1914	0.2710	0.4602	0.1585	117.2941	29.3657
3	Intermedio	Ganados	83	1.2369	0.2603	0.0939	0.1896	137.4096	14.2880
3	Intermedio	Todos	100	1.2292	0.2613	0.1562	0.2301	133.99	19.1346
4	Intermedio	Perdidos	20	1.1987	0.3509	0.4389	0.1773	119.1	30.5973
4	Intermedio	Ganados	80	1.1570	0.2227	0.0714	0.1694	130.7625	15.2068
4	Intermedio	Todos	100	1.1653	0.2520	0.1449	0.2253	128.43	19.6516
5	Principiante	Perdidos	3	0.3817	0.1093	0.4444	0.0962	21.0	1.0
5	Principiante	Ganados	97	0.3775	0.1190	0.0257	0.1095	31.9588	8.2537
5	Principiante	Todos	100	0.3776	0.1182	0.0383	0.1303	31.63	8.3432
6	Principiante	Perdidos	8	0.4490	0.1778	0.4644	0.1677	23.625	12.3397
6	Principiante	Ganados	92	0.4043	0.1180	0.0303	0.1206	33.2282	7.7744
6	Principiante	Todos	100	0.4079	0.1233	0.0651	0.1714	32.46	8.5545

Tabla 6: Más Estadísticas del Buscaminas

Serie de Juegos:	Record de Ganar:			El Juego Más Lento:				Número de Juegos:	Tiempo Total en Segundos:
	Tiempo en Segundos:	Número de Clicks:	Riesgo Tomado:	Tiempo en Segundos:	Resultado:	Número de Clicks:	Riesgo Tomado:		
1	2.4595	316	0.0	21.5702	Perdido	228	0.8742	100	422.0810
2	2.6427	315	0.0	43.4602	Perdido	307	0.6565	100	425.0510
3	0.7058	102	0.0	1.9334	Ganado	148	0.0	100	122.9161
4	0.6621	103	0.0	2.307	Perdido	156	0.6667	100	116.5306
5	0.172	16	0.0	0.9915	Ganado	41	0.2186	100	37.7655
6	0.1697	17	0.0	0.9573	Ganado	55	0.5873	100	40.7891

CAPÍTULO 4

4. ALGORITMO: CARACTERIZADOR EVOLUTIVO

4.1. Introducción

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 [20]. 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 [11]. 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. Según el neurocientífico Gerald Edelman, el cerebro funciona mediante un principio llamado darwinismo neuronal. Los memes, la contraparte mental de los genes, evolucionan en la mente combinando, mutando y seleccionando patrones de actividad neuronal que representan percepciones y planes [33,34,35].

4.2. Explicación Matemática del Algoritmo

El caracterizador evolutivo es la generalización de la regresión lineal, regresión polinómica y redes neuronales [20]. Para generalizar algoritmos, es necesario pensar de manera abstracta tratando de describir los aspectos comunes de todos estos algoritmos y perdiendo algunos detalles específicos.

Básicamente, se tiene una superficie parametrizable F que es

continua y derivable:

$$F:\mathbb{R}^{n_i+n_p}\rightarrow\mathbb{R}^{n_o} \tag{4.1}$$

$$F(i_1,i_2,\ldots,i_{n_i},p_1,p_2,\ldots,p_{n_p})=(o_1,o_2,\ldots,o_{n_o}) \tag{4.2}$$

$$F(I,P)=O \tag{4.3}$$

I e P son los hiper-vectores de entrada.

$$I=(i_1,i_2,\ldots,i_{n_i}) \tag{4.4}$$

$$P=(p_1,p_2,\ldots,p_{n_p}) \tag{4.5}$$

O es el hiper-vector de salida.

$$O=(o_1,o_2,\ldots,o_{n_o}) \tag{4.6}$$

$Pair_i$ es el patrón que asocia un hiper-vector de entrada I_i con un hiper-vector de salida O_i.

$$Pair_i=(I_i,O_i) \tag{4.7}$$

Si tenemos una lista de patrones asociativos {$Pair_1$, $Pair_2$, … , $Pair_n$},

el objetivo es encontrar los parámetros P que maximizen la igualdad de F y que minimizen su error de (4.8) [20].

Normalmente, las hiper-funciones o campos vectoriales en R^n como F van desde un hiper-vector de entrada y generan un hiper-vector de salida. Ese es el método deductivo que va desde la causa al efecto. Pero como se desea programar inteligencia artificial, el objetivo es hacer inducción causal que va desde el efecto (un hiper-vector de salida) hacia las posibles causas (varios hiper-vectores de entrada). Los hiper-vectores de entrada tienen 2 componentes: Las constantes I y las variables o parámetros P. Es posible que una optimización no tenga constantes I, sólo hiper-vectores de salida O y parámetros P.

Al igual que la regresión lineal, este método inductivo emplea el método de optimización de mínimos cuadrados [2]. En donde se minimiza la suma del cuadrado de los errores o diferencias entre los hiper-vectores de salida reales O y los hiper-vectores de salida calculados por la función:

$$Error = \sum_{i=1}^{n} \left\| O_i - F\left(I_i, P\right) \right\|^2 \qquad (4.8)$$

Si uno desea calcular los parámetros P óptimos que minimizen localmente o globalmente la curva del error, es necesario calcular el gradiente del error con respecto a los parámetros P [36]:

$$\nabla Error = \frac{\partial Error}{\partial P} = \left(\frac{\partial Error}{\partial p_1}, \frac{\partial Error}{\partial p_2}, \ldots, \frac{\partial Error}{\partial p_{n_p}}\right) \tag{4.9}$$

Las derivadas parciales del error con respecto a los parámetros pueden aproximarse numéricamente con la siguiente ecuación [36]:

$$\frac{\partial Error(p_1, p_2, \ldots, p_{n_p})}{\partial p_i} = \lim_{\Delta P \to 0} \frac{Error(p_1, p_2, \ldots, p_i + \Delta p, \ldots, p_{n_p}) - Error(p_1, p_2, \ldots, p_i - \Delta p, \ldots, p_{n_p})}{2 \cdot \Delta p} \tag{4.10}$$

Luego se iteran los parámetros para que desciendan por la curva del error hasta llegar a un mínimo local o global [37]:

$$P_{n+1} = P_n - factor \cdot \frac{\nabla Error}{\|\nabla Error\|} \tag{4.11}$$

Hasta aquí no hay nada nuevo, es simplemente la técnica del descenso de gradiente aplicada a la curva del error cuadrático de un campo vectorial F(I,P) = O. Pero este algoritmo se vuelve más rápido y más preciso cuando los factores que multiplican al gradiente se ajustan por separado. Un ejemplo de este fenómeno se ve claramente

cuando en esta tesis se evolucionaban en conjunto parámetros de ángulos y parámetros de coordenadas cartesianas. Cuando el factor de descenso común a todos estos parámetros era muy grande, los ángulos saltaban sin coherencia. Y cuando el factor de descenso era muy pequeño de tal manera que los ángulos evolucionaban coherentemente, las coordenadas cartesianas evolucionan demasiado lento. Al separar un factor de descenso para los ángulos y otro factor de descenso para las coordenadas, la evolución mejoró tremendamente, no sólo en precisión sino también en rapidez. Y cuando se separaron todos los factores de descenso a un factor por variable, la precisión y la rapidez mejoraron aún más.

$$(p_1, p_2, \ldots, p_{n_p})_{n+1} = (p_1, p_2, \ldots, p_{n_p})_n - \frac{1}{\|\nabla Error\|} \cdot \left(factor_1 \cdot \frac{\partial Error}{\partial p_1}, factor_2 \cdot \frac{\partial Error}{\partial p_2}, \ldots, factor_{n_p} \cdot \frac{\partial Error}{\partial p_{n_p}}\right) \quad (4.12)$$

Los factores de descenso se los ajusta tentativamente por ensayo y error. Cada factor se lo prueba 3 veces por iteración: Un factor un tanto mayor para acelerar el descenso, el mismo factor anterior para continuar el descenso a la misma velocidad y un factor un tanto menor para frenar un poco el descenso y ganar precisión. De los 3 factores a probar, se escoge el factor que produce el menor error. Es decir, si un factor mayor produce un menor error que los demás, es

necesario acelerar un poco la evolución. Y si un factor menor produce un menor error que los demás, es necesario frenar un poco porque nos acercamos a un mínimo local o global, o a un punto silla [36]. Entonces es necesario evolucionar lentamente para ganar precisión.

Hablando en términos matemáticos, se define un factor para saltar órdenes de magnitud en pequeños intervalos. Es decir, cada orden de magnitud implica una multiplicación o división por 10. Pero se puede escalar o descender en intervalos más pequeños con la siguiente ecuación:

$$FactorMagnitud = 0.1^{\frac{1}{intervalos}} \tag{4.13}$$

En base al factor anterior, se definen los 3 factores de evolución: El que frena, el estable y el que acelera:

$$FactorFrenando_n = factor_{n-1} \cdot FactorMagnitud \tag{4.14}$$

$$FactorEstable_n = factor_{n-1} \tag{4.15}$$

$$FactorAcelerando_n = \frac{factor_{n-1}}{FactorMagnitud} \tag{4.16}$$

Y luego se elige el factor que produce el mínimo error:

$$NuevoFactor = \underset{x \in \{ FactorFrenando, FactorEstable, FactorAcelerando \}}{argmin} Error(x) \tag{4.17}$$

La evolución de un grupo de parámetros finaliza cuando los factores de descenso se vuelven muy pequeños. Eso es un indicador de que ya hubo convergencia. La convergencia puede darse en un mínimo local o global dependiendo de los parámetros iniciales. Es decir, el espacio de patrones está particionado por campos de atracción. Para explorar más exhaustivamente el espacio de patrones y sus mínimos locales y global, es necesario regar una lluvia de patrones iniciales que poco a poco evolucionan, descienden y se acumulan en los mínimos.

Por ahora, todo parece tan abstracto. En el siguiente capítulo se explicarán 2 ejemplos prácticos del caracterizador evolutivo en acción: La caracterización de puntos de referencia elípticos y la caracterización del campo vectorial de la visión.

Vale la pena mencionar que en el curso online “Coding the Matrix: Linear Algebra through Applications to Computer Science” [38], el

Prof. Philip Klein indica que la regresión lineal es una proyección del plano parametrizable con respecto a los hiper-puntos a ajustar con la curva del error. 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 [28]: “The entire effort of artificial intelligence is essentially a fight against computers’ rigidity.”

CAPÍTULO 5

5. APLICACIÓN: SISTEMA BÁSICO DE VISIÓN ARTIFICIAL

5.1. Introducción

Básicamente, 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. Un programa como QuickTime para Mac (Opción: File → New Movie Recording) o Cheese Webcam Booth para Ubuntu. El objetivo es capturar una foto de una hoja con puntos

de referencia circulares como en la Figura 5.1 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.

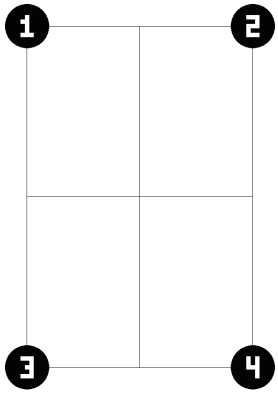


Figura 5.1: Hoja con puntos de referencia

Normalmente cuando se programa gráficos tridimensionales por computadora, las fórmulas de la perspectiva nos permiten ir de las coordenadas tridimensionales de los objetos (causas) hacia las coordenadas bidimensionales de la proyección en el plano de visión (efectos) [25]. Es un proceso de deducción. Pero cuando se programa inteligencia artificial, el proceso se invierte. Se transforma en un proceso de inducción causal que va de los efectos a las causas. En

este sistema básico de visión artificial, la inteligencia artificial nos permite ir de los puntos bidimensionales (efectos) proyectados en el plano de la visión hacia los parámetros tridimensionales de la cámara que generaron dicha proyección (causas).

Figura 5.2: Captura de pantalla de un programa que permite ver lo que una webcam filma

En resumen el usuario captura la imagen que filma la webcam desde un punto de vista como en la Figura 5.2. Luego el sistema básico de visión artificial ubica bien los puntos de referencia con ayuda del usuario. (Esto paso se explicará en el subcapítulo “Uso y Demostración de la Aplicación”) Para después inferir el punto de vista y la dirección de la webcam.

5.2. Caracterización de los Puntos de Referencia

Se puede decir que 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.

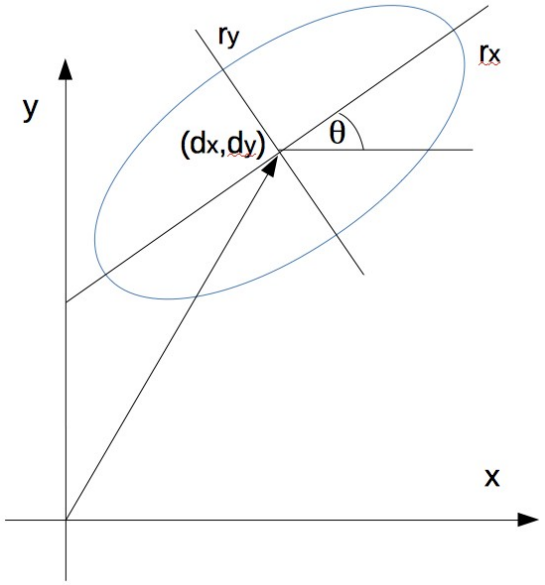


Figura 5.3: Elipse escalada, rotada y trasladada en el plano 2D

Tradicionalmente, en sistemas deductivos uno calcula la elipse en el siguiente orden: Escalación, rotación y traslación. Pero en un sistema

inductivo, el orden cambia y sus operaciones también se vuelven inversas: Traslación (el punto menos el desplazamiento), rotación (sentido contrario a las manecillas del reloj) y escalación (división para los radios de la elipse).

Para caracterizar la elipse rotada, se usa la función radio inverso que toma como parámetros las coordenadas (x,y) de cualquier punto en el plano 2D y dice si este punto está dentro de la elipse (RadioInverso < 1) o fuera de la elipse (RadioInverso > 1). En términos matemáticos, el radio inverso de la elipse es:

$$RadioInverso(x,y)=\left\| \begin{bmatrix} \frac{1}{rx} & 0 \\ 0 & \frac{1}{ry} \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{pmatrix} x-dx \\ y-dy \end{pmatrix} \right\| \quad (5.1)$$

La función sigmoide de (5.2) y Figura 5.4, también conocida en el campo de las redes neuronales como la función desigualdad continua, es de gran utilidad para determinar el color correspondiente de los pixeles que dibujan a la elipse y al fondo que rodea a la elipse. Dentro de la elipse es gris oscuro casi negro (aproximadamente 0) y fuera de la elipse es gris claro casi blanco (aproximadamente 1) [37].

$$sigmoide(x, inclinación, desplazamiento) = \frac{1}{1+e^{inclinación*(x-desplazamiento)}} \quad (5.2)$$

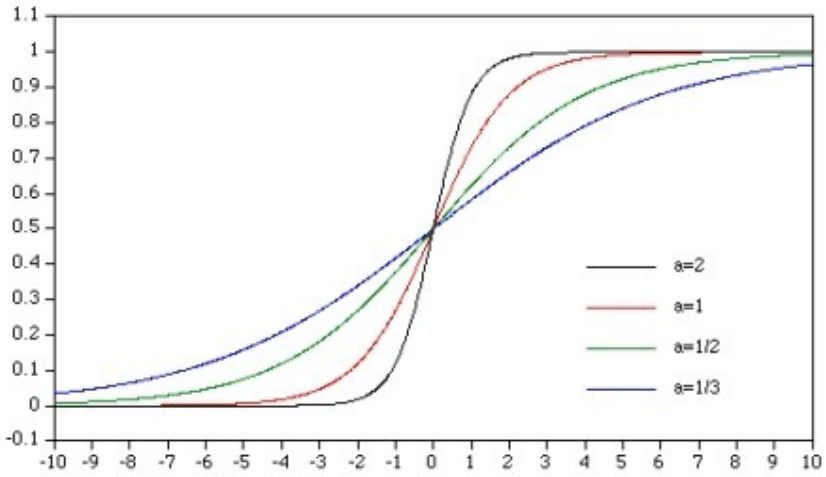


Figura 5.4: Función sigmoide con diferentes inclinaciones [39]

El color negro es menos susceptible a los cambios de iluminación que el color blanco. Para saber la correcta tonalidad de la elipse y del fondo que rodea a la elipse, es necesario usar un algoritmo de clustering que clasifica 2 clusters: los colores parecidos al negro (dentro de la elipse) y los colores parecidos al blanco (fondo que rodea a la elipse).

En la tesis se usa un algoritmo de clustering aglomerativo que empieza con N grupos de 1 elemento cada uno y luego va agrupando

los grupos con menor diferencia de promedios o elementos estereotípicos hasta que se obtienen un número de grupos fijo. En el caso de los colores descrito anteriormente, sólo hay 2 grupos: El grupo de pixeles parecido al color negro y el grupo de pixeles parecido al color blanco.

$$\text{color}(x,y) = negro + (blanco - negro) \cdot sigmoide(RadioInverso(x,y), 15, 1) \quad (5.3)$$

Con la ecuación anterior se obtiene el color en escala de grises para cada punto en el plano 2D. El parámetro 15 en la inclinación significa que la pendiente es muy empinada. Y el parámetro 1 en el desplazamiento significa que la división de regiones ocurre cuando el radio inverso es 1, como se dijo anteriormente.

Es importante que esta función sea continua para que pueda ser usada por la técnica de descenso de gradiente. Por eso sus regiones, fuera y dentro de la elipse, fueron separadas por la función sigmoide y no con funciones de pertenencias basadas en < (menor que) ó > (mayor que) que no son continuas.

Es importante darse cuenta que esta fórmula caracteriza elipses rotadas completamente negras y no tienen números dentro de ellas.

De hecho, los números son errores que son ignorados gracias a la flexibilidad del caracterizador evolutivo.

Una vez que el usuario y el sistema inteligente han localizado bien los puntos de referencia, se procede a inferir los parámetros del campo vectorial de la visión.

5.3. Caracterización del Campo Vectorial de la Visión

La caracterización del campo vectorial de la visión se realiza en cinco pasos: Encontrar una fórmula de rotación generalizada en 3D. Rotar la base ortonormal (i,j,k) en torno a j (ángulo A), en torno al i rotado (ángulo B) y en torno al k rotado (ángulo C). Trasladar los puntos 3D de los objetos de tal forma que la ubicación de la cámara quede en el origen de ese espacio. Proyectar los puntos 3D trasladados sobre la base ortonormal rotada [29]. Calcular la perspectiva en base al método de los triángulos rectángulos.

El orden de estos pasos es importante porque con la fórmula de rotación generalizada en 3D se rotan las bases ortonormales en torno a j (ángulo A), i (ángulo B), y k (ángulo C). Con estas bases

ortonormales rotadas se proyectan los puntos de los objetos [29]. Pero antes de proyectar los puntos, se debe trasladarlos para que la cámara sea el centro del mundo virtual como lo es el ojo humano. Una vez proyectados los puntos, se calcula su perspectiva para que los objetos cercanos se vean grandes y los objetos lejanos se vean pequeños, con respecto a la cámara caracterizada con las bases rotadas y trasladadas.

La ecuación de rotación generalizada es la siguiente:

$$p' = (1-\cos\theta)\cdot\left(n+\left[\frac{r\cdot(p-n)}{r\cdot r}\right]\right)+\cos\theta\cdot p+\frac{r\times(p-n)}{\|r\|}\cdot\sin\theta \qquad (5.4)$$

Donde:

$$p = punto\ a\ rotar$$

$$p' = punto\ rotado\ en\ 3\mathrm{D}$$

$$\theta = ángulo\ de\ rotación$$

$$n, r = vectores\ de\ la\ recta\ de\ rotación\ \mathrm{RR(t)} = n + r\cdot t$$

Gráficamente, se puede observar:

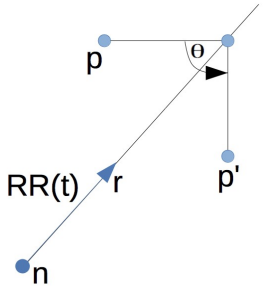


Figura 5.5: Rotación generalizada en 3D

Al rotar la base ortonormal (i,j,k) en torno a j con un ángulo A, obtenemos la base ortonormal (i_2,j_2,k_2). Al rotar la base ortonormal (i_2,j_2,k_2) en torno a i_2 con un ángulo B, obtenemos la base ortonormal (i_3,j_3,k_3). Y al rotar la base ortonormal (i_3,j_3,k_3) en torno a k_3 con un ángulo C, obtenemos la base ortonormal (i_4,j_4,k_4). O mejor dicho (i_R,j_R,k_R) donde el subíndice R significa rotado. El resultado es el siguiente:

$$i_R = \begin{pmatrix} \sin A \cdot \sin B \cdot \sin C + \cos A \cdot \cos B \\ \cos B \cdot \sin C \\ \cos A \cdot \sin B \cdot \sin C - \sin A \cdot \cos B \end{pmatrix} \tag{5.5}$$

$$j_R = \begin{pmatrix} \sin A \cdot \sin B \cdot \cos C - \cos A \cdot \sin C \\ \cos B \cdot \cos C \\ \cos A \cdot \sin B \cdot \cos C + \sin A \cdot \sin C \end{pmatrix} \tag{5.6}$$

$$k_R = \begin{pmatrix} \sin A \cdot \cos B \\ -\sin B \\ \cos A \cdot \cos B \end{pmatrix} \tag{5.7}$$

Entonces cada punto de cada objeto debe ser trasladado al sistema de coordenadas de la cámara. Para esto a cada punto se le resta la posición de la cámara:

$$v = p - o_c \tag{5.8}$$

Y esta resta vectorial se proyecta con respecto a la base ortonormal rotada [29]:

$$H = \{ i_R, j_R, k_R \} \tag{5.9}$$

$$proy_H v = (v \cdot i_R) i_R + (v \cdot j_R) j_R + (v \cdot k_R) k_R \tag{5.10}$$

A los puntos proyectados se les debe calcular la perspectiva por medio de la relación de triángulos. Viendo la Figura 5.6, fácilmente se puede deducir las siguientes ecuaciones:

$$PP=\left(\frac{Xp\cdot D\cdot \text{factor2D}}{Zp},\frac{Yp\cdot D\cdot \text{factor2D}}{Zp}\right) \tag{5.11}$$

Donde D es la distancia al plano de proyección. Y factor2D es el factor de ajuste para el plano de proyección. Cuando se transforman los puntos de 3D a 2D, el plano de proyección en la cámara oscura es muy pequeño. Por esto, es necesario el factor2D para obtener imágenes de tamaño coherente con los pixeles de la pantalla.

$$F=D\cdot \text{factor2D} \tag{5.12}$$

Para simplificar la fórmula, se multiplica D y factor2D para reemplazarlos por F.

$$PP=\left(\frac{Xp\cdot F}{Zp},\frac{Yp\cdot F}{Zp}\right) \tag{5.13}$$

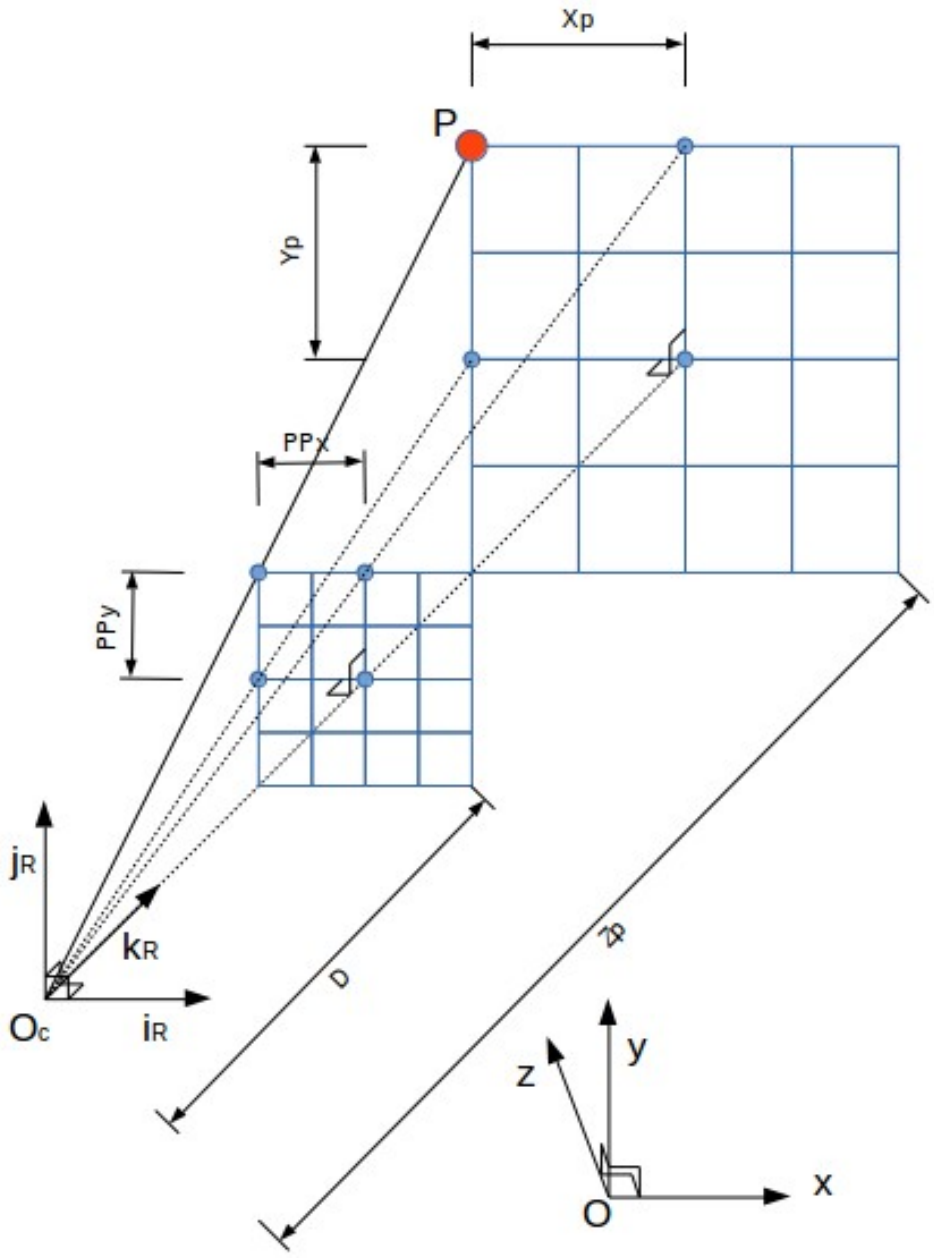


Figura 5.6: Relación de triángulos de los puntos proyectados

Y listo, ya se dedujo el campo vectorial de la visión. Con este campo vectorial, se puede hacer gráficos 3D por computadora (causas →

efectos) o también se puede inferir los parámetros de la cámara con el caracterizador evolutivo (efectos → causas).

5.4. Transformaciones e Isomorfismos entre Cuadriláteros

Para corregir la perspectiva de la imagen dentro de la página con puntos de referencia es necesario encontrar las transformaciones e isomorfismos entre cuadriláteros. Para esto, se caracteriza un cuadrilátero generalizado en el que sus vértices pueden cruzarse de forma no lineal. El objetivo es encontrar un mapeo que va desde el cuadrado unitario hacia el cuadrilátero generalizado. Ese mapeo corresponde a las coordenadas (t_i,t_j) en la figura siguiente:

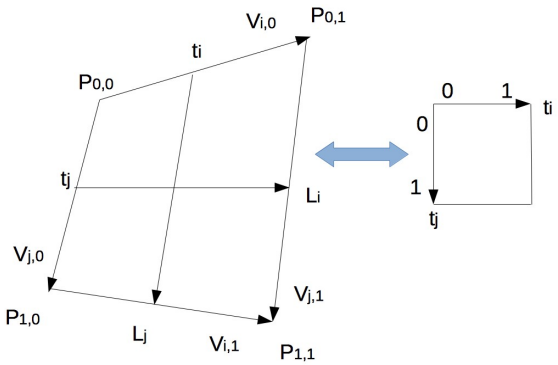


Figura 5.7: Cuadrilátero generalizado

Observando bien la figura anterior, se puede inferir las siguientes

ecuaciones. Pero primero hay que analizar la ecuación de la recta formada por 2 puntos:

$$L(t) = p_0 + (p_1 - p_0) \cdot t \tag{5.14}$$

Esta ecuación representa una recta con parámetro t. Cuando t=0, L(0)=p0. Y cuando t=1, L(1)=p1.

Si se aplica (5.14) a los puntos $P_{0,0}$ y $P_{1,0}$ con parámetro t_j, obtenemos la recta $V_{j0}(t_j)$.

$$V_{j0}(t_j) = P_{0,0} + (P_{1,0} - P_{0,0}) \cdot t_j \tag{5.15}$$

Si se aplica (5.14) a los puntos $P_{0,1}$ y $P_{1,1}$ con parámetro t_j, obtenemos la recta $V_{j1}(t_j)$.

$$V_{j1}(t_j) = P_{0,1} + (P_{1,1} - P_{0,1}) \cdot t_j \tag{5.16}$$

Si se aplica (5.14) a los puntos $V_{j0}(t_j)$ y $V_{j1}(t_j)$ con parámetro t_i, obtenemos la recta $L_i(t_i,t_j)$.

$$L_i(t_i,t_j)=V_{j0}(t_j)+[V_{j1}(t_j)-V_{j0}(t_j)]\cdot t_i \quad (5.17)$$

Si se reemplaza $V_{j0}(t_j)$ (5.15) y $V_{j1}(t_j)$ (5.16) en (5.17), se obtiene:

$$L_i(t_i,t_j)=P_{0,0}+(P_{1,0}-P_{0,0})\cdot t_j+[P_{0,1}+(P_{1,1}-P_{0,1})\cdot t_j-(P_{0,0}+(P_{1,0}-P_{0,0})\cdot t_j)]\cdot t_i \quad (5.18)$$

Factorizando ti y tj en (5.18), se obtiene el mapeo desde el cuadrado unitario hacia el cuadrilátero generalizado:

$$L_i(t_i,t_j)=P_{0,0}+(P_{1,0}-P_{0,0})\cdot t_j+(P_{0,1}-P_{0,0})\cdot t_i+(P_{1,1}-P_{0,1}-P_{1,0}+P_{0,0})\cdot t_i\cdot t_j \quad (5.19)$$

Para hacer el mapeo inverso, desde el cuadrilátero generalizado hacia el cuadrado unitario, primero hay que simplificar un poco (5.19). P4, VX, VY y LO son partes de (5.19):

$$P4=P_{1,1}-P_{1,0}-P_{0,1}+P_{0,0} \quad (5.20)$$

$$VX=P_{0,1}-P_{0,0} \quad (5.21)$$

$$VY=P_{1,0}-P_{0,0} \quad (5.22)$$

$$LO=L_i(t_i,t_j)-P_{0,0} \quad (5.23)$$

De este modo, (5.19) queda bastante simplificada para poder calcular su inversa más fácilmente:

$$LO = VX \cdot t_i + VY \cdot t_j + P4 \cdot t_i \cdot t_j \tag{5.24}$$

Si se transforma (5.24) de su forma vectorial a las ecuaciones de sus componentes (x,y), se obtiene el siguiente sistema de ecuaciones:

$$\left.\begin{array}{l} LO_x = VX_x \cdot t_i + VY_x \cdot t_j + P4_x \cdot t_i \cdot t_j \\ LO_y = VX_y \cdot t_i + VY_y \cdot t_j + P4_y \cdot t_i \cdot t_j \end{array}\right\} \tag{5.25}$$

Despejando t_j en ambas ecuaciones se obtiene:

$$t_j = \frac{LO_y - VX_y \cdot t_i}{VY_y + P4_y \cdot t_i} = \frac{LO_x - VX_x \cdot t_i}{VY_x + P4_x \cdot t_i} \tag{5.26}$$

Después de muchas operaciones algebraicas, en resumen se obtiene:

$$(VX \times P4) \cdot t_i^2 + (VX \times VY - LO \times P4) \cdot t_i - LO \times VY = 0 \tag{5.27}$$

Entonces t_i está expresado en términos de una ecuación cuadrática que se resuelve con (5.29). Y t_j está expresado en términos de t_i. Hay

que recordar que los valores válidos para (t_i,t_j) deben ser números reales (no números imaginarios) entre 0 y 1. Con ese dato se descartan varias posibilidades.

$$a \cdot x^2 + b \cdot x + c = 0 \tag{5.28}$$

$$x = \frac{-b \pm \sqrt{b^2 - 4 \cdot a \cdot c}}{2 \cdot a} \tag{5.29}$$

Una vez que se tiene los mapeos desde el cuadrado unitario hacia el cuadrilátero generalizado y viceversa, al componer ambos mapeos se puede mapear desde cualquier cuadrilátero generalizado hacia cualquier otro cuadrilátero generalizado.

5.5. Uso y Demostración de la Aplicación

Primero que nada, es necesario imprimir la hoja con los puntos de referencia. Para esto hay que abrir una ventana de terminal y entrar al directorio de la tesis. Para luego ejecutar el script page-of-numbers.sh y de esta forma se puede capturar e imprimir esta imagen con los puntos de referencia:

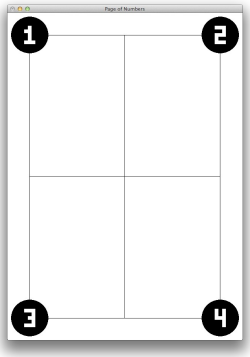


Figura 5.8: Aplicación mostrando los puntos de referencia

Hay que asegurarse de que la impresión de la imagen ocupe toda la hoja. Hay opciones de impresión para lograrlo. Una vez impresa la página, hay que medir las dimensiones del rectángulo exterior de la hoja. Es decir, hay que medir las distancias entre las líneas paralelas para luego tipearlas cuando el programa las pida.

Las dimensiones por defecto son 15.4 cm y 23.3 cm. Se puede escribir la distancia en centímetros, pulgadas o incluso metros. El sistema es lo suficientemente flexible para adaptarse a estas unidades y los parámetros de la cámara serán inferidos con estas mismas unidades. Y asimismo el error de los puntos proyectados

estará expresado con el cuadrado de estas mismas unidades.

Cabe recalcar que el error de los puntos proyectados es aproximadamente 1 ó 2 centímetros. Este error se debe a la rareza de los sistemas reales. Uno podría pensar que la rectitud de las trayectorias de la luz iban a crear modelos platónicos perfectos de la visión pero no es así. Principalmente porque los lentes de la cámara no hacen correspondencias perfectas con el modelo matemático basado en una “pinhole camera” sin lentes. Pero también se debe a las imperfecciones y a la naturaleza probabilística y parcialmente observable de los sistemas raros. Sin embargo, el error de medición de los parámetros inferidos es de 5 mm como se verá más adelante.

Luego en la ventana de terminal se ejecuta el script vision-app-for-<OS>.sh. Donde <OS> es el sistema operativo. Hay soporte para 2 sistemas operativos: Mac y Ubuntu.

El sistema de visión tiene 2 modos o fases de operación: Calibración e inferencia. Estas opciones se eligen al inicio del programa. Al principio de las 2 opciones siempre se pide las dimensiones del rectángulo exterior que fueron medidas anteriormente.

En la opción de calibración, luego el programa pide la altura donde está la cámara, independientemente e invariantemente de la posición de la cámara en el plano. Es necesario medir la altura lo más precisamente posible para que el sistema calibre bien el factor de conversión 3D/2D de la cámara. Al final del proceso, es necesario anotar en algún lado el factor de conversión calculado porque este será pedido en la siguiente fase de inferencia. En el campo vectorial de la visión, la altura permanece constante en la fase de calibración para poder inferir correctamente los otros parámetros. Es necesario decirle al programa cuál es la altura de la cámara porque los parámetros de la cámara pueden ser ambiguos. Un ejemplo de esto es la película antigua de Godzilla en la cual los efectos especiales se hacían con muñecos de Godzilla y maquetas de ciudades. El que ve la película no puede calcular las dimensiones de Godzilla y de las ciudades. Parecería que son muy grandes pero en realidad no lo son. El mismo tipo de ambigüedades ocurre aquí en el sistema de visión. Por eso es necesario decirle al sistema con qué dimensiones se está trabajando. Debido a las propiedades de lineales del campo vectorial de visión, no es posible resolver esta ambigüedad de otra forma.

En la opción de inferencia, luego el programa pide el factor de conversión 3D/2D de la cámara que fue calculado en la fase de calibración. Este factor de conversión permanece constante en el campo vectorial de la visión para poder inferir los otros parámetros de la cámara.

Después de configurar bien el sistema, debería aparecer una ventana como esta:

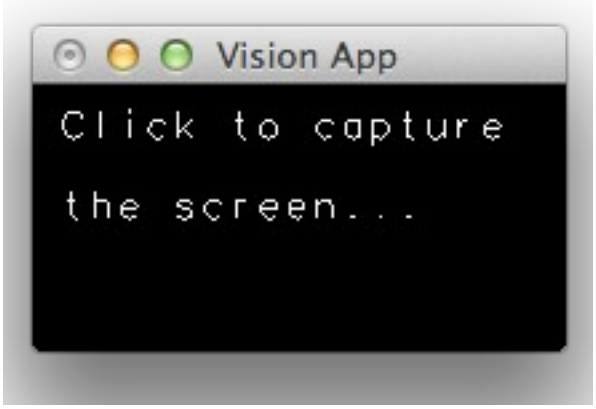


Figura 5.9: Pantalla inicial de la Vision App

Una vez que aparece esta imagen, es necesario cargar una aplicación como QuickTime para Mac (Opción: File → New Movie Recording) o Cheese Webcam Booth para Ubuntu que muestre lo que la webcam está filmando. O si se tiene una imagen pre-capturada,

mostrarla en algún programa que muestre imágenes. Y luego se llama a la Vision App y se hace click donde dice “Click to capture the screen”.

Figura 5.10: Vision App a punto de capturar la pantalla

Vale la pena hacer énfasis en que el zoom de la ventana que muestra lo que filma la cámara debe ser el mismo que el zoom en el cual se calculó el factor de conversión 3D/2D de la cámara. Si no es así, el proceso de inferencia generará datos erróneos. Siempre que se utilice cámaras distintas o zooms de ventana distintos, es necesario volver a calcular el factor de conversión 3D/2D de la cámara.

Una vez capturada la pantalla, la Vision App se maximizará para ocupar casi toda la pantalla y mostrará la imagen capturada en blanco

y negro como en la Figura 5.11. En este estado, la Vision App le pedirá al usuario que indique donde están los puntos 1, 2, 3 y 4, respetando ese orden. Se puede agrandar el círculo de búsqueda con la tecla + y achicarlo con la tecla -. Se marca un punto de referencia con un click izquierdo y se borra el último punto de referencia marcado con la tecla Delete. Al final del proceso, cuando ya se han marcado los 4 puntos de referencia de manera correcta, se presiona la tecla Enter para continuar con el siguiente proceso.

Figura 5.11: Selección de puntos de referencia en la imagen capturada

Una vez marcados los puntos, estos comienzan a adaptarse perfectamente a la forma de las elipses escaladas y rotadas, con el fin de minimizar los errores. Esta evolución puede finalizar

automáticamente para pasar al siguiente proceso. Pero si se demora mucho, se presiona la tecla Enter para continuar.

Luego la Vision App comienza a evolucionar los parámetros de la cámara hasta que finalmente llega a una solución. La imagen central está rectificada con las fórmulas de mapeos entre cuadriláteros y es vista desde arriba de esta con la fórmula del campo vectorial de la visión 3D. La imagen del fondo indica cuando los parámetros de la cámara fueron inferidos correctamente en el momento en que los puntos negros del cuadrilátero corresponden con las elipses proyectadas.

Una línea roja une la posición inferida de la cámara y el centro de la imagen y va en dirección del vector k de la base ortonormal inferida de la cámara. Las direcciones de los vectores i e j también son mostradas con 2 pequeñas líneas de color verde y azul, respectivamente. Es importante ver la correspondencia de ejes de colores (verde y azul) entre la foto bidimensional y la base ortonormal de la cámara. Esto ayuda a darse cuenta cuánto ha rotado y se ha desplazado la cámara.

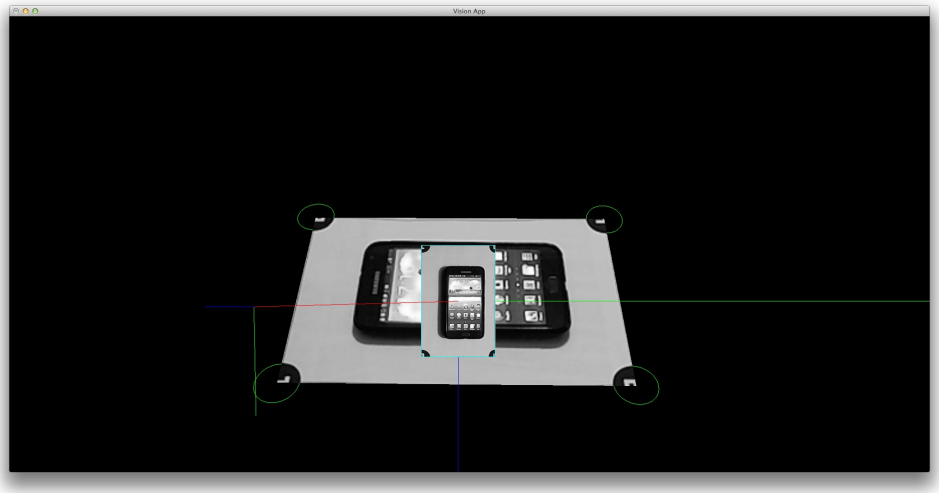


Figura 5.12: La evolución de los parámetros de la cámara

Vale la pena mencionar que los puntos proyectados corresponden a los vectores de salida del campo vectorial de la visión. Y los puntos 3D corresponden a los vectores de entrada:

$$(-0.5\cdot Ancho, 0, -0.5\cdot Alto) \tag{5.30}$$

$$(0.5\cdot Ancho, 0, -0.5\cdot Alto) \tag{5.31}$$

$$(-0.5\cdot Ancho, 0, 0.5\cdot Alto) \tag{5.32}$$

$$(0.5\cdot Ancho, 0, 0.5\cdot Alto) \tag{5.33}$$

Como trabajo pendiente a futuro queda la detección automática de

puntos de referencia por medio de una exploración exhaustiva de toda la pantalla a manera de lluvia de detectores de patrones. Y también queda pendiente la detección automática de los números con redes neuronales profundas [37].

5.6. Estadísticas y Mediciones del Sistema Básico de Visión Artificial

Para probar la precisión del sistema básico de visión artificial, se tomó 24 mediciones correspondientes a las combinaciones de las 4 direcciones de la hoja con puntos de referencia (norte, sur, este y oeste), los 3 offsets en el plano horizontal (30 cm, 50 cm y 70 cm) y las 2 alturas (30 cm y 60 cm). A los 3 offsets del plano horizontal se le restó 2.2 cm de la muesca de la webcam. Para entender mejor como se tomó las mediciones, es necesario observar esta figura siguiente:

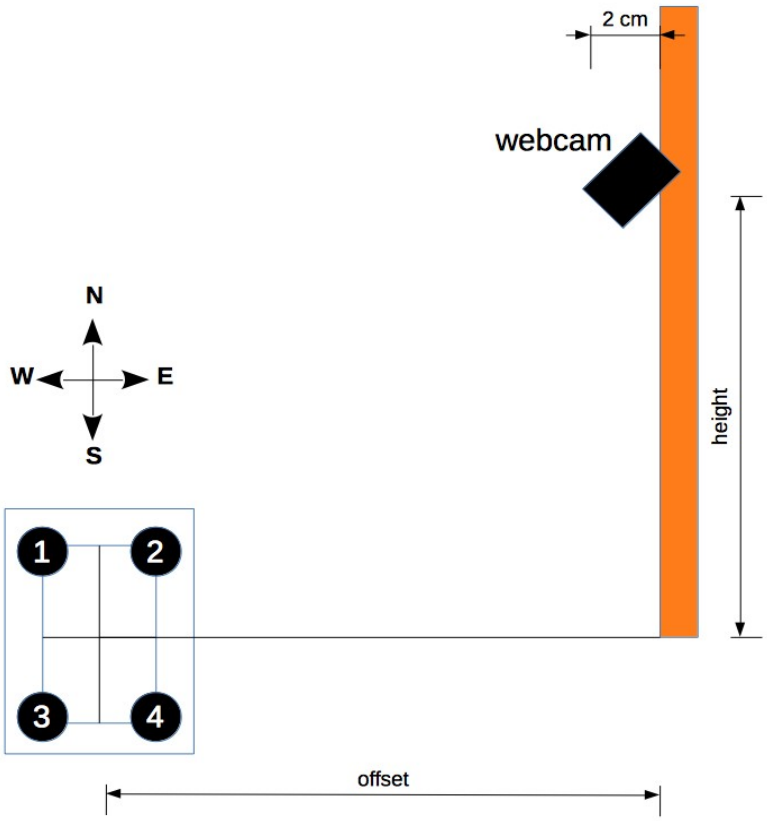


Figura 5.13: Mediciones del Sistema Básico de Visión Artificial

Antes de tomar las mediciones, se tuvo que calibrar el factor de conversión 3D/2D de la webcam. Dado que este proceso toma mucho tiempo, sólo se utilizo 3 mediciones:

Tabla 7: Estadísticas para calibrar el factor de conversión 3D/2D de la webcam

Iteraciones:	Tiempo: [segundos]	Error cuadrático:	Altura: [cm]	Factor de cámara:	Dirección:	Desplazamiento:		
						Valor esperado: [cm]	Medición: [cm]	Error: [cm]
100000	2015.4832	0.9826	60	1277.1183	Este	28	28.0198	0.0198
100000	2010.8421	1.6909	60	1288.3211	Este	48	47.3566	0.6433
100000	2021.6348	5.1731	60	1290.9700	Este	68	68.0391	0.0391

De los 3 cálculos, el promedio del factor de la cámara es 1285.4698. Este valor se utiliza para el proceso de inferencia.

Luego se calcula la posición de la cámara como se describió en la figura anterior. A los valores esperados de la cámara se le resta 2.2 cm correspondientes a la muesca de la cámara. Es decir, la protuberancia que tiene la cámara por delante de la medición esperada.

Las estadísticas son las siguientes:

Tabla 8: Estadísticas de los cálculos de las distancias de la webcam

Iteraciones	Tiempo [segundos]	Error cuadrático	Factor de cámara	Dirección	Desplazamiento			Altura		
					Valor esperado [cm]	Medición [cm]	Error [cm]	Valor esperada [cm]	Medición [cm]	Error [cm]
1958	39.1499	204.0601	1285.4698	Este	27.8	27.8009	0.0009	30	30.6527	0.6527
2770	55.1554	38.5474	1285.4698	Este	47.8	47.7598	0.0401	30	29.6811	0.3188
4047	90.2000	3.0226	1285.4698	Este	67.8	68.4744	0.6744	30	30.4828	0.4828
10846	224.0536	2.2205	1285.4698	Este	27.8	27.2033	0.5966	60	61.2113	1.2113
6024	124.9674	1.8377	1285.4698	Este	47.8	46.8416	0.9583	60	60.2994	0.2994
6692	133.7225	5.0329	1285.4698	Este	67.8	67.3712	0.4287	60	60.4175	0.4175
3220	64.3977	33.6266	1285.4698	Norte	-27.8	-28.0765	0.2765	30	29.9949	0.0050
3440	68.4950	9.2270	1285.4698	Norte	-47.8	-48.3357	0.5357	30	29.7187	0.2812
10748	215.5826	8.3584	1285.4698	Norte	-67.8	-67.1638	0.6361	30	29.2844	0.7155
14339	286.6637	9.9277	1285.4698	Norte	-27.8	-27.7006	0.0993	60	59.3759	0.6240
17360	350.0106	3.6259	1285.4698	Norte	-47.8	-48.5414	0.7414	60	59.8273	0.1726
14544	293.1470	0.4360	1285.4698	Norte	-67.8	-67.5554	0.2445	60	59.7247	0.2752
1571	31.6026	72.2548	1285.4698	Sur	27.8	28.3951	0.5951	30	29.5943	0.4056
1669	33.3013	20.8452	1285.4698	Sur	47.8	48.2883	0.4883	30	30.5630	0.5630
3348	66.7890	4.9322	1285.4698	Sur	67.8	68.3086	0.5086	30	29.7976	0.2023
9365	189.3054	0.6006	1285.4698	Sur	27.8	28.3256	0.5256	60	59.1956	0.8043
4755	95.3014	9.6260	1285.4698	Sur	47.8	47.8732	0.0732	60	58.1753	1.8246
4393	87.8515	2.2746	1285.4698	Sur	67.8	68.2217	0.4217	60	60.0830	0.0830
2344	47.0125	60.4118	1285.4698	Oeste	-27.8	-27.8883	0.0883	30	29.6982	0.3017
2865	58.7811	17.1971	1285.4698	Oeste	-47.8	-46.9789	0.8210	30	30.1520	0.1520
4863	100.2109	12.8456	1285.4698	Oeste	-67.8	-66.8387	0.9612	30	29.3582	0.6417
12651	254.4063	10.6184	1285.4698	Oeste	-27.8	-27.7996	0.0003	60	59.3799	0.6200
11137	225.1244	0.4220644024	1285.4698	Oeste	-47.8	-47.3683	0.4316	60	59.3393	0.6606
6067	121.9956	2.4085655473	1285.4698	Oeste	-67.8	-66.6308	1.1691	60	59.6182	0.3817

Con los datos estadísticos de la tabla anterior, se calcula el error promedio y su desviación estándar del desplazamiento y de la altura:

Tabla 9: Errores promedio en el cálculo de distancias

Desplazamiento [cm]		Altura [cm]	
Error promedio	**Desviación estándar**	**Error promedio**	**Desviación estándar**
0.4715798397	0.3254029717	0.5040757275	0.3889052403

Es decir, los errores promedios del desplazamiento y de la altura son aproximadamente 5 mm. Lo cual indica que el sistema de visión artificial es muy preciso. El ojo humano no es capaz de estimar distancias con semejante precisión. Es muy probable que el error sea causado principalmente por la imprecisión de las mediciones manuales y rústicas y por el temblor de las manos.

La Figura 5.14 es un experimento de percepción visual en el que uno se da cuenta lo importante que es tener 2 ojos. Con 2 ojos es muy fácil hacer tocar los 2 borradores de los lápices. Mientras que con 1 ojo es difícil y uno casi siempre comete errores de percepción de la profundidad. Esto demuestra claramente que la visión humana necesita estereopsis para percibir la profundidad y además estima distancias en base a relaciones entre tamaños conocidos, no estima con centímetros. En cambio, el sistema básico de visión artificial de esta tesis no necesita de estereopsis para calcular profundidades y distancias con precisión milimétrica.

Figura 5.14: Con 1 ojo es difícil calcular la profundidad para hacer que los borradores se toquen [40]

CAPÍTULO 6

6. ALGORITMO: RUTEADOR CAUSAL JERÁRQUICO

6.1. Introducción

El ruteador causal jerárquico sirve para que la máquina explore espacios raros [19], aprenda patrones causales a través del ciclo de percepción-acción [23], auto-organice el conocimiento aprendido [5] y cree soluciones y comportamientos emergentes en tiempo real [3]. Esta red markoviana [15] fue inspirada por la generación de patrones neuro-motrices [14] que son el producto de la adaptación de los

mapas cerebrales [41] al tratar de imitar las geometrías diferenciales complejas [42] generadas por la mecánica de Lagrange [43]. Pero en teoría, si se modifica adecuadamente esta red (por ejemplo: representar patrones más generalmente y combinarla con otras teorías de la computación cognitiva) [11,37], esta red tendría el potencial de convertirse en un sistema de inteligencia artificial general (AGI) [12,44]. La red es básicamente un proceso de decisión markoviano jerárquico (Hierarchical MDP) con programación dinámica bottom-up [10,15,45]. Debido a sus similitudes con el cerebro, se puede decir que la red es un inconsciente artificial capaz de aprender y generar reflejos artificiales [46]. Es un intento para darle creatividad e intuición artificial a las máquinas [3]. Afirmar que los MDPs son modelos del subconsciente no es tan especulativo como parece. De hecho, Ray Kurzweil está creando en Google simulaciones de la mente con Hierarchical Hidden Markov Models (HHMM) que son la contraparte perceptiva de los Partially Observable Semi-Markov Decision Processes (POSMDP) [47,48]. Además, neurocientíficos computacionales como Rajesh Rao y Edmund Rolls usan MDPs en sus modelos neuronales [31,46].

En el mundo real, hay sistemas complejos, altamente

multidimensionales y multicausales, causalmente enredados, no-continuos, rugosos, difusos, llenos de detalles sutiles, casi impredecibles, muy caóticos, sin simetría y probabilísticos. A estos sistemas, la hija de Richard Bellman, Dr. Kirstie Bellman, les llamó “sistemas raros” o “bizarre systems” [19].

6.2. Explicación Matemática del Algoritmo

Cuando uno lee libros sobre la psicología de la creatividad, se da cuenta que la creatividad es un largo proceso y un individuo tarda aproximadamente 10 años para volverse un experto en un dominio determinado al punto de encontrar soluciones creativas a problemas complejos. Este período puede variar dependiendo de qué tan difícil sea el campo de estudio, de qué tan apasionado e inteligente sea el individuo y de cuántas facilidades tenga para experimentar y aprender [49].

A diferencia de lo que muchos creen, la creatividad es un proceso de aprendizaje no-supervisado [3]. Si bien es cierto que la educación supervisada ayuda mucho a acortar el tiempo del proceso creativo, la creación de nuevo conocimiento no es un proceso de aprendizaje

supervisado. La primera vez que se inventa algo o se descubre algo, se lo hace con las técnicas de aprendizaje no-supervisado porque es un territorio mental no explorado y el cerebro debe adaptar sus redes cognitivas para representar dicho territorio mental de la mejor forma posible [41]. Cuando uno tiene preconcepciones y prejuicios acerca de un área del conocimiento, por lo general aprende mal y el cerebro trata de hallar las pequeñas piezas de evidencia que apoyen sus preconcepciones y a la vez ignorar las grandes piezas de evidencia que contradicen sus preconcepciones. Esto no es totalmente malo porque puede ayudar a simplificar nuestros modelos mentales. Los prejuicios cognitivos no son 100% dañinos porque tienen ciertas ventajas evolutivas. El cerebro no es perfecto, sólo es lo suficientemente bueno como para sobrevivir con dignidad.

El método científico es un proceso creativo mucho más perfeccionado aunque no es 100% perfecto y es mucho más lento que el sentido común debido a su mayor grado de perfección. El método científico ha sido ampliamente estudiado y puede ser de gran ayuda para entender como funciona el proceso creativo [50].

En breves palabras, la creatividad consiste en aprender una gran

cantidad de patrones y causalidades que se agrupan por similitud y se auto-organizan en base a correspondencias analógicas para formar subconscientemente redes cognitivas que representan los fenómenos que ocurren en el entorno [51]. Estas redes cognitivas de patrones complejos que fueron aprendidas en el pasado, nos ayudan a entender el presente y también nos ayudan a predecir el futuro que es totalmente desconocido y también predicen las partes desconocidas del pasado y del presente [23]. Por ejemplo: Una teoría científica nos ayuda a entender el mundo. Un invento es un mecanismo causal complejo que ayuda a resolver un problema [52]. Una obra de arte es un patrón complejo que busca causar una percepción en el observador. La inteligencia artificial es la máxima expresión de la creatividad porque esta es una teoría científica, es un invento tecnológico y también es un arte [48].

Si se estudia profundamente las obras del genio matemático Richard Bellman quien inventó la programación dinámica, el aprendizaje por refuerzo y sus famosas ecuaciones, uno se da cuenta que los procesos de decisión markovianos (MDP o Markov Decision Processes) encajan casi perfectamente con la teoría de los procesos creativos descrita anteriormente y también son compatibles con la

teoría de los cognits del Dr. Joaquín Fuster, neurocientífico cognitivo [10,23,47]. Sólo es necesario agregar unos cuántos principios de la inteligencia a los POSMDP (Partially-Observable Semi-Markov Decision Processes) para crear un sistema de inteligencia artificial general. Pero esta afirmación es algo especulativa y necesita ser comprobaba con muchísimos experimentos de programación.

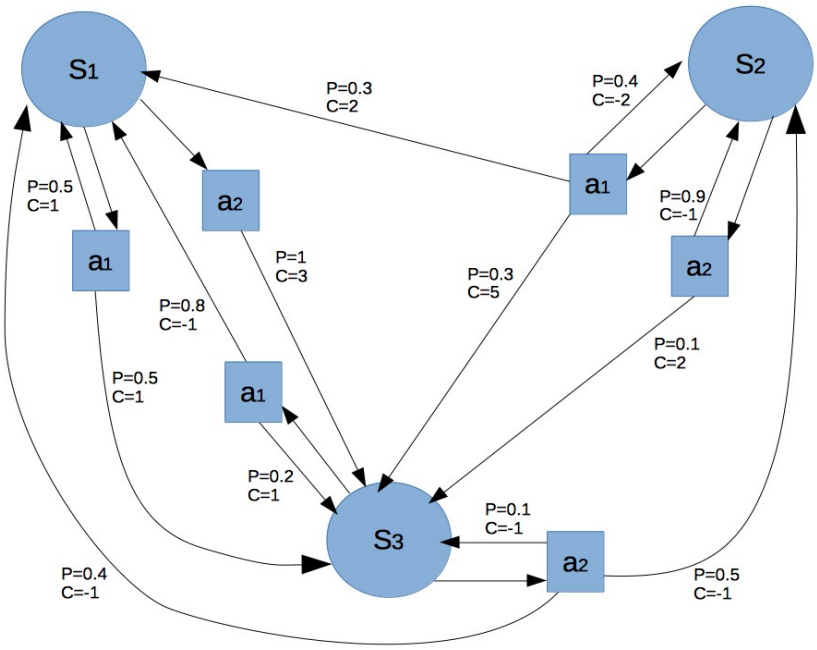


Figura 6.1: Proceso de decisión Markoviano con 3 estados y 2 acciones

En la figura anterior, hay 3 estados S1, S2 y S3. Y hay 2 acciones por estado a1 y a2. Cada estado se conecta con otros estados o consigo mismo a través de las acciones que tienen un resultado causal

probabilístico. Como toda probabilidad, la suma de todas las probabilidades asociadas a una acción provenientes de un estado en particular es uno, sin importar cuántos resultados o consecuencias existan. Además de las probabilidades asociadas a cada acción proveniente de un estado, hay un costo asociado que puede ser cualquier valor real. El costo puede ser pequeño, grande o incluso infinito. También puede ser positivo (costo), negativo (recompensa) o cero (indiferente) [10,47].

Los modelos de transición pueden ser pre-programados o aprendidos (learning automaton) [45,53]. El aprendizaje de los modelos de transición puede ocurrir dentro de la vida del organismo (ontogenia) o ser común a la especie que evoluciona (filogenia) [54]. Las memorias filogenéticas de la especie son heurísticas abstractas y comunes para toda la especie. Son los instintos y sentimientos que están localizados principalmente en las regiones subcorticales del cerebro o el cerebro primitivo que es común para una gran cantidad de especies [55,56,57]. En cambio, el neocórtex es un invento relativamente reciente de la evolución. Los mamíferos tienen un gran neocórtex. Y la corteza prefrontal es un invento aún más reciente de la evolución que nos permite aprender, crear y predecir [57]. Hacia el interior, la

dirección subcortical, las redes neuronales son más reactivas, instintivas, invariantes, sentimentales y abstraen los aspectos de la cognición comunes a la especie [54,57]. Hacia el exterior, la dirección cortical, las redes neuronales son más proactivas, aprendidas, flexibles, neuroplásticas, racionales y abstraen las experiencias de la vida del individuo [57]. Los grandes primates (gorilas, chimpancés, orangutanes, bonobos y humanos) tienen una gran corteza prefrontal [50]. Y los primates humanos tienen una corteza prefrontal aún más sofisticada que nos hace muy inteligentes en comparación con los primates no-humanos [58]. Nuestra inteligencia superior no sólo se debe a la corteza prefrontal sino también a nuestro sofisticado patrón de conectividad canónico de las columnas corticales y a nuestro sofisticado conectoma [59]. Pero ese es un tema que transciende esta tesis.

Los estados de las redes cognitivas pueden representar cualquier cosa. Desde algo muy simple hasta algo muy complejo. Algo continuo o discreto. Algo con relaciones jerárquicas o sin jerarquías de complejidad. Algo preciso o flexible. Etcétera. Pero lo que es común a todas estas representaciones es que los estados del sistema nunca representan exactamente los mismos estados observados. Incluso si

se abstrae de tal forma que 2 estados observados por el sistema son exactamente iguales, siempre tendrán de diferencia el tiempo en el cual fueron observados. Siempre hay algo que los diferencia. Pero para ahorrar recursos computacionales, los estados similares se agrupan con algún algoritmo de similitud en clusters, bins o algún tipo de categorización. Entonces las transiciones entre estados tienen un componente probabilístico dado que los estados nunca son 100% iguales y no se garantiza la aplicabilidad de las leyes deterministas del entorno, que por cierto son puramente platónicas. Las similitudes entre los estados observados que son categorizados en uno o varios de los estados del sistema son puramente analógicas como la teoría de probabilidades. Las probabilidades se basan en analogías [60].

El estudio de cómo el cerebro hace analogías por medio de la categorización es un tema candente en la neurociencia cognitiva. Douglas Hofstadter es un pionero de esta área y recientemente escribió un libro muy interesante sobre el tema: “Surfaces and Essences: Analogy as the Fuel and Fire of Thinking”. El mecanismo exacto por el cual el cerebro hace analogías todavía es desconocido. Pero existen muchos modelos simples que imitan esta función cognitiva de manera interesante, aunque aún incompleta [28].

Dependiendo del tipo de representación de analogías en los cerebros artificiales o en los cerebros biológicos, varía mucho el tipo de inteligencia. El software actual es impresionantemente inflexible a causa de que no puede establecer correspondencias analógicas entre patrones similares. La realidad es extremadamente compleja y caótica. Es muy difícil o quizás imposible encontrar 2 entidades matemáticamente iguales en la realidad. Es por esto que las matemáticas con las cuales los computadores actuales funcionan son puramente platónicas y no pueden capturar eficazmente la complejidad del mundo real. Uno puede programar un sistema que realice un montón de cálculos exactos de una manera extremadamente rápida y precisa porque las matemáticas que gobiernan esos modelos son totalmente precisas e inflexibles.

Pero que tal si se presenta el caso en el que un robot tiene que abrir una puerta para resolver un problema. Primero que nada, no existen 2 puertas 100% iguales en el mundo. Entonces es necesario categorizar el concepto de puertas de una manera flexible e imprecisa. Una vez creadas las redes conceptuales que representan las puertas, el robot puede predecir y extrapolar comportamientos y

propiedades similares entre puertas similares. En cambio, de la forma en que los sistemas actuales son pre-programados, el robot vería la puerta y al no ser exactamente igual a la puerta que le programaron, el robot la descartaría como si fuera un objeto diferente con diferentes propiedades. Es decir, el robot cometería un error tipo 2 o falso negativo. Los falsos negativos son categorizaciones que son erróneamente rechazadas por el proceso de inferencia. En este caso, el robot podría tener la capacidad para entender y actuar sobre la puerta pero no logra conectar las sensaciones visuales con las redes conceptuales relacionadas con la puerta. Esto es sólo un ejemplo entre los billones de aplicaciones que las analogías podrían resolver.

Un compromiso de diseño (engineering tradeoff) que los sistemas analógicos traen es que si se evita mucho los errores del tipo 2 o falsos negativos, el sistema tiene la tendencia a cometer muchos errores del tipo 1 o falsos positivos. Es decir, los sistemas perceptuales ven cosas o establecen conexiones donde en realidad no las hay (pareidolias). Evolutivamente hablando, los errores de tipo 1 son inofensivos, mientras que los errores de tipo 2 pueden ser letales. Por ejemplo, si un animal ve un patrón con forma de ojos en la selva, piensa que es un depredador y se asusta, no hay problema

porque sólo fue un susto. Mientras que si un animal ve unos ojos en la selva y no los asocia con un depredador, el depredador puede potencialmente comer al animal. Hay muchos ejemplos similares. De esta forma, la selección natural moldeó nuestros cerebros para que sean flexibles e imperfectos: Nuestros cerebros son muy susceptibles a falsos positivos y raramente descartan falsos negativos. Nuestros cerebros tienen la capacidad de extrapolar y predecir propiedades y comportamientos de objetos desconocidos basándose en similitudes con experiencias pasadas. Esto hace que nuestros cerebros parezcan hiper-completos y expertos a pesar de haber explorado sólo una pequeña porción de nuestro mundo. Esto es un pre-requisito para crear mentes artificiales creativas, flexibles, extrapolares y llenas de recursos mentales. Es por esto que nuestros cerebros pueden sobrevivir decentemente en este mundo complejo y nuestros computadores simplemente no pueden entender la complejidad de nuestro mundo.

Para crear es necesario tener una mente abierta y flexible que sea capaz de examinar una gran cantidad de experiencias y de establecer asociaciones entre los conocimientos aprendidos. Uno no nace ni con las experiencias ni con las asociaciones. Estas se aprenden a lo largo

de la vida. Pero hay que reconocer que uno nace con instintos comunes a la especie. Decidir qué experiencias y qué asociaciones son relevantes e importantes de recordar es un algoritmo de búsqueda de patrones sobresalientes. Hay 2 estrategias extremas relacionadas con este algoritmo. La primera estrategia extrema es tratar de recordar cualquier cosa sin juicio crítico, lo cual hace perder tiempo y recursos mentales valiosos. Es un error de tipo 1 en el que se recuerda falsos positivos. En este caso la hipótesis es decidir que patrón es sobresaliente o no. La segunda estrategia extrema es ser demasiado crítico al aprender experiencias y al establecer asociaciones. Este es un error de tipo 2 en el que se podría descartar conocimientos y asociaciones sobresalientes. Una mente demasiado crítica le quita alas a su creatividad porque a veces es necesario cometer errores. Sobretodo en campos del conocimiento inexplorados en el que la inteligencia y la memoria no ayudan mucho en las fases iniciales del aprendizaje. En el aprendizaje no-supervisado, es inevitable cometer errores y uno no debería tomarlo como algo malo. Los errores más bien son parte del territorio. Para conocer bien el territorio, es necesario explorarlo exhaustivamente incluyendo los errores. Conocer bien los errores ayuda a cortar drásticamente el espacio de patrones como en el ejemplo del buscaminas [35].

Albert Einstein dijo algo sobre qué tan inevitable es cometer errores en el aprendizaje: “Una persona que nunca ha cometido un error, nunca ha intentado aprender algo nuevo.” Thomas Edison dijo que cometer errores es sólo una forma de conocer cómo funcionan las cosas: “No he fallado. Sólo he encontrado 10.000 formas que no funcionan.”

Otro punto importante de las analogías es que estas no deberían ser pre-programadas en el sistema, sino más bien deberían ser aprendidas por el sistema, de preferencia con un algoritmo de aprendizaje no-supervisado que adapte de la mejor manera posible sus redes cognitivas y represente el entorno sin prejuicios ni preconcepciones. El objetivo máximo del campo de la inteligencia artificial general, que es crear una máquina con inteligencia similar o mejor que los humanos, parecería imposible cuando se asume que las todas las expresiones de inteligencia son pre-programadas por ingenieros. En realidad, esta tarea es imposible de esa forma porque todas las expresiones de inteligencia son más numerosas que todos los átomos del universo, debido a las explosiones combinatoriales. Pre-programar funciones cognitivas es un camino errado y sin salida.

El método alternativo se llama machine learning o aprendizaje automático que consiste en programar algoritmos capaces de aprender y crear representaciones muy flexibles del entorno. De esta forma, las funciones cognitivas emergen de los cerebros biológicos y pueden emerger de los cerebros artificiales, a través del aprendizaje y la auto-organización de las experiencias [7, 53].

Los patrones son estructuras espacio-temporales de materia/energía que forman objetos reales o representan abstracciones de objetos reales. Los patrones de alto nivel son necesariamente abstracciones de objetos reales o relaciones entre patrones. No pueden ser objetos reales. Un patrón simple puede ser parte de un patrón complejo que contiene estructuras formadas por patrones simples. Los patrones no sólo representan cosas concretas sino también relaciones abstractas, asociaciones, analogías, transformaciones, etc. [41]

Tanto los estados del sistema como sus posibles acciones pueden ser representados por patrones. Hay muchísimas formas de representar patrones y muchísimos algoritmos que lidian con cada tipo de patrón. Para esta tesis, los patrones de estados se representan con lógica difusa que mide el grado de pertenencia de las mediciones de las

variables implicadas con respecto a unos bins o contenedores multidimensionales o de varias variables asociadas. La función de pertenencia sería el mínimo de las funciones triangulares de pertenencia para cada variable del sistema, lo cual genera hiper-pirámides cuya cúspide coincide con el centro del contenedor multidimensional como en la Figura 6.2. Ese es el borrosificador. El desborrosificador sería el promedio ponderado de las acciones activadas. Existe fuerte evidencia neuro-fisiológica que en el cerebro existen borrosificadores y desborrosificadores. En el curso online “The Brain and Space” en Coursera, la profesora Jennifer Groh explicó los conceptos de maps y meters que corresponden a los borrosificadores y desborrosificadores, respectivamente [61,62].

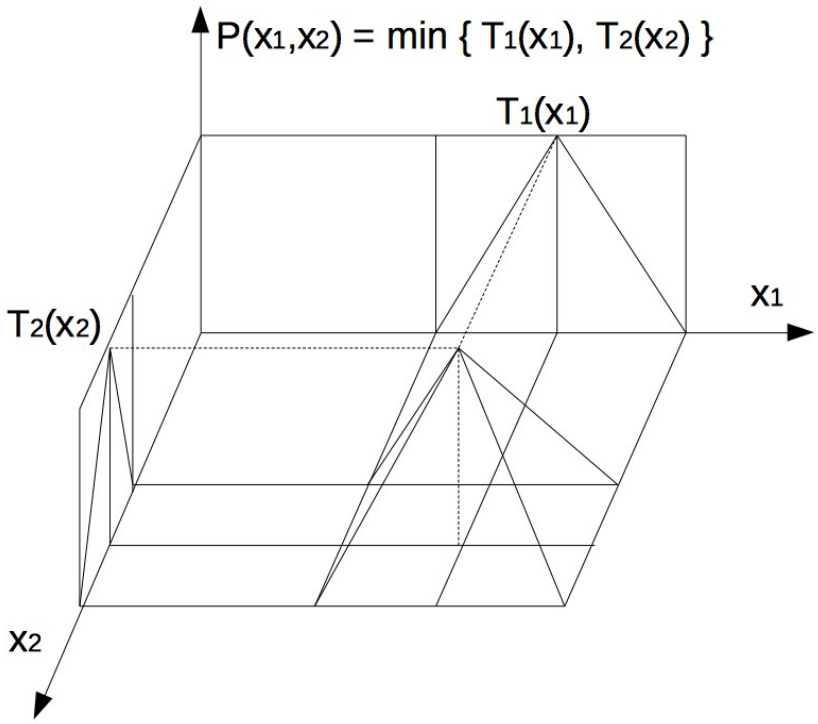


Figura 6.2: Borrosificador de hiper-pirámides basado en el mínimo de funciones de pertenencia triangulares

La auto-organización de la red se haría con el algoritmo de reinforcement learning estableciendo correspondencias analógicas entre los contenedores multidimensionales del sistema que representan los estados y las acciones. En este caso, la analogía se hace a nivel de estados observados vecinos y se asume por continuidad del hiper-espacio causal que los estados vecinos tienen comportamientos y propiedades similares [63].

Una manera muy intuitiva, simple y a la vez efectiva para modelar la

neuroplasticidad de las neuronas, es usar correlaciones entre eventos para emular la causalidad, en caso de que los mismos eventos siempre producen los mismos resultados para todos los casos. Es cierto que la correlación no necesariamente implica causalidad. Pero la correlación es el pre-requisito de la causalidad y si la correlación entre antecedentes y consecuentes se repite para todos los casos de un mismo tipo de evento sin excepción (entropía informacional baja), uno puede comenzar a sospechar la existencia de la causalidad o al menos una aproximación bastante buena de causalidad. Otro método es tomar en cuenta el componente del tiempo y el componente de la complejidad o contexto que es el comodín para resolver ambigüedades [64].

En el libro “El Orden Sensorial”, el Dr. Joaquín Fuster (1999) parafraseó a Friedrich von Hayek:

> **La experiencia opera sobre eventos fisiológicos y los arregla en una estructura u orden el cual se vuelve la base de su significado mental. En el curso del desarrollo ontogenético o filogenético, un sistema de conexiones es formado el cual registra la frecuencia relativa con la cual grupos diferentes de estímulos internos o externos han actuado juntos sobre el organismo. Cada impulso individual o grupo de impulsos sobre su ocurrencia evoca otros impulsos los cuales corresponden a los estímulos que en el pasado han acompañado su ocurrencia [65].**

Queda claro que el cerebro es una máquina de inferencias probabilísticas. Entonces el método de maximum likelihood entre eventos se vuelve un buen caracterizador para la neuroplasticidad. ¿Pero qué tal si hay pocos eventos? Habría que incorporar el Laplace smoothing. ¿Y qué tal si el sistema funciona de una manera, el cerebro artificial aprende esta manera y luego el sistema cambia su comportamiento? La neuroplasticidad, por definición, debería resolver este problema. Y cualquier modelo matemático que caracterice la neuroplasticidad debería adaptarse constantemente a los cambios del sistema. Entonces, una solución simple es no usar todo el historial de eventos, sino más bien crear una cola de eventos FIFO (First In, First Out) que sólo recuerde una cantidad de eventos N y que los cálculos de probabilidades mediante maximum-likelihood estimation se realice en base a esos N eventos recordados y no a todos los eventos. En este caso las asociaciones se harían a nivel de contenedores multi-dimensionales [31].

Volviendo al modelo simple del subconsciente y la creatividad, los procesos de decisión markovianos se modelan con las ecuaciones de reinforcement learning. Hay varios tipos de reinforcement learning.

Este es uno de ellos [63]:

$$Cost(S,G,a)=\sum_{S'} P(S,S',a)\cdot[T(S,S',a)+\gamma\cdot Cost(S',G,\Pi(S',G))] \quad (6.1)$$

$$\Pi(S,G)=\underset{a}{argmin}\{Cost(S,G,a)\} \quad (6.2)$$

$$Cost(G,G,a)=0 \quad (6.3)$$

Donde:

S = Estado

S' = Estado siguiente

G = Estado Objetivo

a = Acción

Cost (S,G,a) = Costo de rutear desde el estado S hasta el estado objetivo G a través de la acción a

P (S,S',a) = Probabilidad de ir desde el estado S hasta el estado S' a través de la acción a

T (S,S',a) = Costo de la transición desde el estado S hasta el estado S' a través de la acción a

Π (S,G) = Política o acción refleja de cómo ir desde el estado S hasta el estado objetivo G

γ = factor que reduce el peso de los estados futuros en la toma de decisiones para favorecer a los estados más actuales

Si uno examina cuidadosamente estas ecuaciones y el proceso de decisión markoviano de la Figura 6.1, uno se da cuenta que los costos se propagan y se diluyen de manera probabilística desde los objetivos hasta cada estado alcanzable del sistema. Y si uno es más observador aún, queda claro que las iteraciones de estas ecuaciones a lo largo del tiempo exploran todas las posibles combinaciones de caminos causales que puedan haber hasta cierto horizonte de iteraciones. Y estas posibles combinaciones de caminos causales incluyen ciclos y lazos repetitivos, también conocidos como obsesiones. Los caminos futuros y los posibles ciclos repetitivos no afectan mucho las decisiones causales gracias al factor de reducción gamma.

Y si uno observa el desarrollo evolutivo de estas ecuaciones a lo largo del tiempo, especialmente cuando los campos causales del sistema varían, uno probablemente se da cuenta que las buenas noticias (acortamiento o apertura de caminos causales) se propagan bien gracias al argmin de la fórmula de las políticas. Pero las malas

noticias (alargamiento o bloqueo de caminos causales) NO se propagan bien debido al argmin de la fórmula de las políticas. Si el cerebro funciona con mecanismos similares al reinforcement learning y es muy probable que esto sea verdad, se puede decir que la mala propagación de las malas noticias tiene que ver con el síndrome del miembro fantasma que ha sido estudiado por el neurocientífico Vilayanur Ramachandran. Una persona nace con todas sus extremidades y aprende a lo largo de la vida los campos causales de estas. Luego ocurre la mala noticia de que pierde una extremidad y el cerebro al buscar los caminos causales de mínima acción, sigue prefiriendo los caminos causales de las representaciones cerebrales del miembro amputado en vez de adaptarse a la fea realidad. Otra explicación de dicho fenómeno es que el cerebro, el sistema nervioso y el cuerpo siempre trata de sanar y reconectar los nervios cortados y es preferible mantener esas representaciones fantasmas en caso que ocurra dicha sanación y reconexión. Pero con el tiempo dichas representaciones fantasmas son reutilizadas e invadidas por otras representaciones cerebrales vecinas. Por eso cuando las memorias referenciables por contenido de esas zonas vecinas son activadas, los miembros fantasmas también se activan [66, 67].

Para eliminar el síndrome del miembro fantasma en los cerebros artificiales, es necesario explorar todos los caminos posibles para analizar el origen de los costos causales. Si estos costos causales están basados en buenas condiciones del pasado, se debería verificar si estas condiciones todavía son válidas o si ya no lo son. El problema es que esto es muy costoso debido a las explosiones combinatoriales que ocurren con cada iteración del subconsciente artificial. Y este seguimiento produce una arborización. Y en esa arborización se debería comprobar la existencia de ciclos. Un gran problema.

La solución es aproximar el MDP y hacerlo un PAC MDP (Probably Approximately Correct Markov Decision Process). En vez de tomar en cuenta todos los posibles caminos causales para calcular el costo, sólo se toma en cuenta el camino más probable y menos costoso. El resultado es el siguiente:

$$Cost(S,G)=\min_{a}\{\max_{S'} P(S,S',a)\cdot[T(S,S',a)+Cost(S',G)]\} \tag{6.4}$$

$$Cost(G,G)=0 \tag{6.5}$$

Donde:

S = Estado

S' = Estado siguiente

G = Estado Objetivo

a = Acción

Cost (S,G) = Costo de rutear desde el estado S hasta el estado objetivo G

P (S,S',a) = Probabilidad de ir desde el estado S hasta el estado S' a través de la acción a

T (S,S',a) = Costo de la transición desde el estado S hasta el estado S' a través de la acción a

Esta fórmula no lo resuelve todo. Sólo hace que el cálculo de los costos causales se base en 1 sólo camino en vez de un árbol de múltiples caminos. Cuando el costo se basa en 1 sólo camino causal, es mucho más fácil validar si las condiciones descritas por ese camino todavía existen en la realidad y si no existen ciclos (obsesiones o círculos viciosos). Al verificarlo, el síndrome del miembro fantasma y las obsesiones o círculos viciosos desaparecen del cerebro artificial. Pero sus costos son estimaciones aproximadas. Es un algoritmo de PAC learning, no un algoritmo exacto. Pero el

cerebro no es perfecto, sólo lo suficientemente bueno como para sobrevivir con dignidad.

Otro problema significativo es que la programación dinámica es computacionalmente costosa y es necesario hacer algo al respecto sobretodo cuando el espacio mental tiene muchos estados. La solución a esto es hacer ruteos locales y formar clusters o agrupaciones de estados en el siguiente nivel de complejidad. A este nivel, el ruteo se hace a nivel de clusters. Y así sucesivamente, se van creando clusters que simplifican y relajan los cálculos sin preocuparse demasiado por los detalles de bajo nivel. Sino más bien, el ruteo se puede hacer a nivel global. El ruteo jerárquico evita las explosiones combinatoriales. La jerarquización debería ocurrir justo cuando los ruteos locales comienzan a volverse computacionalmente complejos.

El cerebro es jerárquico y explota la abstracción para simplificar sus procesos. Existe una gran cantidad de evidencia neuro-fisiológica y epistemológica que lo confirma. El Dr. Joaquín Fuster dedicó más de 40 años de su vida a mapear las funciones cognitivas en los cerebros de los primates y resumió sus hallazgos con el siguiente gráfico que

muestra la naturaleza jerárquica del cerebro [59]:

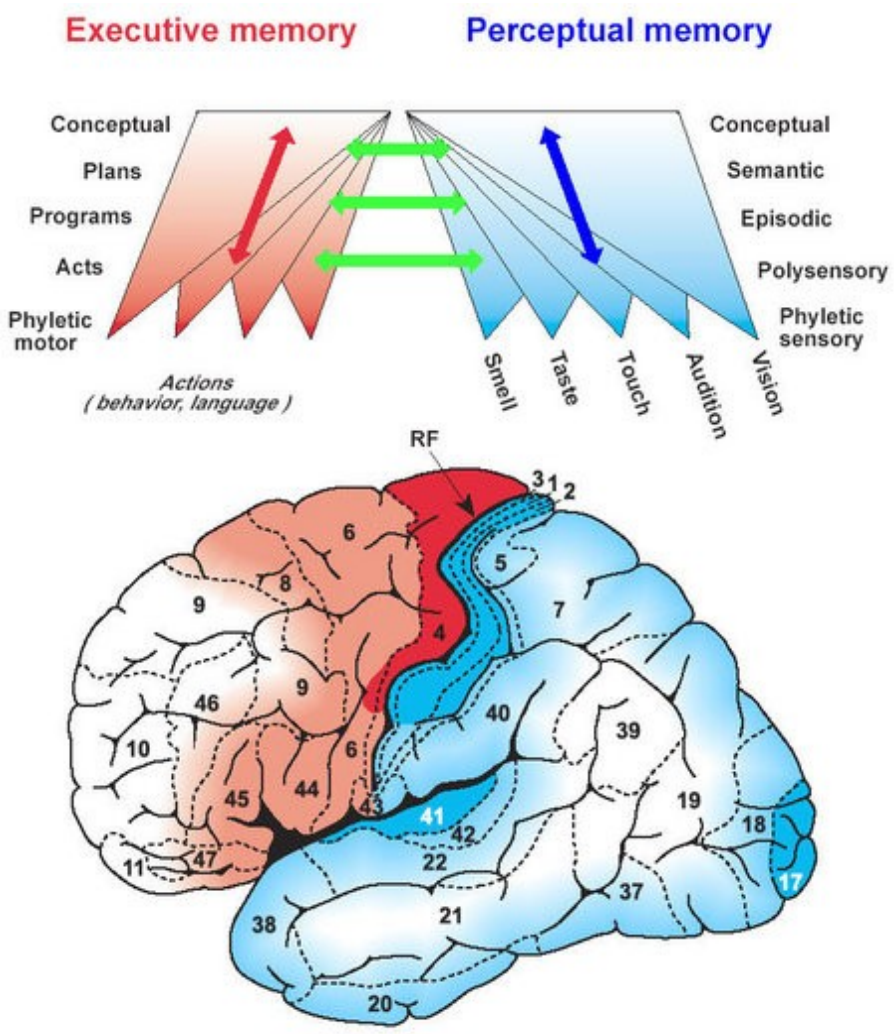


Figura 6.3: El mapeo de la memoria cortical y la teoría de los cognits del Dr. Joaquín Fuster [68]

Un problema serio que se debería evitar es la múltiple concurrencia:

Cuando los procesos de aprendizaje colisionan con los procesos de

auto-organización del subconsciente. Lo ideal es que el cerebro aprenda la geometría causal del sistema y luego auto-organice este conocimiento aprendido para que emerjan soluciones creativas y que se ejecuten sin pensar a través de acciones reflejas o políticas. Es muy probable que el subconsciente se auto-organice mientras dormimos. Pero sería interesante que las máquinas trabajen incesantemente sin dormir y que a la vez sean capaces de crear comportamientos reflejos y emergentes en tiempo real. Es ahí cuando los problemas de múltiple concurrencia aparecen. Este problema se resolvió muy fácilmente con Haskell, Software Transactional Memory (STM), transparencia referencial y valores inmutables. En lenguajes imperativos y con estado compartido como Java, resolver esto es bastante complicado [16].

6.3. Algunas Analogías entre los Sistemas Nerviosos Biológicos y los Sistemas Nerviosos Artificiales

El algoritmo de Bellman-Ford es una forma determinista de programación dinámica y tiene la siguiente fórmula:

$$D(x,y,z)=D(x,z,z)+\min_{w}\{D(z,y,w)\} \tag{6.6}$$

Todo algoritmo tiene estructuras de memoria y punteros que pueden ser diagramados. Al diagramar las estructuras de memoria y punteros generados por el algoritmo de Bellman-Ford con soporte de lógica difusa, se obtiene un gráfico similar a este:

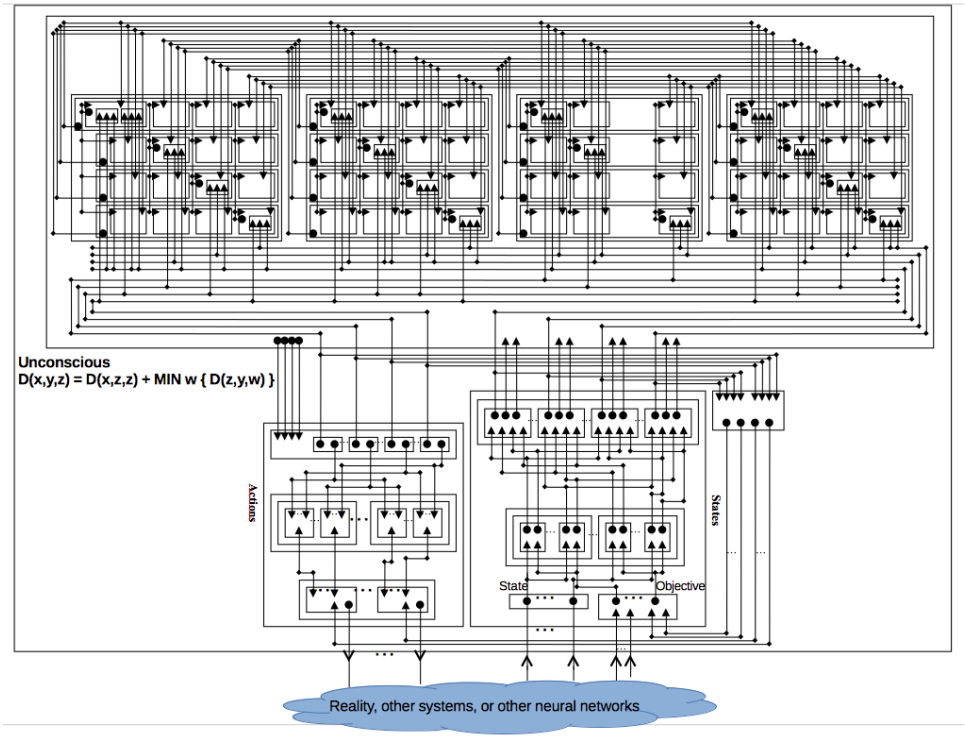


Figura 6.4: Representación gráfica del algoritmo Bellman-Ford

Este gráfico es una simplificación muy radical de lo que en realidad ocurre en la memoria del computador cuando se ejecuta el algoritmo

de ruteo causal. Las estructuras y punteros generados son mucho más abundantes.

Si se compara la parte superior de la Figura 6.4 con la parte superior de la Figura 6.5, uno se da cuenta que la capa L1 de las columnas corticales se parece a las interconexiones auto-asociativas entre los cognits artificiales (columnas corticales) de las matrices del algoritmo de Bellman-Ford.

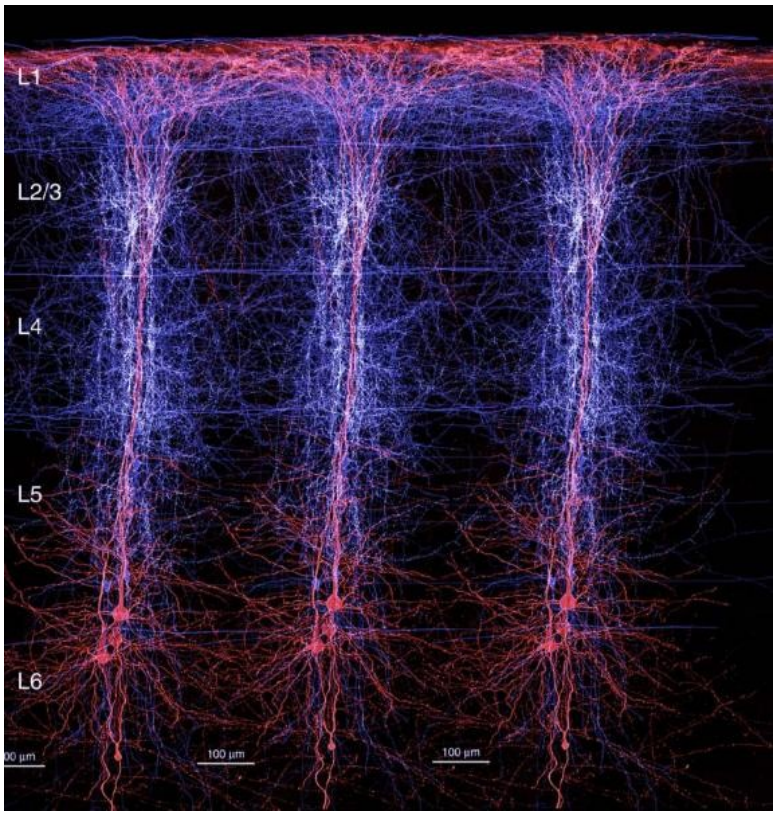


Figura 6.5: 6 capas de las columnas corticales [69]

Y también se parecen a las proyecciones auto-asociativas de las redes neuronales recurrentes en Figura 6.6 y Figura 6.7.

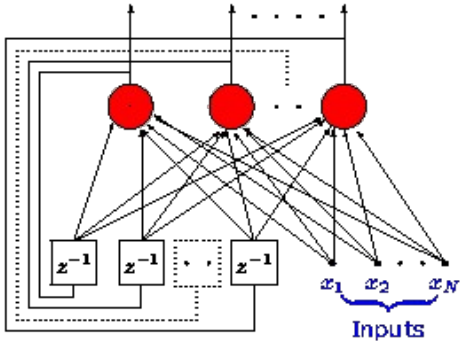


Figura 6.6: Red neuronal recurrente [70]

Las redes neuronales recurrentes procesan secuencias temporales de patrones como lo hacen los modelos markovianos. Y también son capaces de hacer reinforcement learning [71].

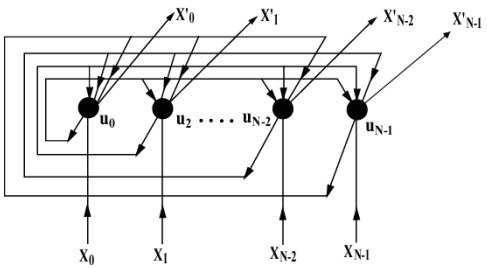


Figura 6.7: Otra red neuronal recurrente [72]

En las redes neuronales feedforward de varias capas (Figura 6.8) se puede ver que tan interrelacionadas están las neuronas artificiales: Todas las neuronas artificiales de una capa se conectan con todas las neuronas artificiales de otra capa para encontrar correlaciones. Las redes neuronales recurrentes (Figura 6.6 y Figura 6.7) son muy similares pero estas conectan una capa más profunda con otra capa menos profundas creando ciclos o secuencias temporales de patrones [71].

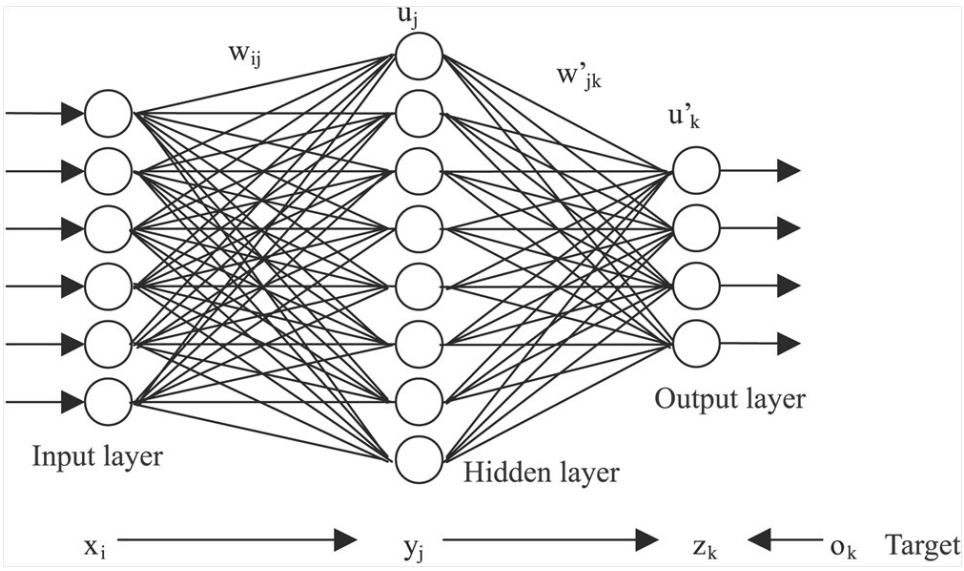


Figura 6.8: Red neuronal de 3 capas con muchas sinapsis [73]

La capa L1 de las columnas corticales (Figura 6.5) y la materia blanca (Figura 6.9) se parecen mucho a las conexiones de las redes

neuronales artificiales (Figura 6.8): Son axones que establecen correlaciones de todos los cognits de una región contra todos los cognits de otra región, directa o indirectamente [23,37]. La materia blanca está conformada por axones que interconectan varias regiones de materia gris que está conformada por cognits o columnas corticales capaces de reconocer y generar patrones a través de las interconexiones con otros cognits [59].

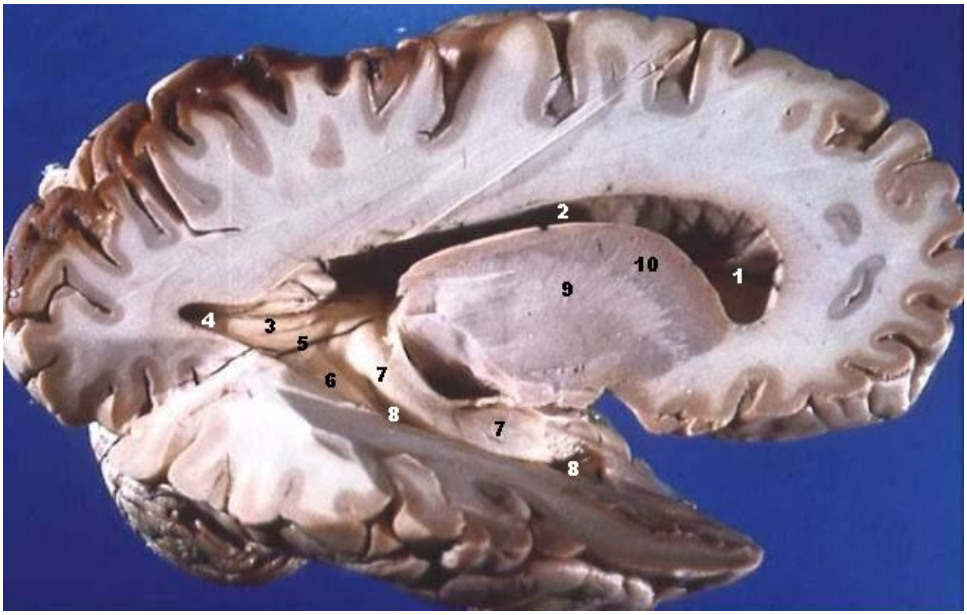


Figura 6.9: Materia blanca conformada por axones que conectan la materia gris [74]

Un patrón contiene estructuras y relaciones que conforman patrones más complejos y así sucesivamente. Los patrones son relaciones estructurales sobresalientes. Las redes neuronales llevan este

paradigma al extremo porque correlacionan todos los cognits de una región contra todos los cognits de otra región para hallar relaciones estructurales sobresalientes. Es la representación más universal y holista que puede haber: El teorema de aproximación universal de Cybenko [37].

En el libro “On Intelligence”, Jeff Hawkins (2004) explicó como funcionan las memorias corticales:

> **La regiones corticales descubren como los sensores están relacionados, memorizan la secuencia de correlaciones entre ellos y usan esta memoria para predecir como los sensores se comportarán en el futuro [6].**

En el libro “On Intelligence”, Jeff Hawkins (2004) definió a los invariantes neuronales:

> **La memorias son almacenadas en una forma que captura la esencia de las relaciones, no los detalles del momento. Cuando ves, sientes o escuchas algo, el cortex toma las sensaciones detalladas y altamente específicas y las convierte en invariantes. Los invariantes son almacenados en memoria. Y son los invariantes de cada patrón de sensación los que se comparan. El almacenamiento, recuerdo y reconocimiento de memorias ocurre a nivel de invariantes [6].**

¿Qué son los invariantes neuronales? En física, cualquier invariante o

función obtenida por el proceso del reduccionismo es, por definición, una relación causal [15]. Las redes neuronales son aproximaciones funcionales adaptadas a los datos experimentales [71]. Este hecho sugiere que las redes neuronales son relaciones causales complejas o invariantes neuronales. Generalmente, los invariantes de la física son funciones uniformes mientras que los invariantes neuronales pueden tener patrones arbitrarios.

En el libro “On Intelligence”, Jeff Hawkins (2004) también definió a los patrones neuronales:

> **En todo momento, un conjunto de fibras disparará pulsos eléctricos, también llamados potenciales de acción o picos, mientras otras fibras permanecerán calladas. Un patrón es la actividad colectiva de un conjunto de fibras [6].**

La arborización en las dendritas y los axones (Figura 6.10) es una forma fractal de correlacionar rangos amplios de neuronas, como las redes neuronales artificiales lo hacen. Los árboles fractales en las neuronas maximizan la superficie de contacto 2D de las sinapsis y minimizan el volumen 3D de las neuronas que es necesario para alcanzar dicha superficie de contacto 2D. Esta dicotomía entre 2D y 3D crea una dimensión fractal entre 2 y 3 [75]. El isomorfismo de las

columnas corticales y sus heterarquías generan otro fractal probabilístico [76].

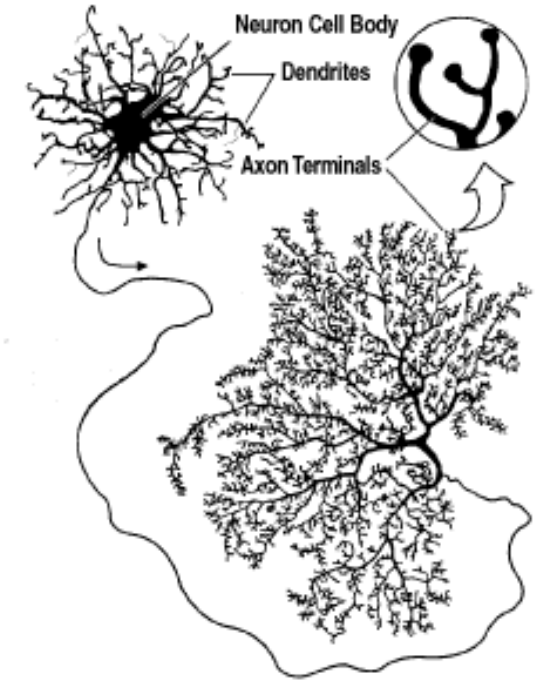


Figura 6.10: Árbol dendrítico y árbol axónico de la neurona [77]

Los fractales puros e infinitamente auto-similares de dimensión 1.x tienen un perímetro infinito con formas auto-similares que está encerrado en una superficie finita. Los fractales puros e infinitamente auto-similares de dimensión 2.x tienen una superficie infinita con formas auto-similares que está encerrada en un volumen finito. En los fractales probabilísticos, la auto-similitud no necesariamente es

exacta ni tampoco infinita y sus patrones pueden variar. Los fractales probabilísticos de dimensión 1.x maximizan su perímetro con el que forman patrones auto-similares y minimizan la superficie que los encierra. Los fractales probabilísticos de dimensión 2.x maximizan su superficie con la que forman patrones auto-similares y minimizan el volumen que los encierra. El libro “Introducing Fractal Geometry” explica todo esto y además sugiere que las neuronas no son las únicas en explotar las propiedades de las estructuras arborícoras y fractálicas. Los capilares, el sistema circulatorio, los pulmones y otros órganos también lo hacen. Muchas estructuras en el cuerpo humano están compuestas de diferentes tipos de fractales (con diferentes dimensiones fractálicas) que están entretejidos entre ellos. Por ejemplo: Nuestros pulmones fractálicos abarcan el área de una cancha de tenis dentro del volumen de tan sólo unas pocas pelotas de tenis. La superficie de contacto entre los alveolos y el aire se maximiza. Y el volumen de los pulmones se minimiza para caber dentro del pecho [75]. Otro ejemplo: Nuestra corteza cortical se comprime en circunvoluciones que maximizan su área y por ende maximizan el número de columnas corticales o cognits que están encerrados en un volumen craneal limitado [6].

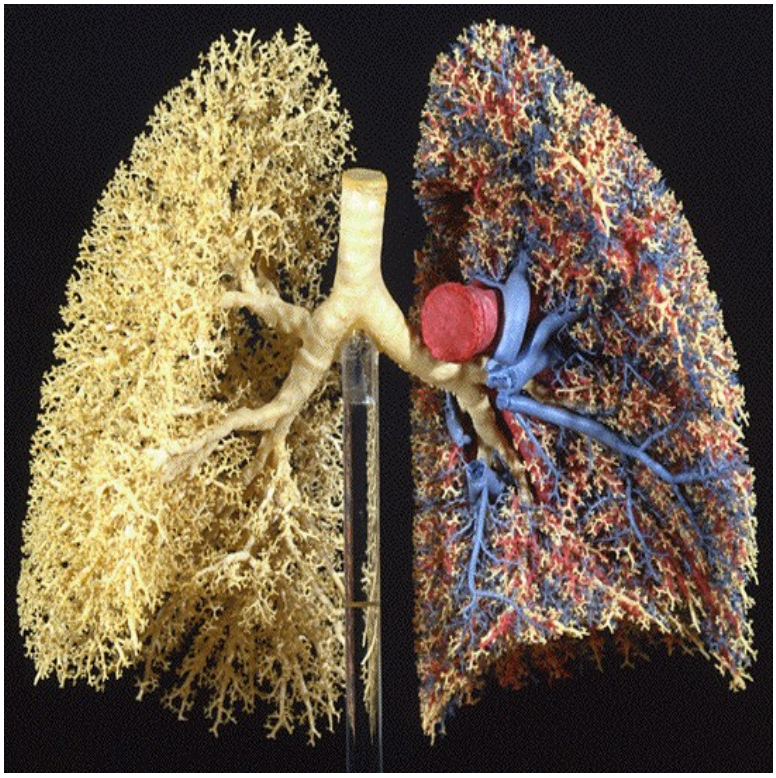


Figura 6.11: Nuestros pulmones fractálicos abarcan el área de una cancha de tenis dentro del volumen de tan sólo unas pocas pelotas de tenis [78]

Una neurona promedio tiene entre 1.000 y 10.000 sinapsis. Esto quiere decir que las ilustraciones de las neuronas biológicas y las redes neuronales artificiales se quedan cortas con respecto al número de sinapsis o conexiones que realmente tienen las neuronas reales. Las neuronas spindle son las neuronas de las emociones en el sistema límbico y están mucho más interconectadas puesto que tienen 100.000 sinapsis aproximadamente. Esto sugiere que las emociones son patrones más holistas, más complejos y de largo

plazo [48].

Un ejemplo muy sencillo de patrones son los patrones de costura (Figura 6.12). A bajo nivel, se puede observar formas y medidas que son relaciones estructurales sobresalientes. A alto nivel, se puede observar la geometría euclideana y el sistema métrico que son comunes para todas las mediciones de bajo nivel.

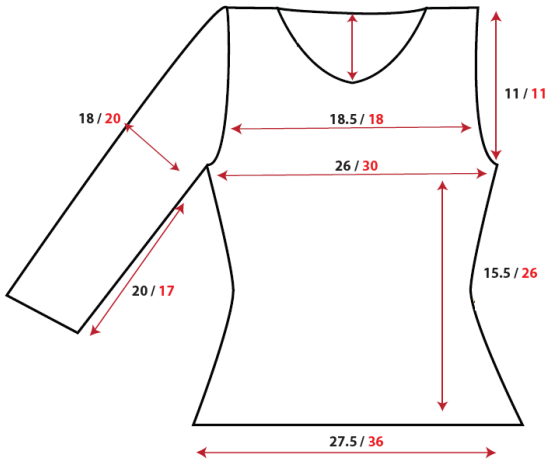


Figura 6.12: Patrón de costura con medidas y relaciones sobresalientes [79]

Otro ejemplo muy sencillo de patrones son los patrones y las relaciones estructurales sobresalientes en rostros. En la Figura 6.13, se puede ver que las fotos son traducidas a un lenguaje de relaciones estructurales sobresalientes o invariantes para luego ser comparadas

con las bases de datos de patrones de rostros. Los cerebros hacen algo similar a esto [6].

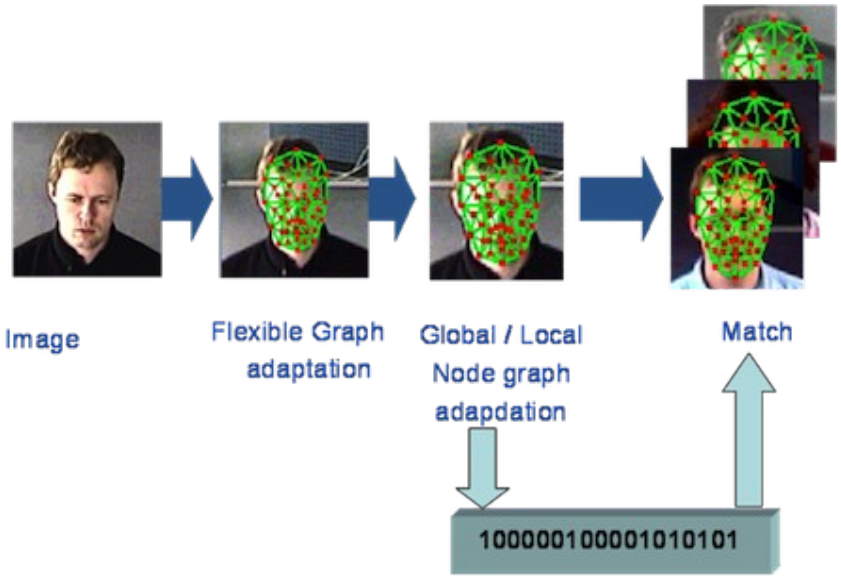


Figura 6.13: Patrones y relaciones estructurales sobresalientes en rostros [80]

En los sistemas de inteligencia artificial, nunca se debe pre-programar algoritmos pre-existentes. Es mejor dejar que la máquina descubra las geometrías causales de la Realidad y las relaciones estructurales sobresalientes (patrones). La reducción es una exploración proactiva e inconsciente de todo el espacio de recursos mentales, patrones de la mente, e hipótesis. No es una receta directa y pre-programada para resolver un problema. No es un sistema reduccionista. Es más bien un problema inverso en el cual la mente trata holísticamente de

encontrar la receta. Si la mente no puede encontrar los patrones mentales correspondientes para resolver problemas o para explicar fenómenos, la mente trata de aprender o crear los elementos claves y las piezas faltantes del rompecabezas mental. Entonces, tanto el tiempo del algoritmo de reducción como las recetas resultantes son totalmente impredecibles, imperfectos, y no-garantizados. Las recetas exitosas son siempre guardadas [3].

En el caso del reconocimiento de patrones, las características sobresalientes son aquellas que son críticas y cruciales para reconocer el patrón. Si se quitan de la representación de patrones, el reconocedor de patrones comenzará a fallar. Lo mismo se aplica a los modelos de pensamiento: Si se omite una característica crucial y crítica, el comportamiento de la abstracción se desviará completamente de la entidad real. Y también se aplica a la planeación: Si se omite un ingrediente crucial y crítico en la receta para resolver el problema, no se logrará alcanzar el objetivo. La búsqueda de patrones sobresalientes es simplemente la navaja de Ockham aplicada a las funciones cerebrales. Y también está profundamente relacionada con la criticalidad: Algunas cosas pueden no tener importancia. Otras son muy críticas. Así se aprende qué es

importante [3].

Los procesos de aprender patrones sobresalientes y su reforzamiento a través de la prueba de sus predicciones están interrelacionados porque son experimentales. La búsqueda de patrones sobresalientes es como el estereotipado. Por ejemplo: Uno puede predecir que un perro ladrará y tendrá 4 patas y una cola, si sólo se ve la cabeza del perro. Si no es así y las predicciones del estereotipo son incorrectas, será un perro amorfo con patrones anómalos o una categoría distinta [6].

La solución para el problema de la reducción autónoma consiste en inervar un conjunto hipercompleto de neuronas artificiales y a través del darwinismo neuronal se selecciona el conjunto de patrones temporo-espaciales que importan para la tarea. Sea esta reconocimiento de patrones, abstracción o planeación. Un conjunto simple pero no más simple de lo necesario. En otras palabras, es una meta-optimización en la cual se busca la topología neuronal más simple que pueda representar correctamente los datos que se desea entender. Meta-optimizar topologías neuronales encuentra patrones sobresalientes porque la meta-optimización obliga a los patrones a

pasar por el canal más estrecho que es capaz de reconstruir los datos [37]. Los cerebros inervan señales casi redundantes porque sólo las asimetrías pueden ser detectadas y porque patrones sobresalientes pueden ser encontrados en señales correlacionadas.

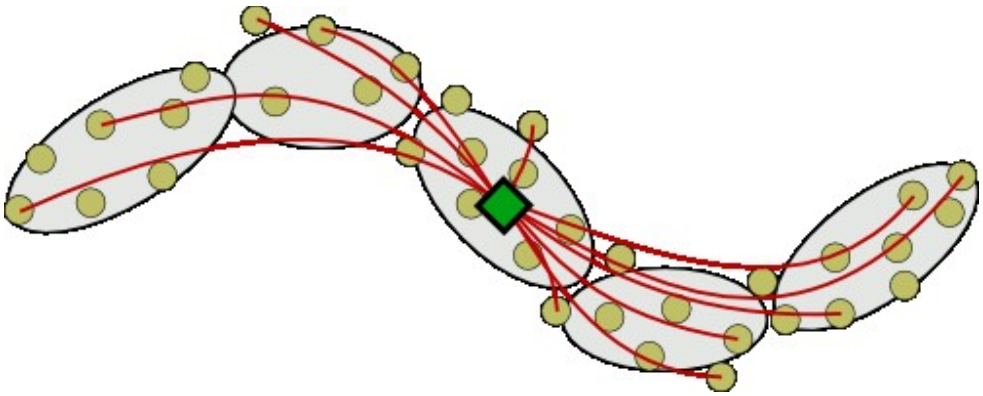


Figura 6.14: Señales cuasi-redundantes y correlacionadas en los sistemas nerviosos [81]

En el libro “Recurrent Neural Networks and Soft Computing”, Igor Belič (2012) explicó la diferencia entre las aproximaciones basadas en modelos y las aproximaciones sin modelos:

> **Otro detalle muy importante que justifica el uso de las redes neuronales es la ausencia de una descripción matemática apropiada del problema a modelar. Las redes neuronales son aproximadores sin modelos (Figura 6.15), esto significa que son capaces de modelar sin tener conocimiento previo de la naturaleza del sistema a modelar. Para las técnicas clásicas de aproximación, a menudo es necesario conocer el modelo matemático básico del problema aproximado. La aproximación de los mínimos**

cuadrados (modelos de regresión), por ejemplo, busca el mejor ajuste de los datos proveídos para la función conocida que representa el modelo (Figura 6.16) [71].

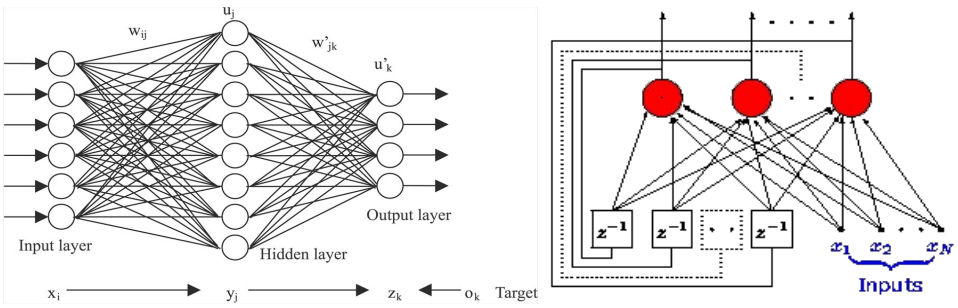


Figura 6.15: Las redes neuronales son aproximadores sin modelos

Recordando el capítulo del sistema básico de visión artificial:

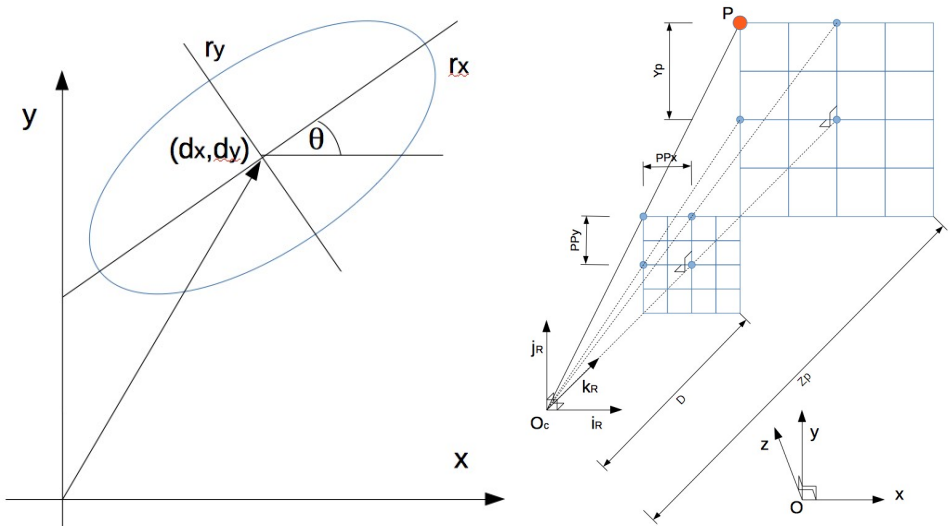


Figura 6.16: La aproximación de mínimos cuadrados (modelos de regresión) busca el mejor ajuste de los datos proveídos para las funciones conocidas que representan los modelos

Por ejemplo, las cuerdas vocales, bocas, labios y lenguas obedecen a las leyes de la física pero tienen patrones estructurales complejos que los hacen dominios raros. Entonces los cerebros nunca caracterizan las voces con matemáticas. Las redes neuronales más bien aprenden su esencia holista sin explicaciones causales.

En general, X1, X2, ... , Xn obedecen a las leyes de la física pero tienen patrones estructurales complejos que los hacen dominios raros. Entonces los cerebros nunca caracterizan sus efectos con matemáticas. Las redes neuronales más bien aprenden su esencia holista sin explicaciones causales.

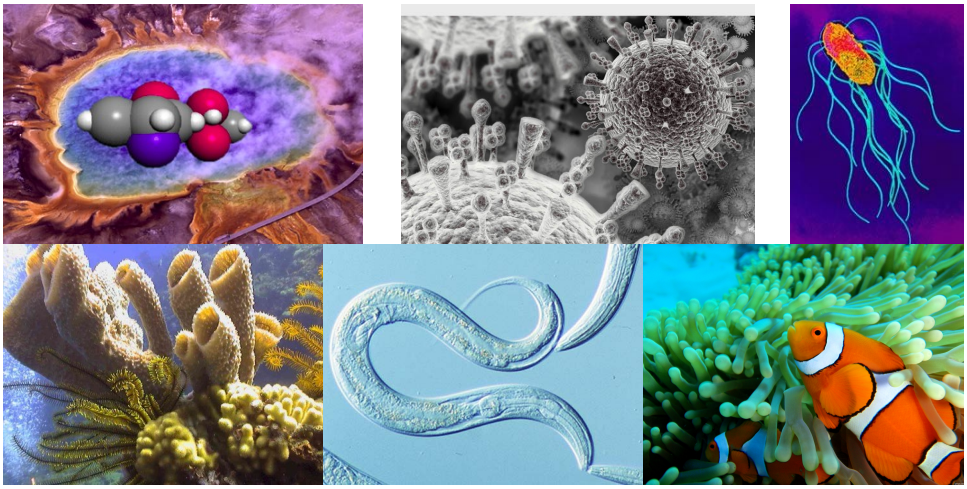


Figura 6.17: Algunas ramificaciones de la historia de la vida según todos los biólogos y neurocientíficos del mundo [82,83,84,85,86,87]

En la Figura 6.17, se puede ver algunas ramificaciones de la historia de la vida. La sopa primordial, predicha hacia el pasado por Stanley Miller, dio origen a los componentes del DNA: T, C, G, A. Estas moléculas son capaces de codificar, interiorizar y auto-replicar información relevante a la supervivencia de los organismos. Estas moléculas formaron sinergias y simbiosis en varios niveles de complejidad para dar origen a protovirus, virus, proto-organismos-unicelulares, organismos unicelulares, organismos pluricelulares, reflejos, movimientos, sensaciones, percepciones, predicción de acciones, cerebros, etc [54].

En la evolución biológica, la evolución tomó el camino de la mimetización de inversas causales para predecir los movimientos por muchas razones. Si no fuera así, no existiríamos. En la evolución tecnológica, los investigadores de inteligencia artificial afrontamos la misma decisión. Tenemos que decidir entre la mimetización de inversas causales para predecir los movimientos de los robots o modelar inflexiblemente y perfectamente con el reduccionismo. El camino de la mimetización de inversas es aproximado y propenso a errores. En cambio, el camino del reduccionismo inflexible y perfecto es imposible. Porque los sistemas raros tienen observabilidad parcial;

son muy complejos y caóticos; y es necesario transferir sus patrones sensados desde la Realidad hacia los cerebros artificiales. En otras palabras, es necesario aprender sus patrones e interrelaciones. No hay otro camino [3].

6.4. Demostración del Ruteo Causal con un Laberinto Editable

Para ver la demostración del Laberinto Editable, se abre una ventana de terminal. Luego se entra al directorio de la tesis y se ejecuta el script editable-maze.sh. Se debería ver una ventana como esta:

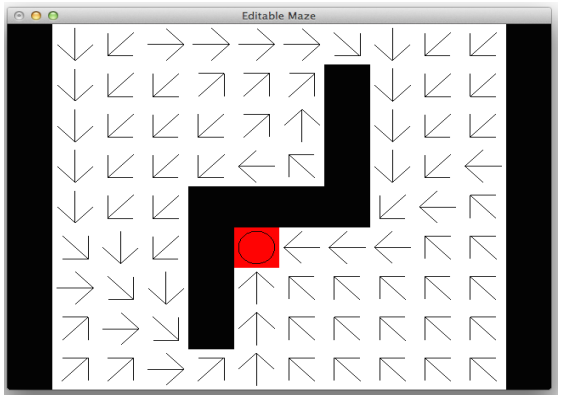


Figura 6.18: Aplicación del Laberinto Editable

Si se hace click izquierdo en el laberinto, se crean o se borran sus bloques. Y de esta forma la geometría causal del sistema cambia. Y

en medio de estos cambios, el cerebro artificial es capaz de auto-organizar y reprogramar sus conexiones causales en tiempo real, sin colisiones de memoria entre los procesos de aprendizaje y de auto-organización. (Gracias a Haskell [9] y a STM [16].)

El círculo rojo es el objetivo y las flechas indican los reflejos condicionados o políticas para alcanzar ese objetivo. El objetivo se lo cambia automáticamente al mover el ratón.

Si se modifica lo suficiente el laberinto, se crean buenas y malas noticias. Ambos tipos de noticias son propagadas sin ningún problema. Y la naturaleza aproximada y probabilística del algoritmo no afecta la coherencia de las decisiones.

El modelo de transición del robot virtual en grid world es el siguiente: Si se mueve en una dirección, es probable que se mueva a esa dirección pero también es probable que no se mueva o que sí se mueva pero un poco desviado hacia los lados.

Cuando recién se propagan los costos a través de los caminos causales o cuando hay malas noticias, uno se da cuenta como los

costos se propagan desde los objetivos hacia todos los estados del sistema. Así es como funciona la programación dinámica.

Esta demostración no tiene capacidad de aprendizaje porque el campo causal es totalmente observable por el sistema. Sin embargo, el ruteador causal debería aprender los campos causales a partir de la exploración y generar reflejos condicionados a partir de la auto-organización. No todos los campos causales son campos invariantes. Los campos invariantes tienen una entropía informacional baja puesto que concentran toda su causalidad en distribuciones probabilísticas más estrechas. Por ejemplo: Las sinergias neuro-motrices y la geometría proyectiva de la visión. Los reflejos condicionados (Figura 6.18) son rutas causales precalculadas aplicables a campos invariantes. Pueden ser ejecutivos (políticas) o perceptivos (predicciones).

CAPÍTULO 7

7. PROBLEMA NO RESUELTO: BRAZO ROBÓTICO QUE ESCRIBE EN UNA PIZARRA VIRTUAL

7.1. Introducción

Si se toma el problema del doble péndulo invertido y se le agrega 2 motores capaces de generar torques para controlarlo, se obtiene un brazo robótico. Este brazo robótico se lo puede modelar con la mecánica de Lagrange. El resultado es un espacio multidimensional altamente caótico, continuo, simétrico y platónico. Este brazo virtual

es puramente platónico. En realidad no existen brazos reales que sean continuos y simétricos en el mundo real.

El cerebro es capaz de capturar imperfectamente la esencia de los sistemas raros a través de las redes complejas de patrones. Estas redes aprenden la complejidad de los sistemas raros a través del darwinismo neuronal descrito por el neurocientífico Gerald Edelman. Es un proceso de asociación, adaptación, predicción, expectativa y refuerzo o debilitamiento dependiendo si se cumplen o no las predicciones [5,13,88].

Las redes complejas de patrones pueden aprender tanto los sistemas raros como los sistemas simétricos producidos por la ciencia y las matemáticas. Porque los sistemas simétricos son mucho más predecibles que los sistemas raros.

En teoría, si se le provee suficientes recursos computacionales a un cerebro artificial, este sería capaz de aprender las geometrías causales y simétricas del brazo robótico descrito en este capítulo. Pero lamentablemente, este brazo es impresionantemente incontrolable puesto que no es un brazo común y corriente. La forma

en que se aplica los torques al brazo son muy similares a como los motores eléctricos mueven a un ventilador. O sea, que controlar este brazo virtual tiene una dificultad similar a la que tiene controlar la posición de las aspas de un ventilador. De hecho, si se aplica la máxima potencia a los 2 torques en la misma dirección, el brazo virtual parece un ventilador. Controlar este brazo virtual es un poco más difícil que controlar precisamente la posición de las aspas de un ventilador porque las posiciones de las aspas tienen simetría bilateral y de esa forma hay equilibrio estático y dinámico. Mientras que controlar el brazo virtual siempre tiene la tendencia a caer y controlarlo precisamente mientras esta arriba es muy complicado. Además, hay que recordar que este brazo robótico es un doble péndulo invertido con 2 motores controladores. Lo que es equivalente a tener un ventilador con una aspa y en el extremo de la aspa hay otro ventilador con una aspa. Todo lo dicho anteriormente, se puede probar en la demostración virtual. En vez de utilizar motores rápidos y que pueden girar completamente, sería mejor utilizar “gimbal motors” que son más lentos, más potentes y no pueden girar completamente. Los robots reales son mucho más controlables que el robot virtual de la tesis.

Para controlar ventiladores o brazos virtuales como este, es necesario discretizar el espacio simétrico en contenedores multidimensionales usando lógica difusa. Una discretización difusa debería tener al menos los siguientes elementos para ser algo aceptable: Cada articulación debería tener al menos 16 ángulos (cada porción del cuadrante tendría 22.5 grados) y 5 velocidades angulares (muy negativo, negativo, neutro, positivo y muy positivo). O sea, 80 combinaciones por cada articulación. Y hay 2 articulaciones: 80^2 = 6.400 combinaciones por las 2 articulaciones. Cada motor tendría al menos 5 tipos de torques: Muy negativo, negativo, neutro, positivo y muy positivo. Como hay 2 motores entonces habrían 5^2 = 25 tipos de sinergias motrices o combinaciones de acciones. 6.400 * 25 = 160.000 combinaciones de variables difusas por brazo. Y a esto hay que hacer iteraciones para auto-organizar subconscientemente los reflejos emergentes del sistema. Cada iteración del sistema tiene orden polinómico. Es decir, 160.000^N cálculos por iteración del subconsciente artificial, donde N es el orden del polinomio. Lo cual imposibilita momentáneamente la aplicabilidad de este cerebro artificial debido a la maldición de la dimensionalidad de Bellman [89].

Controlar perfectamente a este brazo robótico virtual para que dibuje

las letras en la pizarra virtual podría incluso requerir más resolución cognitiva que el estimado anterior. La imposibilidad de aplicar este cerebro artificial va a cambiar pronto puesto que actualmente hay GPUs con 5,000+ cores y con 12 Gb de memoria pero cuestan $3,000 aproximadamente. Es decir, gracias a la ley de los retornos acelerados y la ley de Moore, pronto los computadores tradicionales tendrán similares características a un precio reducido. Incluso, los smartphones tendrán similares características [76,90].

Para resolver este problema con menos recursos computacionales habría que resolver el binding problem que consiste en identificar regiones con características similares y sus relaciones en el espacio caótico del sistema dinámico del brazo virtual, con el fin de reducir la dimensionalidad del problema. Esto es similar a la identificación de objetos y sus relaciones en una escena. Es similar al binding entre las sensaciones y las percepciones. Desafortunadamente, el binding problem es un problema no resuelto de la inteligencia artificial que miles de pensadores lo han analizado y lo han tratado de resolver durante miles de años. Pero es muy probable que pronto será resuelto. Es probable que el deep learning ayude a resolverlo. Las redes neuronales recurrentes (RNN) son similares a los procesos de

decisión markovianos (MDP). Pero a diferencia de los MDP, las RNN son naturalmente jerarquizables y reducen la dimensionalidad del problema por medio de regiones complejas y sobresalientes de patrones que generalizan la lógica difusa. Es probable que la solución del aprendizaje y auto-organización en tiempo real de geometrías causales venga por el camino de las RNN. Lo malo es que las RNN son mucho más costosas computacionalmente que los MDP. Pero hay métodos matemáticos muy prometedores que van a acelerar radicalmente el tiempo computacional de las redes neuronales en general.

A pesar de no haber resuelto el problema, la geometría del brazo robótico es un ejemplo claro de un espacio simétrico y sirve para contrastarlo con los espacios raros que el cerebro maneja. Hay que enfatizar la importancia de los espacios raros en el cerebro. Los cerebros de los animales nunca aprenden matemáticas reduccionistas y sin embargo se mueven y ven mucho mejor que los robots basados en matemáticas reduccionistas. Un ser humano puede aprender matemáticas reduccionistas pero estas son simbólicas y de alto nivel. Para moverse o para ver, los seres humanos y los animales en general nunca usan matemáticas

reduccionistas. Los bebés nunca aprenden la mecánica de Lagrange para moverse ni la geometría proyectiva para ver. Más bien mapean la esencia y lo más relevante de los espacios raros. Los cerebros realizan sus funciones cognitivas mediante mecanismos asociados a estructuras moleculares mientras que la inteligencia artificial funciona con algoritmos y matemáticas ejecutadas en una máquina de Turing.

Asimismo, los animales nunca calculan inversas ni nunca hacen inducción causal como los matemáticos lo hacen. (Leer los capítulos 4 y 5.) Los matemáticos caracterizan los fenómenos de la naturaleza con curvas y fórmulas mediante el reduccionismo científico. Y luego, si necesitan calcular algo, despejan las variables o calculan la solución de las ecuaciones diferenciales. Los animales nunca hacen esto. Más bien, mediante sus redes cognitivas, los animales aprenden los patrones, el contexto y la aplicabilidad de las inversas de los espacios raros en los cuales viven. Esta es la forma como los animales hacen inducción causal y solucionan el problema inverso de la inteligencia. Por cierto, aprender el control difuso de los robots también es un problema inverso.

7.2. Modelo Matemático Reduccionista del Brazo Robótico

Para modelar un brazo robótico virtual, la siguiente figura es un buen modelo físico:

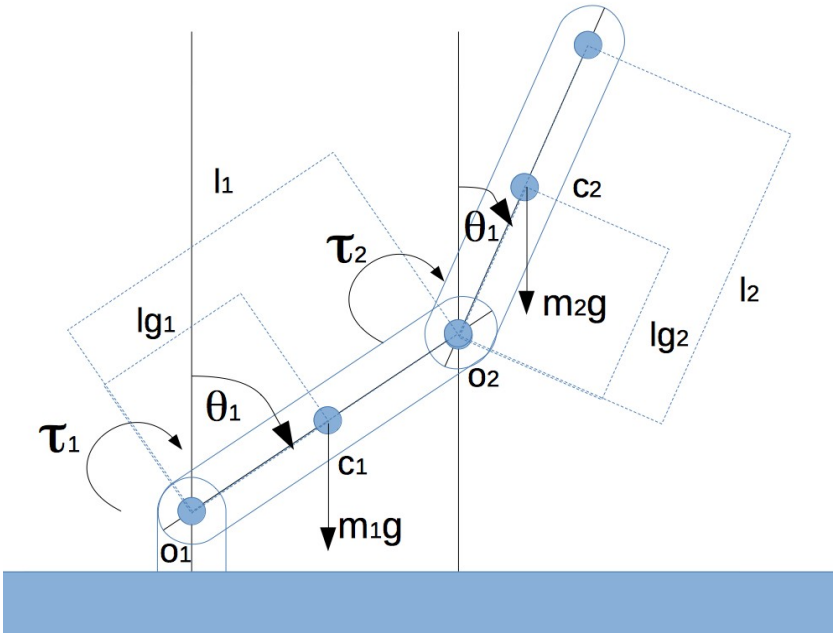


Figura 7.1: Modelo físico del brazo robótico

En la figura anterior, las partes del brazo tienen forma arbitraria y los centros de masa son C_1 y C_2. Pero en la tesis, las partes del brazo se modelan como placas rectangulares girando en torno a su extremo como en la siguiente figura:

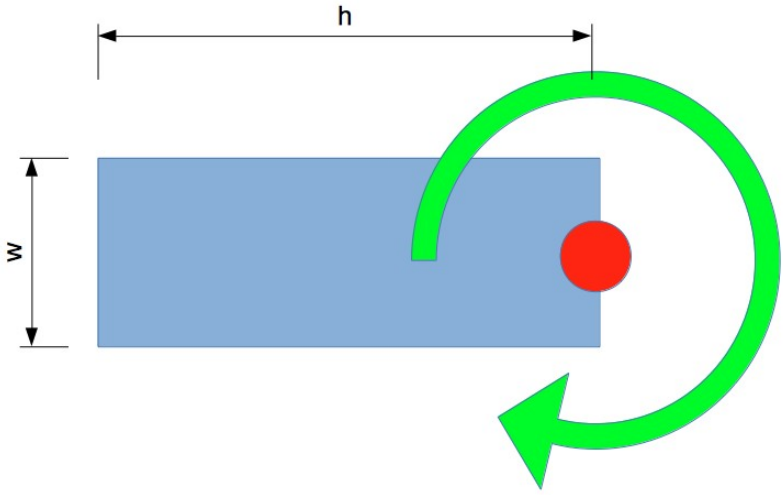


Figura 7.2: Placa rectangular girando en torno a su extremo

El momento de inercia de una placa rectangular que gira en torno a su extremo es [1]:

$$I_e = \frac{m \cdot h^2}{3} + \frac{m \cdot w^2}{12} \tag{7.1}$$

El momento de inercia de (7.1) fue calculado por la siguiente integral volumétrica:

$$I_P = \int_V \rho(r) \cdot r^2 \cdot dV \tag{7.2}$$

Antes de proceder al cálculo del Lagrangiano, es necesario calcular

los puntos relevantes del sistema. Para esto es necesario la matriz de rotación en sentido de las manecillas del reloj:

$$R(-\theta)=\begin{bmatrix}\cos\theta & \sin\theta\\ -\sin\theta & \cos\theta\end{bmatrix} \tag{7.3}$$

Para calcular la posición de una varilla que gira en torno a su extremo en sentido de las manecillas del reloj y que su ángulo inicial es el eje de las y, es necesario usar estas fórmulas:

$$d_i=\begin{bmatrix}\cos\theta_i & \sin\theta_i\\ -\sin\theta_i & \cos\theta_i\end{bmatrix}\cdot\begin{pmatrix}0\\ D_i\end{pmatrix}=\begin{pmatrix}0+D_i\cdot\sin\theta_i\\ 0+D_i\cdot\cos\theta_i\end{pmatrix}=\begin{pmatrix}D_i\cdot\sin\theta_i\\ D_i\cdot\cos\theta_i\end{pmatrix} \tag{7.4}$$

Usando las fórmulas anteriores, se puede calcular la posición del centro de masa 1:

$$C_1=O_1+l_{g1}\cdot d_1=\begin{pmatrix}O_{1x}+l_{g1}\cdot\sin\theta_1\\ O_{1y}+l_{g1}\cdot\cos\theta_1\end{pmatrix}=\begin{pmatrix}x_1\\ y_1\end{pmatrix} \tag{7.5}$$

Y la velocidad del centro de masa 1:

$$\dot{C}_1=\begin{pmatrix}\dot{x}_1\\ \dot{y}_1\end{pmatrix}=\begin{pmatrix}l_{g1}\cdot\cos\theta_1\cdot\dot{\theta}_1\\ -l_{g1}\cdot\sin\theta_1\cdot\dot{\theta}_1\end{pmatrix} \tag{7.6}$$

También se puede calcular la posición de la segunda articulación:

$$O_2 = O_1 + l_1 \cdot d_1 = \begin{pmatrix} O_{1x} + l_1 \cdot \sin\theta_1 \\ O_{1y} + l_1 \cdot \cos\theta_1 \end{pmatrix} \tag{7.7}$$

Y se puede calcular el centro de masa 2:

$$C_2 = O_2 + l_{g2} \cdot d_2 = \begin{pmatrix} O_{2x} + l_{g2} \cdot \sin\theta_2 \\ O_{2y} + l_{g2} \cdot \cos\theta_2 \end{pmatrix} = \begin{pmatrix} O_{1x} + l_1 \cdot \sin\theta_1 + l_{g2} \cdot \sin\theta_2 \\ O_{1y} + l_1 \cdot \cos\theta_1 + l_{g2} \cdot \cos\theta_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \tag{7.8}$$

Y la velocidad del centro de masa 2:

$$\dot{C}_2 = \begin{pmatrix} \dot{x}_2 \\ \dot{y}_2 \end{pmatrix} = \begin{pmatrix} l_1 \cdot \cos\theta_1 \cdot \dot{\theta}_1 + l_{g2} \cdot \cos\theta_2 \cdot \dot{\theta}_2 \\ -l_1 \cdot \sin\theta_1 \cdot \dot{\theta}_1 - l_{g2} \cdot \sin\theta_2 \cdot \dot{\theta}_2 \end{pmatrix} \tag{7.9}$$

Una vez que se ha calculado todas las coordenadas relevantes y sus respectivas derivadas con respecto al tiempo (derivadas temporales), se procede a calcular el Lagrangiano:

$$L = \frac{1}{2} \cdot I_1 \cdot \omega_1^2 + \frac{1}{2} \cdot m_1 \cdot v_1^2 - m_1 \cdot g \cdot y_1 + \frac{1}{2} \cdot I_2 \cdot \omega_2^2 + \frac{1}{2} \cdot m_2 \cdot v_2^2 - m_2 \cdot g \cdot y_2 \tag{7.10}$$

Donde:

$$v_i^2 = \dot{x_i}^2 + \dot{y_i}^2 \tag{7.11}$$

Al reemplazar todas las coordenadas y sus derivadas temporales en el Lagrangiano y al simplificar las expresiones resultantes, obtenemos la siguiente expresión final del Lagrangiano:

$$L = \frac{1}{2}\cdot\dot{\theta_1}^2\cdot[I_1 + m_1\cdot {l_{g1}}^2] + \frac{1}{2}\cdot I_2\cdot\dot{\theta_2}^2 - m_1\cdot g\cdot(O_{1y} + l_{g1}\cdot\cos\theta_1) - m_2\cdot g\cdot(O_{1y} + l_1\cdot\cos\theta_1 + l_{g2}\cdot\cos\theta_2) + \frac{1}{2}\cdot m_2\cdot[{l_1}^2\cdot\dot{\theta_1}^2 + 2\cdot l_1\cdot l_{g2}\cdot\dot{\theta_1}\cdot\dot{\theta_2}\cdot\cos(\theta_1 - \theta_2) + {l_{g2}}^2\cdot\dot{\theta_2}^2] \tag{7.12}$$

Ahora se procede a aplicar las ecuaciones generalizadas del movimiento y de las fuerzas externas, para de esta forma obtener al final el sistema de ecuaciones diferenciales que describen el comportamiento del sistema dinámico asociado al brazo robótico [43]:

$$\frac{\partial}{\partial t}\left[\frac{\partial L}{\partial \dot{q}_j}\right] - \frac{\partial L}{\partial q_j} = Q_j = \sum_{i=1}^{n} F_i\cdot\frac{\partial r_i}{\partial q_j} \tag{7.13}$$

Del lado izquierdo está el movimiento del sistema. Y del lado derecho están las fuerzas externas que influyen sobre la trayectoria del sistema. El torque sólo genera fuerzas que están alineadas con la trayectoria del sistema. Entonces los torques van directamente en las

ecuaciones. En el caso de las fricciones, ocurre algo similar puesto que la fricción es proporcional a la velocidad angular, pero negativa. Entonces las fuerzas externas que influyen sobre el sistema quedan expresadas de esta forma:

$$Q_j = \sum_{i=1}^{n} F_i \cdot \frac{\partial r_i}{\partial q_j} \tag{7.14}$$

$$\tau = r \times F \tag{7.15}$$

$$Q_1 = \tau_1 - k_{f1} \cdot \dot{\theta}_1 \tag{7.16}$$

$$Q_2 = \tau_2 - k_{f2} \cdot \dot{\theta}_2 \tag{7.17}$$

Ahora se procede a calcular las ecuaciones generalizadas del movimiento para el brazo robótico virtual:

$$Q_1 = \frac{\partial}{\partial t}\left[\frac{\partial L}{\partial \dot{\theta}_1}\right] - \frac{\partial L}{\partial \theta_1} \tag{7.18}$$

$$Q_2 = \frac{\partial}{\partial t}\left[\frac{\partial L}{\partial \dot{\theta}_2}\right] - \frac{\partial L}{\partial \theta_2} \tag{7.19}$$

Calculando cada uno de los componentes del movimiento y simplificando sus expresiones matemáticas, se obtiene:

$$\frac{\partial L}{\partial \dot{\theta}_1} = \dot{\theta}_1 \cdot [I_1 + m_1 \cdot {l_{g1}}^2 + m_2 \cdot {l_1}^2] + m_2 \cdot l_1 \cdot l_{g2} \cdot \cos(\theta_1 - \theta_2) \cdot \dot{\theta}_2 \tag{7.20}$$

$$\frac{\partial}{\partial t}\left[\frac{\partial L}{\partial \dot{\theta}_1}\right] = [I_1 + m_1 \cdot {l_{g1}}^2 + m_2 \cdot {l_1}^2] \cdot \ddot{\theta}_1 + m_2 \cdot l_1 \cdot l_{g2} \cdot [-\sin(\theta_1 - \theta_2) \cdot (\dot{\theta}_1 - \dot{\theta}_2) \cdot \dot{\theta}_2 + \cos(\theta_1 - \theta_2) \cdot \ddot{\theta}_2] \tag{7.21}$$

$$\frac{\partial L}{\partial \theta_1} = m_1 \cdot g \cdot l_{g1} \sin\theta_1 + m_2 \cdot g \cdot l_1 \cdot \sin\theta_1 - m_2 \cdot l_1 \cdot l_{g2} \cdot \dot{\theta}_1 \cdot \dot{\theta}_2 \cdot \sin(\theta_1 - \theta_2) \tag{7.22}$$

$$\frac{\partial L}{\partial \dot{\theta}_2} = \dot{\theta}_2 \cdot [I_2 + m_2 \cdot {l_{g2}}^2] + m_2 \cdot l_1 \cdot l_{g2} \cdot \dot{\theta}_1 \cdot \cos(\theta_1 - \theta_2) \tag{7.23}$$

$$\frac{\partial}{\partial t}\left[\frac{\partial L}{\partial \dot{\theta}_2}\right] = \ddot{\theta}_2 \cdot [I_2 + m_2 \cdot {l_{g2}}^2] + m_2 \cdot l_1 \cdot l_{g2} \cdot [\ddot{\theta}_1 \cdot \cos(\theta_1 - \theta_2) - \dot{\theta}_1 \cdot \sin(\theta_1 - \theta_2) \cdot (\dot{\theta}_1 - \dot{\theta}_2)] \tag{7.24}$$

$$\frac{\partial L}{\partial \theta_2} = m_2 \cdot g \cdot l_{g2} \cdot \sin\theta_2 + m_2 \cdot l_1 \cdot l_{g2} \cdot \dot{\theta}_1 \cdot \dot{\theta}_2 \cdot \sin(\theta_1 - \theta_2) \tag{7.25}$$

Una vez calculados los componentes del movimiento y las fuerzas externas del sistema, se procede a generar el sistema de ecuaciones diferenciales que describen el sistema dinámico del brazo robótico:

$$Q_1=\frac{\partial}{\partial t}\left[\frac{\partial L}{\partial \dot{\theta}_1}\right]-\frac{\partial L}{\partial \theta_1}=\tau_1-k_{f1}\cdot\dot{\theta}_1 \tag{7.26}$$

$$Q_2=\frac{\partial}{\partial t}\left[\frac{\partial L}{\partial \dot{\theta}_2}\right]-\frac{\partial L}{\partial \theta_2}=\tau_2-k_{f2}\cdot\dot{\theta}_2 \tag{7.27}$$

Reemplazando los valores de la fórmula anterior, se obtiene el sistema de ecuaciones diferenciales en forma matricial que describen el sistema dinámico del brazo robótico [91]:

$$a_{1,1}=I_1+m_1\cdot {l_{g1}}^2+m_2\cdot {l_1}^2 \tag{7.28}$$

$$a_{1,2}=a_{2,1}=m_2\cdot l_1\cdot l_{g2}\cdot\cos(\theta_1-\theta_2) \tag{7.29}$$

$$a_{2,2}=I_2+m_2\cdot {l_{g2}}^2 \tag{7.30}$$

$$b_1=-m_2\cdot l_1\cdot l_{g2}\cdot\sin(\theta_1-\theta_2)\cdot\dot{\theta}_2^{\,2}+g\cdot\sin\theta_1\cdot(m_1\cdot l_{g1}+m_2\cdot l_1)+\tau_1-k_{f1}\cdot\dot{\theta}_1 \tag{7.31}$$

$$b_2=m_2\cdot l_1\cdot l_{g2}\cdot\sin(\theta_1-\theta_2)\cdot\dot{\theta}_1^{\,2}+m_2\cdot g\cdot l_{g2}\cdot\sin\theta_2+\tau_2-k_{f2}\cdot\dot{\theta}_2 \tag{7.32}$$

$$\begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix}\cdot\begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix}=\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{7.33}$$

Parece muy complicado e intrincado, pero este sistema de ecuaciones diferenciales en forma matricial no es tan difícil de resolver de manera numérica. Para resolverlo, asumimos que las únicas variables del sistema son las aceleraciones angulares y el resto de variables permanecen constantes. De esta forma, se puede calcular las aceleraciones angulares por medio de la regla de Cramer del álgebra lineal:

$$\ddot{\theta}_1 = \frac{\begin{vmatrix} b_1 & a_{1,2} \\ b_2 & a_{2,2} \end{vmatrix}}{\begin{vmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{vmatrix}} \tag{7.34}$$

$$\ddot{\theta}_2 = \frac{\begin{vmatrix} a_{1,1} & b_1 \\ a_{2,1} & b_2 \end{vmatrix}}{\begin{vmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{vmatrix}} \tag{7.35}$$

Al principio, se le asigna al sistema unos valores iniciales muy cercanos al cero. No deberían ser cero porque se crearía un equilibrio irrealista. Los ceros tienen propiedades matemáticas especiales. Romper la simetría es necesario para causar un pequeño movimiento que cause desequilibrio.

$$\theta_1 \approx 0 \tag{7.36}$$

$$\omega_1 \approx 0 \tag{7.37}$$

$$\theta_2 \approx 0 \tag{7.38}$$

$$\omega_2 \approx 0 \tag{7.39}$$

Se comienza con esos valores iniciales. Posteriormente en cada iteración, se calcula las aceleraciones angulares con (7.34) y (7.35). Y se calcula los siguientes valores de los ángulos y de las velocidades angulares con las siguientes ecuaciones:

$$\omega = \frac{\partial \theta}{\partial t} \approx \frac{\Delta \theta}{\Delta t} \tag{7.40}$$

$$\theta_{n+1} = \theta_n + \omega \cdot \Delta t \tag{7.41}$$

$$\alpha = \frac{\partial \omega}{\partial t} \approx \frac{\Delta \omega}{\Delta t} \tag{7.42}$$

$$\omega_{n+1} = \omega_n + \alpha \cdot \Delta t \tag{7.43}$$

Al usar las ecuaciones anteriores, se itera constantemente el sistema haciendo pequeñas modificaciones a las variables del sistema. Es necesario calcular en cada iteración el tiempo transcurrido que normalmente está en el orden de milisegundos.

Este brazo robótico tiene muchos parámetros que fueron tuneados manualmente y que aparentemente funcionan bien haciendo que la animación parezca realista. Pero los parámetros pueden ser mejorados. Habría que hacer más experimentos.

La animación es muy fluida y en tiempo real. Esto se debe a la forma que se calcula la solución del sistema de ecuaciones diferenciales que caracteriza el sistema dinámico del brazo robótico. Si se hubiera utilizado otro método numérico como Runge–Kutta o incluso un método simbólico por medio de grupos Lie junto con análisis funcional, la animación no sería muy fluida que digamos por la gran cantidad de cálculos que se necesitare hacer. Entonces es aconsejable extrapolar y utilizar el método de la tesis para resolver otros sistemas dinámicos gobernados por sistemas de ecuaciones diferenciales que son muy complejas e intrincadas. El modelo de spikes neuronales de Hodgkin-Huxley podría beneficiarse del método

de la tesis [42,92].

En este capítulo, la regla de Cramer no sólo resuelve el complicado sistema de ecuaciones diferenciales del brazo robótico; sino que también muestra qué tan causalmente enredados son los sistemas dinámicos basados en Lagrangianos. La regla de Cramer forma divisiones de determinantes de las matrices con las variables del sistema y de esta forma todas las variables se mezclan con todas las variables. Si los sistemas dinámicos de baja dimensionalidad pueden ser muy causalmente enredados, los cerebros son incluso más causalmente enredados y difíciles de resolver a través del reduccionismo. Otro ejemplo de esto es que el profesor Dale Purves admitió que el enredo causal en el problema inverso de la visión es muy difícil de resolver a través de métodos reduccionistas. La evolución del cerebro resolvió esto de una forma muy inteligente: No hay necesidad de calcular inversas como los matemáticos lo hacen. Las redes adaptativas pueden aprender los patrones, contextos y aplicabilidades de las inversas en vez de calcularlas.

El reduccionismo aísla los fenómenos para intentar caracterizarlos con curvas platónicas. Pero la naturaleza no está aislada. Los

campos causales de la naturaleza forman sinergias y antagonismos, creando patrones discontinuos, asimétricos y discretos. Por ejemplo, hay sistemas raros como los lenguajes naturales y las interacciones sociales que son anti-algebraicos porque a menudo carecen de propiedades algebraicas como la totalidad, asociatividad, identidad, invertibilidad y conmutatividad. La teoría de patrones es más general y soluciona los problemas de la matemática platónica. Los patrones y relaciones discontinuos, asimétricos y discretos de la naturaleza están reflejados en la estructura del sistema nervioso: Grupos neuronales (patrones) y axones (relaciones). Los patrones discretos pueden emular los campos continuos, pero no viceversa. Las redes adaptativas pueden representar tanto los sistemas raros como los mapas de las fórmulas simétricas. Pero las fórmulas simétricas no pueden representar los sistemas raros.

Al observar la Figura 7.3, uno debería darse cuenta que los 2 sistemas son fundamentalmente el mismo tipo de problema. Quien no esté de acuerdo, no es capaz de extrapolar, carece de imaginación, carece de creatividad y comete un error de tipo 2 (falso negativo). Un aspecto fundamental de la creatividad y la imaginación es la capacidad de extrapolar algunos aspectos de los conceptos a través

de analogías. Esta capacidad es crucial en dominios versátiles como los modelos de Markov y también las redes neuronales cuyas 10 líneas de pseudocódigo son aplicables a cualquier problema gracias al teorema de aproximación universal de Cybenko. Hay que abrir la mente a un sinnúmero de aplicaciones.

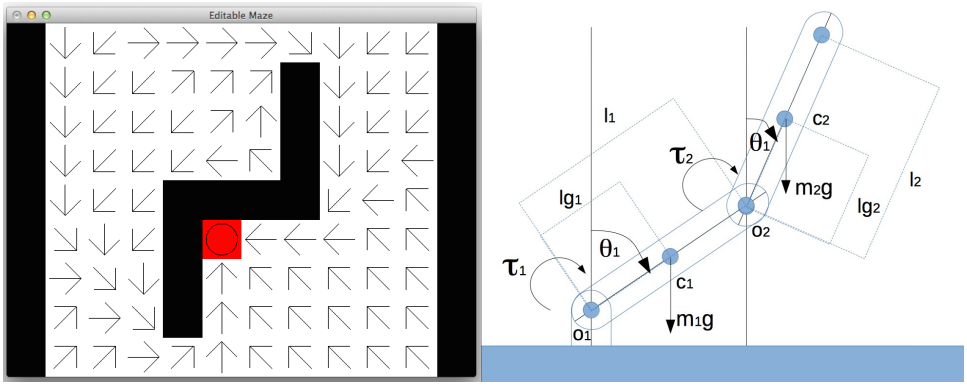


Figura 7.3: Fundamentalmente, son el mismo tipo de problema

En este caso los espacios de estados y acciones del brazo robótico pueden particionarse en bins multidimensionales convirtiéndose en un grid world que puede resolverse con MDP. Pero grid world es muy rectangular e inflexible. Por esto, las redes neuronales son más aplicables a la robótica porque generalizan la lógica difusa y sus estados son regiones no-euclideanas, complejas, sobresalientes y con propiedades similares. Además las redes neuronales recurrentes son naturalmente jerarquizables y reducen la dimensionalidad de los

sistemas. Pero por el momento, las redes neuronales son más lentas que los MDP.

7.3. Uso y Demostración de la Aplicación

Para ver la demostración, es necesario abrir el terminal y entrar al directorio de la tesis. Y luego ejecutar el script robotic-arm.sh. Este script funciona para los sistema operativos Mac y Ubuntu.

A continuación, se ve la aplicación del brazo robótico virtual el cual puede ser controlado por los 2 torques de los motores eléctricos. En la esquina inferior derecha está el espacio de acciones cuyo eje de las X corresponde con el torque 1 del brazo robótico (motor en la primera articulación) y cuyo eje de las Y corresponde con el torque 2 del brazo robótico (motor en la segunda articulación). Con las flechas direccionales del teclado es posible mover el cursor del espacio de acciones y también se puede moverlo arrastrando el botón izquierdo del ratón.

También es posible escribir los caracteres a dibujar en la pizarra. Son

caracteres válidos las letras, los números, el espacio y el guión. Al escribir caracteres por medio del teclado, estos se grafican con un color gris claro que casi ni se ve en el fondo blanco. Pero cuando el extremo del brazo robótico toca sus pixeles, estos pixeles automáticamente se grafican con color negro que contrasta fuertemente con el fondo blanco. Si se modifican los caracteres, habrán algunos pixeles dibujados por el brazo robótico que ya no corresponden con las letras a dibujar. Estos pixeles se marcan con rojo para indicar que son pixeles erróneos que deben ser borrados. Para borrarlos, el extremo del brazo robótico debe pasar por ellos y estos pixeles automáticamente vuelven a ser blancos.

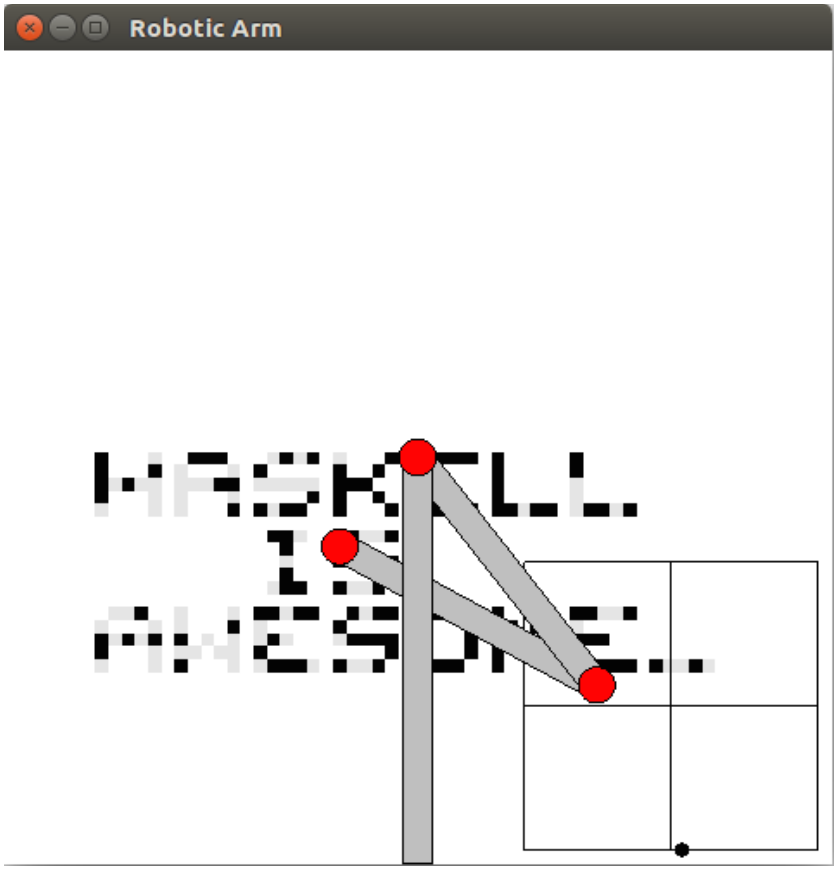


Figura 7.4: El videojuego del brazo robótico

El objetivo del videojuego es cambiar todos los pixeles de color gris claro a negro y todos los pixeles de color rojo a blanco. Cuando se logra este objetivo, el fondo de la pantalla se vuelve amarillo para indicar que se ha ganado el videojuego:

Figura 7.5: El videojuego del brazo robótico cuando se ha ganado

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

1. La computación clásica hace deducción causal (de causas a efectos) basada en modelos causales pre-programados. Mientras que la inteligencia artificial va en el sentido contrario de la causalidad porque hace inducción causal (de efectos a causas) que aprende modelos causales [12].

2. La inducción causal es un aspecto fundamental de la inteligencia. Probablemente es el más importante porque la inducción causal ayuda a predecir. Los 3 algoritmos de la tesis (el reconstructor de imágenes

mentales, el caracterizador evolutivo y el ruteador causal jerárquico) hacen inducción causal y por esto son aplicables a una amplia gama de problemas de inteligencia artificial, no sólo a las aplicaciones descritas en la tesis [12,44].

3. Las matemáticas reduccionistas son un caso específico de las redes complejas de patrones (holismo). Porque las matemáticas reduccionistas emergen del pensamiento que está conformado por redes complejas de patrones. En otras palabras, las redes complejas de patrones podrían ser consideradas como una especie de meta-matemáticas.

4. Las redes neuronales conforman todo el conocimiento, todas las percepciones (incluyendo los sentimientos) y todos los planes. Se forman a lo largo de la vida con la experiencia por el establecimiento de conexiones y asociaciones entre neuronas. Por eso las personas piensan diferente. Pero también hay similitudes de pensamientos entre las personas gracias a que se percibe un mismo entorno [23].

5. La memoria está representada por códigos relacionales. Uno percibe patrones y estos tienen sentido y significado por las relaciones entre

sus partes. El todo, el significado de los patrones, lo definen las relaciones entre las partes, y no es reducible a las partes en sí. El todo es mucho más que la suma de las partes por separado. Son las relaciones entre las partes que le dan un valor agregado al todo [59].

6. Las memorias primarias, tanto sensoriales como motoras, son modulares. Pero a medida que se sube en la jerarquía cortical, la memoria cortical se va haciendo más interconexa, más asociativa, más compleja, más amplia, más difusa y más robusta [23].

7. Las redes neuronales profundas (deep learning) pueden ser vistas como campos causales traslapados que reconstruyen imágenes mentales porque las neuronas artificiales representan regiones de activación e inhibición que son delimitadas por hiperplanos. Estas regiones del espacio mental pueden sumarse, traslaparse y componerse para formar características complejas (regiones complejas) para el reconocimiento de patrones complejos. Las redes neuronales aprenden y se entrenan con la técnica de descenso del gradiente, que son su método de auto-organización y meta-programación. Además, el deep learning puede usarse para reducir la dimensionalidad del reinforcement learning. El deep learning puede

hacer reinforcement learning a nivel de patrones sobresalientes cuyos estados serían regiones complejas que generalizan la lógica difusa. Además los procesos de decisión markovianos son difíciles de jerarquizar; mientras que las redes neuronales recurrentes son naturalmente jerarquizables. Pero primero se necesita acelerar su tiempo algorítmico. Un sólo paradigma, el deep learning, cubre los 3 paradigmas de la tesis: Reconstrucción de imágenes mentales, caracterización evolutiva y ruteo causal. Por esto y por muchas razones más, el deep learning tiene un futuro prometedor.

RECOMENDACIONES

1. Se debería usar ampliamente algoritmos de inducción causal en los sistemas de inteligencia artificial. De esta forma se crea inteligencia artificial verdadera capaz de predecir y entender el entorno [6].

2. Se debería limitar el uso de modelos causales pre-programados en la programación de inteligencia artificial: Deducción causal que va desde las causas (modelos) hacia los efectos (comportamiento). Se debería programar a la máquina para que esta sea capaz de aprender (machine learning) los modelos causales a través de la observación y exploración: Inducción causal que va desde los efectos (comportamiento observado) hacia las causas (modelos) [19].

3. Asimismo los programadores deberían evitar en lo posible hacer el proceso de reducción y programar en la máquina los modelos que ellos crearon porque de esta forma no dejan que la máquina sóla encuentre sus modelos. La verdadera inteligencia artificial consiste en hacer que la máquina sea capaz de encontrar sus modelos por sí misma sin ayuda de humanos. Para esto las redes complejas de patrones y el holismo son necesarios. El proceso de reducción va

desde las redes complejas hacia los modelos relevantes para cada problema. Se escoge lo más importante de las redes que esté relacionado con el problema a resolver. Es un algoritmo para encontrar patrones sobresalientes, los cuales son la esencia de algo y sin ellos algo no funciona de la misma manera. En otras palabras, aplicar la navaja de Ockham [3,19].

4. Las redes complejas de patrones son una extensión de las matemáticas reduccionistas. No deberían verse como un sustituto o un competidor de las matemáticas sino más bien como un complemento. Es muy común observar debates acalorados entre holistas y reduccionistas. No debería haber antagonismos de este tipo cuando en realidad son posturas complementarias.

5. Se debería flexibilizar el reconocimiento de patrones para que las máquinas encuentren similitudes y correspondencias analógicas entre diferentes objetos. De esta forma, las redes cognitivas se vuelven hiper-completas, pueden extrapolar relaciones y entender mejor el entorno a pesar de no tener mucha experiencia. Aprender todo lo que hay en el mundo es imposible; por esto, las extrapolaciones y similitudes son necesarias. Como resultado de este proceso, los

errores de tipo 1 (falsos positivos) se acentúan. Pero esto es preferible a la rigidez e inflexibilidad de las matemáticas y computadores actuales que comenten demasiados errores de tipo 2 (falsos negativos) [28].

6. Parafraseando un poco al Dr. Joaquín Fuster, uno no debería enfocarse demasiado en estudiar la estructura de la neurona o peor aún en estructuras sub-neuronales para saber cómo funciona la inteligencia. Porque la inteligencia funciona con códigos relacionales a nivel de la red. Y es irreducible a las partes por separado. Además un grupo celular puede ser parte de muchas redes. Son las redes que le dan semántica a los grupos celulares y no al revés. Pretender entender la inteligencia estudiando la estructura de la neurona sería como pretender entender el significado del lenguaje escrito estudiando la composición química de la tinta. No se entenderá nunca porque el lenguaje es relacional y porque la tinta no es relevante ni es el nivel de complejidad apropiado para entender el lenguaje escrito. Asimismo para entender la inteligencia, uno debería estudiar el nivel de complejidad apropiado: La neurociencia de sistemas (systems neuroscience) que estudia como interactúan las redes cognitivas [93].

7. Estudiar la teoría de los cognits del Dr. Joaquín Fuster que explica y predice una gran cantidad de fenómenos mentales que ocurren en el cerebro [58,59,23,94].

BIBLIOGRAFÍA

[1] Resnick, R., Halliday, D., y Krane, K., Física, CECSA, 1996

[2] Russell, S., and Norvig, P., Artificial Intelligence: A Modern Approach, Prentice Hall, 2009

[3] Anderson, M., Artificial Intuition, http://artificial-intuition.com/, 2007

[4] Corning, P., Nature's Magic: Synergy in Evolution and the Fate of Humankind, Cambridge University Press, 2003

[5] Kelso, S., Dynamic Patterns: The Self-Organization of Brain and Behavior, A Bradford Book, 1995

[6] Hawkins, J., and Blakeslee, S., On Intelligence, Times Books, 2004

[7] Leahey, T., y Harris, R., Aprendizaje y Cognición, Prentice Hall, 1998

[8] Mooney, B., Human Connectome, http://bill-mooney.blogspot.com/2012/10/human-connectome.html, 2014

[9] The Haskell Team, The Haskell Programming Language, http://www.haskell.org/haskellwiki/Haskell, 2013

[10] Bellman, R., Dynamic Programming, Dover Publications, 2003

[11] Martín, B., y Sanz, A., Redes Neuronales y Sistemas Difusos, Alfaomega Ra-Ma, 2002

[12] Legg, S., Machine Super Intelligence, Lulu, 2008

[13] Eliasmith, C., and Anderson, C., Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems, A Bradford

Book, 2004

[14] Bear, M., Connors, B., and Paradiso, M., Neuroscience: Exploring the Brain, Lippincott Williams and Wilkins, 2006

[15] Koller, D., and Friedman, N., Probabilistic Graphical Models: Principles and Techniques, The MIT Press, 2009

[16] Marlow, S., Parallel and Concurrent Programming in Haskell, http://chimera.labs.oreilly.com/books/1230000000929/index.html, 2013

[17] Lipovača, M., Learn You a Haskell for Great Good!, http://learnyouahaskell.com/, 2011

[18] FP Complete, FP Complete, https://www.fpcomplete.com/, 2014

[19] Anderson, M., Bizarre Domains, http://artificial-intuition.com/bizarre.html, 2007

[20] Bishop, C., Pattern Recognition and Machine Learning, Springer, 2007

[21] Wissner-Gross, A. D., and Freer, C. E., Causal Entropic Forces, http://www.alexwg.org/publications/PhysRevLett_110-168702.pdf, 2013

[22] Thagard, P., Coherence in Thought and Action, A Bradford Book, 2002

[23] Fuster, J., Cortex and Mind: Unifying Cognition, Oxford University Press, 2003

[24] Pothos, M., and Wills A., Formal Approaches in Categorization, Cambridge University Press, 2011

[25] Foley, J., van Dam, A., Feiner, S., Hughes, J., y Phillips, R., Introducción

a la Graficación por Computadora, Addison-Wesley Iberoamericana, 1996

[26] Edelman, S., Representation and Recognition in Vision, A Bradford Book, 1999

[27] Murphy, K., A Brief Introduction to Graphical Models and Bayesian Networks, http://www.cs.ubc.ca/~murphyk/Bayes/bnintro.html, 1998

[28] Hofstadter, D., and Sander, E., Surfaces and Essences: Analogy as the Fuel and Fire of Thinking, Basic Books, 2013

[29] Grossman, S., Álgebra Lineal, McGraw-Hill, 1996

[30] Friston, K., The free-energy principle: a unified brain theory?, http://www.fil.ion.ucl.ac.uk/~karl/The%20free-energy%20principle%20A%20unified%20brain%20theory.pdf, 2010

[31] Doya, K., Ishii, S., Pouget, A., and Rao, R., Bayesian Brain: Probabilistic Approaches to Neural Coding, The MIT Press, 2006

[32] Pearl, J., Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann Pub, 1998

[33] Edelman, G., and Mountcastle, V., The Mindful Brain: Cortical Organization and the Group-Selective Theory of Higher Brain Function, The MIT Press, 1982

[34] Edelman, G., Neural Darwinism: The Theory Of Neuronal Group Selection, Basic Books, 1987

[35] Martin, M., The Mocking Memes: A Basis for Automated Intelligence, AuthorHouse, 2006

[36] Stewart, J., Cálculo, Grupo Editorial Iberoamérica, 1994

[37] Freeman, J., y Skapura, D., Redes Neuronales. Algoritmos, aplicaciones y técnicas de programación, Addison-Wesley/Diaz de Santos, 1991

[38] Klein, P., Coding the Matrix: Linear Algebra through Computer Science Applications, Newtonian Press, 2013

[39] Sato, Y., To Reproduce in Artificial, http://yuki-sato.com/wordpress/2014/09/01/%E4%BA%BA%E5%B7%A5%E3%81%A7%E5%86%8D%E7%8F%BE%E3%81%99%E3%82%8B%E3%81%AB%E3%81%AF/, 2014

[40] Unknown, Boy shows why stereopsis is important, http://2.bp.blogspot.com/-o-CMfpw5oLg/U3fispbn63I/AAAAAAAAKZw/o7LJfRqsd1w/s1600/07.jpg, 2014

[41] Groh, J., Making Space: How the Brain Knows Where Things Are, Belknap Press, 2014

[42] Boyce, W., y DiPrima, R., Ecuaciones Diferenciales y Problemas con Valores en la Frontera, LIMUSA Noriega Editores, 1996

[43] Anonym, Lagrangian mechanics, http://en.wikipedia.org/wiki/Lagrangian_mechanics, 2014

[44] Hutter, M., Universal Artificial Intelligence: Sequential Decisions Based On Algorithmic Probability, Springer, 2004

[45] Murphy, K., Machine Learning: A Probabilistic Perspective, The MIT Press, 2012

[46] Rolls, E., Memory, Attention, and Decision-Making: A unifying computational neuroscience approach, Oxford University Press, 2007

[47] Howard, R., Dynamic programming and Markov processes, The MIT Press, 1966

[48] Kurzweil, R., How to Create a Mind: The Secret of Human Thought Revealed, Penguin Books, 2012

[49] Goleman, D., Kaufman, P., y Ray, M., El Espíritu Creativo, Javier Vergara Editor – Grupo Zeta, 1992

[50] Dawkins, R., The Greatest Show on Earth: The Evidence for Evolution, Free Press, 2009

[51] Hemmen, L., and Sejnowski, T., 23 Problems in Systems Neuroscience, Oxford University Press, 2005

[52] Pearl, J., Causality: Models, Reasoning, and Inference, Cambridge University Press, 2009

[53] Mitchell, T., Machine Learning, McGraw-Hill, 1997

[54] Llinás, R., I of the Vortex: From Neurons to Self, The MIT Press, 2001

[55] Martin, M., The Laughing Genes: A Scientific Perspective on Ethics and

Morality, AuthorHouse, 2005

[56] Rolls, E., Emotion and Decision-making Explained, Oxford University Press, 2013

[57] Striedter, G., Principles of Brain Evolution, Sinauer Associates, 2004

[58] Fuster, J., The Prefrontal Cortex, Academic Press, 2008

[59] Fuster, J., Memory in the Cerebral Cortex: An Empirical Approach to Neural Networks in the Human and Nonhuman Primate, The MIT Press, 1999

[60] Churchland, P., Neurophilosophy: Toward a Unified Science of the Mind-Brain, The MIT Press, 1986

[61] Gärdenfors, P., Conceptual Spaces: The Geometry of Thought, Bradford Books, 2004

[62] Carter, R., The Human Brain Book, DK ADULT, 2009

[63] Sutton, R., and Barto, A., Reinforcement Learning: An Introduction, A Bradford Book, 1998

[64] Hebb, D., The Organization of Behavior: A Neuropsychological Theory, Psychology Press, 2002

[65] Hayek, F., The Sensory Order: An Inquiry into the Foundations of Theoretical Psychology, University Of Chicago Press, 1999

[66] Ramachandran, V. S., and Blakeslee, S., Phantoms in the Brain: Probing the Mysteries of the Human Mind, William Morrow Paperbacks,

1999

[67] Blakeslee, S., and Blakeslee, M., The Body Has a Mind of Its Own: How Body Maps in Your Brain Help You Do (Almost) Everything Better, Random House Trade Paperbacks, 2008

[68] Fuster, J., Cortical memory, http://www.scholarpedia.org/article/Cortical_memory, 2007

[69] Biomedical Computation Review, Cortical Columns, http://biomedicalcomputationreview.org/sites/default/files/u38/cells_in_cortex_p12-spr_09.jpg, 2014

[70] Ringwood, J., and Galvin, G., Discrete Time Recurrent Neural Network, http://annet.eeng.nuim.ie/intro/course/chpt5/discrete.shtml, 2008

[71] ElHefnawi, M., and Mysara, M., Recurrent Neural Networks and Soft Computing, InTech, 2012

[72] Rosi, D., Study Artificial Intelligence, http://www.aistudy.co.kr/neural/images/%ED%99%89%ED%95%84%EB%93%9C%20%EB%84%A4%ED%8A%B8%EC%9B%8C%ED%81%AC.gif, 2014

[73] Nokia, Neural Network Image, http://developer.nokia.com/community/wiki/images/a/a1/NeuralNetwork.png, 2014

[74] Beal, J., Human brain right dissected lateral view description,

http://en.wikipedia.org/wiki/White_matter#/media/File:Human_brain_right_dissected_lateral_view_description.JPG, 2005

[75] Lesmoir-Gordon, N., Rood, W., and Edney, R., Introducing Fractal Geometry, Icon Books, 2006

[76] Kurzweil, R., The Age of Spiritual Machines: When Computers Exceed Human Intelligence, Penguin Books, 1999

[77] Dowling J., Creating Mind: How the Brain Works, WW Norton, 1998

[78] Colyer, S., Fractals and Human Biology, http://tetrahedral.blogspot.com/2011/04/fractals-and-human-biology.html, 2011

[79] Matthews, J., How to Knit a Sweater that Fits - Part 3, http://blog.knittingatlarge.com/2012/12/how-to-knit-sweater-that-fits-part-3.html, 2012

[80] Komo Ventures, Biometrics, http://www.komoventures.com/styled-2/styled-11/biometrics.html, 2014

[81] Hauberg, S., Freifeld, O., and Black, M., A Geometric take on Metric Learning, http://machinelearning.wustl.edu/mlpapers/paper_files/NIPS2012_0997.pdf, 2012

[82] Pavel, Evolution and its Problems - The Origin of Life, http://www.shodalap.org/pavel/14679/, 2012

[83] Ibarlucea, B., flue virus through an electron microscope, http://blogs.wayne.edu/biancadesign/2013/01/14/pattern-in-nature-and-design/zygote-media-group-anatomy/, 2013

[84] Devenport, G., Cargill turkey recall: Information and tips for playing it safe, http://www.examiner.com/article/cargill-turkey-recall-information-and-tips-for-playing-it-safe, 2011

[85] Lovest Thou Me, An Object Lesson from the Lowly Sponge, http://lovestthoume.com/Preparation/AnObjectLesson.html, 2013

[86] Simpson, K., The Story of C. Elegans, http://blog.eyewire.org/the-story-of-c-elegans/, 2014

[87] Barracuda, Clown fish, http://www.pxleyes.com/photography-picture/4d18f4e96b509/Clown-fish.html, 2014

[88] Eliasmith, C., How to Build a Brain: A Neural Architecture for Biological Cognition, Oxford University Press, 2013

[89] Thrun, S., Burgard, W., and Fox, D., Probabilistic Robotics, The MIT Press, 2005

[90] Kurzweil, R., Are We Spiritual Machines?: Ray Kurzweil vs. the Critics of Strong A.I., Discovery Institute, 2001

[91] Kreyszig, E., Matemáticas Avanzadas para Ingeniería, LIMUSA Noriega Editores, 1996

[92] Burden, R., and Faires, J., Análisis Numérico, Grupo Editorial

Iberoamérica, 1996

[93] Fuster, J., y Punset, E., El alma está en la red del cerebro, http://www.rtve.es/television/20111111/alma-esta-red-del-cerebro/474693.shtml, 2012

[94] Fuster, J., The Neuroscience of Freedom and Creativity: Our Predictive Brain, Cambridge University Press, 2013

[95] Plantz, A., C Manual de bolsillo, Addison-Wesley Iberoamericana, 1990

ANEXOS

Anexo A: Instalación del Software en Mac

Para hacer funcionar el código de la tesis en los computadores Mac, es necesario instalar Haskell para Mac. Primero se visita el sitio web de Haskell:

http://www.haskell.org/haskellwiki/Haskell

Y luego se hace click en “Download Haskell”. Debería aparecer la siguiente dirección:

http://www.haskell.org/platform/

Para luego hacer click en la opción “Mac”. Debería aparecer la siguiente dirección:

Haskell Platform for Mac OS X

http://www.haskell.org/platform/mac.html

Luego se hace click en “Haskell Platform for Mac OS X, 64 bit” y debería comenzar a descargar el archivo “Haskell Platform 64bit.pkg”.

Antes de instalar Haskell, es necesario cumplir con un pre-requisito. Hay que instalar “Command Line Developer Tools”. Esto se puede encontrar en la dirección:

https://developer.apple.com/opensource/

Para entrar ahí, es necesario tener cuenta de Apple developer y hacer login. Una vez adentro del sistema web, se hace click en “Command Line Tools” y aparecerá la dirección:

https://developer.apple.com/downloads/index.action?=command%20line%20tools

Se selecciona las “Command Line Tools for Xcode” correspondientes al sistema operativo instalado. Se baja el archivo “command_line_tools_for_osx.dmg” y se lo instala. Una vez instalado, ahora sí se puede instalar Haskell con el archivo “Haskell Platform 64bit.pkg”.

En la tesis, algunos programas leen imágenes y otros programas incluso graban imágenes capturadas de la pantalla. Entonces hay que instalar el

módulo JuicyPixels (Codec.Picture). Para esto, se digita los siguientes comandos en el terminal:

```
cabal update
cabal install JuicyPixels
```

Y listo. Ya se puede utilizar el software de la tesis. Para utilizarlo, se abre el terminal. Se entra al directorio de la tesis. Y se ejecuta algunos de los siguientes scripts: editable-maze.sh, haddock-script.sh, inverted-pendulum.sh, minesweeper-solver-for-mac.sh, minesweeper-solver-for-ubuntu.sh, minesweeper.sh, page-of-numbers.sh, remove-executable-files.sh, robotic-arm.sh, vision-app-for-mac.sh o vision-app-for-ubuntu.sh.

Hay que tener presente que en la plataforma Mac, se hace uso de las librerías escritas en lenguaje C para la captura de la pantalla y para la generación de eventos de mouse. Estas librerías ya vienen instaladas con el sistema operativo OS X y por esto no es necesario instalar librerías adicionales. En el caso de Ubuntu, sí es necesario instalar librerías adicionales. Si se quiere saber como se utilizan estas librerías como código nativo en Haskell, hay que revisar el código de los módulos JCKuri.Misc.HScreenCapture y JCKuri.Misc.HRobot [95].

El código fuente del software de la tesis fue editado en jEdit. Para ver el código fuente hay que entrar al directorio de la tesis y luego hay que entrar al subdirectorio JCKuri. Es recomendable ver el código en jEdit. La dirección de jEdit es la siguiente:

http://www.jedit.org/

Anexo B: Instalación del Software en Ubuntu

Para hacer funcionar el código de la tesis en los computadores con Ubuntu, es necesario instalar Haskell para Ubuntu. Primero se visita el sitio web de Haskell:

http://www.haskell.org/haskellwiki/Haskell

Y luego se hace click en “Download Haskell”. Debería aparecer la siguiente dirección:

http://www.haskell.org/platform/

Y luego se hace click en “Linux”. Debería aparecer la siguiente dirección:

http://www.haskell.org/platform/linux.html

Ahí hay varios links para bajar diversas versiones de Haskell para diversos tipos de Linux, incluyendo Ubuntu. Se mostró estos pasos para familiarizarse con estas opciones. Pero hay un método mucho más sencillo para instalar Haskell para Ubuntu.

Para instalar fácilmente Haskell para Ubuntu, se abre el terminal y se ejecuta el siguiente comando:

```
sudo apt-get install haskell-platform
```

Así de simple y automático. Pero toma su tiempo.

En la tesis, algunos programas leen imágenes y otros programas incluso graban imágenes capturadas de la pantalla. Entonces hay que instalar el módulo JuicyPixels (Codec.Picture). Para esto, se digita los siguientes comandos en el terminal:

```
cabal update
cabal install JuicyPixels
```

Hay que tener presente que en la plataforma Ubuntu, se hace uso de unas librerías escritas en lenguaje C para la captura de la pantalla y para la generación de eventos de mouse. Estas librerías no vienen instaladas con el sistema operativo Ubuntu y por esto es necesario instalar librerías adicionales. Si se quiere saber como se utilizan estas librerías como código nativo en Haskell, hay que revisar el código de los módulos

JCKuri.Misc.HScreenCapture y JCKuri.Misc.HRobot [95].

Para que Haskell pueda capturar la pantalla en Ubuntu, es necesario instalar las librerías GTK y PixBuf (las versiones developer). Para instalarlas, se abre el terminal y se ejecuta los siguientes comandos:

```
sudo apt-get install libgdk-pixbuf2.0-dev
sudo apt-get install libgtk2.0-dev
```

Para que Haskell pueda generar eventos autónomos de mouse en Ubuntu, es necesario instalar la extensión XTest de la librería X11 (la versión developer). Para instalarla, se abre el terminal y se ejecuta el siguiente comando:

```
sudo apt-get install libxtst-dev
```

Y listo. Ya se puede utilizar el software de la tesis. Para utilizarlo, se abre el terminal. Se entra al directorio de la tesis. Y se ejecuta algunos de los siguientes scripts: editable-maze.sh, haddock-script.sh, inverted-pendulum.sh, minesweeper-solver-for-mac.sh, minesweeper-solver-for-ubuntu.sh, minesweeper.sh, page-of-numbers.sh, remove-executable-files.sh,

robotic-arm.sh, vision-app-for-mac.sh o vision-app-for-ubuntu.sh.

El código fuente del software de la tesis fue editado en jEdit. Para ver el código fuente hay que entrar al directorio de la tesis y luego hay que entrar al subdirectorio JCKuri. Es recomendable ver el código en jEdit. La dirección de jEdit es la siguiente:

http://www.jedit.org/