Diplomado de especialización de desarrollo de aplicaciones con Inteligencia Artificial

Optimización industrial con Computación Evolutiva

Sesión 7
Neuroevolución

Dr. Soledad Espezua. Ll.
sespezua@pucp.edu.pe

Dr. Edwin Villanueva T.
evillatal@pucp.edu.pe

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

PUCP

ia.inf.pucp.edu.pe

Agenda

- Introducción
- Neuroevolución:
  - Conceptos
  - Características
  - Topologías
- Desafíos en TWEANNs
- NEAT
  - Características
  - Importancia
  - Funcionamiento
  - Aplicaciones
  - Alg. relacionados
- Bibliografía

Introducción

Aprendizaje y evolución

- El aprendizaje y la evolución, desde un punto de vista biológico, son dos formas fundamentales de adaptación.

Introducción

Aprendizaje y evolución

- La investigación actual de RNs se centra principalmente en *Deep learning*, y aprendizaje por refuerzo.

Millares de imágenes etiquetadas como "**Gato**" o "**Perro**"

Una arquitectura *deep learning* tiene que ser entrenado con una gran cantidad de imágenes, para ser capaz de distinguir, por ejemplo, un perro de lo que no es un perro.

Introducción

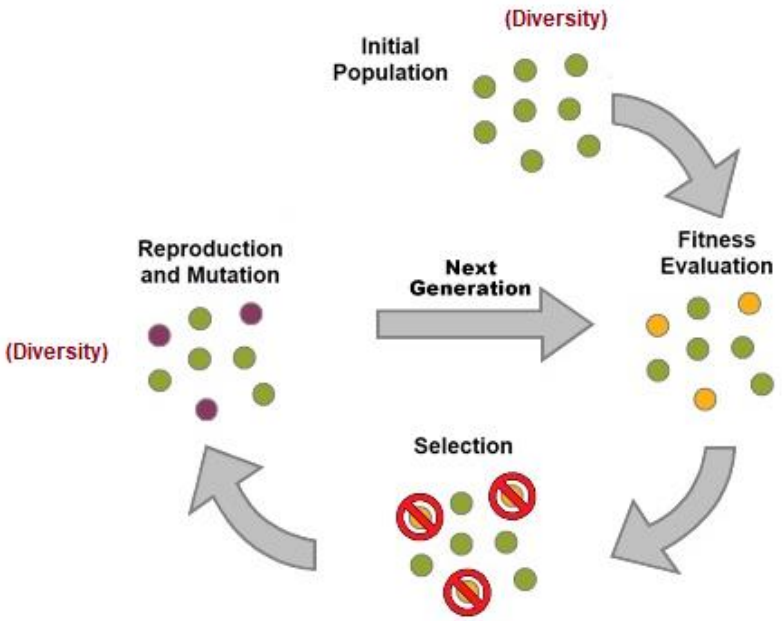
Aprendizaje y evolución

- Los humanos, solamente necesitamos algunas fotos para aprender a qué se parece un perro, pero *deep learning* requiere miles de fotos. Esto, porque los computadores aprenden de forma muy distinta a los humanos.

¿Habrá algo que la naturaleza nos pueda enseñar
para entrenar nuestro modelo de forma más eficiente?

Introducción

- En años recientes ha habido un gran interés en combinar dos conceptos: algoritmos evolucionarios (EAs – *Evolucionary Algorithms*) y redes neuronales artificiales (ANNs – *Artificial Neural Nerworks*), inspirados en el hecho de que los cerebros en sí son el producto de un proceso evolutivo.



EAs
Evolución y Selección natural

RNs:
Sistema nervioso central

Neuroevolución

Neural Network + Evolution = Neuroevolution

La neuroevolución es una técnica de la IA que intenta desencadenar un proceso evolutivo similar al que produjo nuestro cerebro, excepto que esto lo hace dentro de una computadora. Para esto busca formas de evolucionar redes neuronales a través de algoritmos evolucionarios.

La idea central, es producir un modelo que sea lo suficientemente desarrollado como para solucionar un problema, sin que se le muestren tantos ejemplos.
Esta es una idea casi opuesta a la forma en que se entrena un modelo con grandes cantidades de ejemplos como se hace con *deep learning*.

Neuroevolución – Características

- Permite el aprendizaje de parámetros (por ejemplo, funciones de activación), pesos, topologías, reglas e incluso los algoritmos para aprender ellos mismos.
- Difiere de *deep learning* (y *deep reinforcement learning*) al mantener una población de soluciones durante la búsqueda, lo que permite la exploración extrema y la paralelización masiva.
- Mientras que *deep learning* tradicionalmente se enfoca en programar una RN para aprender, la preocupación en neuroevolución se enfoca en modelar el origen de la arquitectura de la RN en sí. Esto puede abarcar desde qué está conectado con qué, los pesos de esas conexiones y (a veces) cómo se permite que esas conexiones cambien.

Neuroevolución- Topologías

Neuroevolución de topología fija

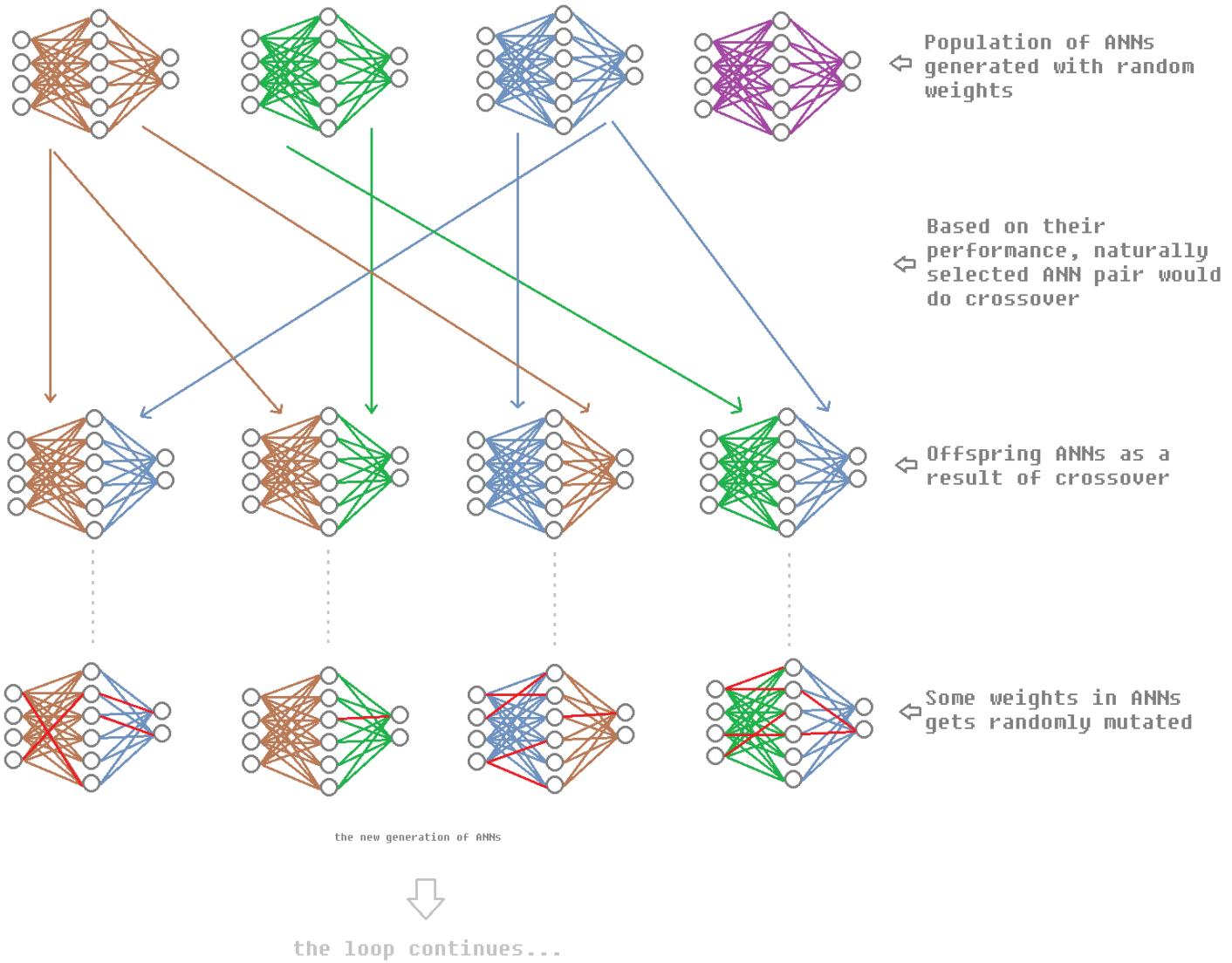
Los pesos de la RN varían, pero la topología es mantenida constante.

El objetivo es optimizar los pesos de conexión que determinan la funcionalidad de una red.

- ✓ Se explora el espacio de los pesos (w) a través del cruce y mutación de vectores de pesos de la red.
- ✓ La evolución decide los pesos, p. ej en lugar de usar el gradiente descendiente.

Neuroevolución- Topologías

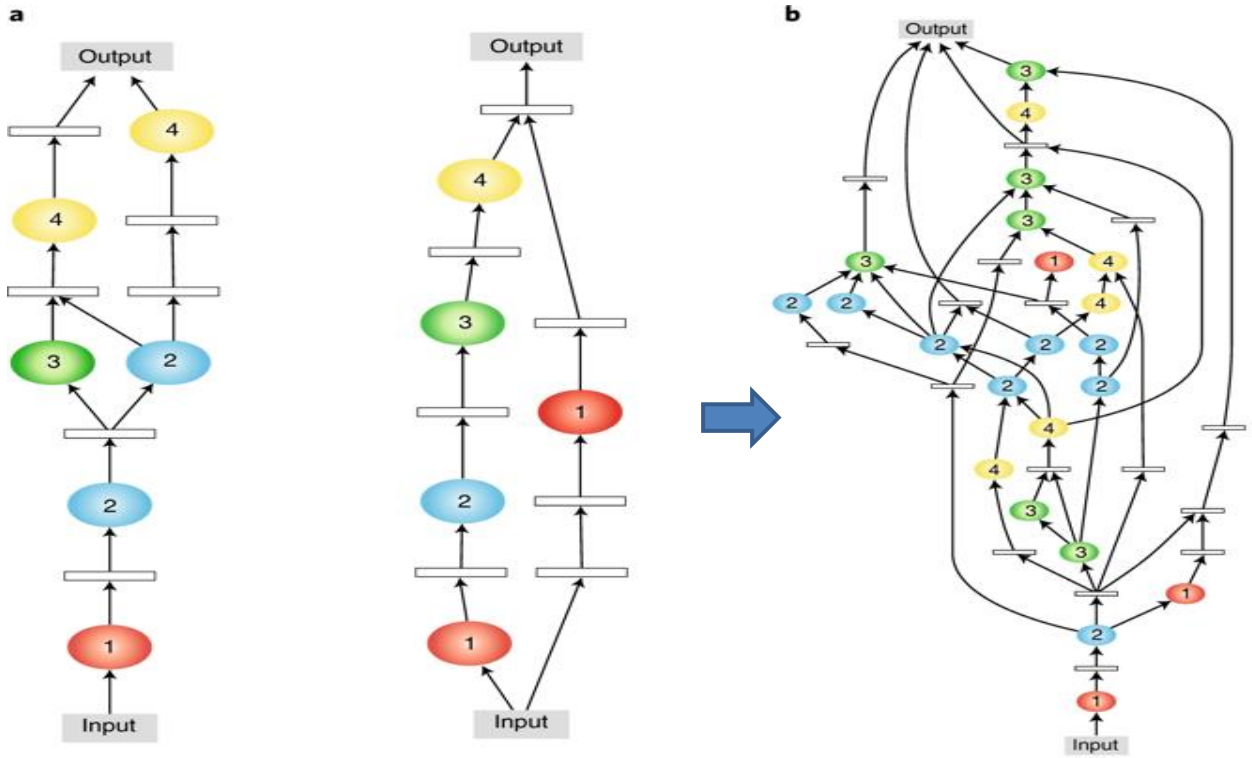
Neuroevolución de topología fija



Neuroevolución- Topologías

***Topology and Weight Evolving Artificial Neural Networks* (TWEANNs)**

Son métodos que abarcan una variedad de ideas acerca de cómo se deben implementar las RNs que evolucionan la topología y los pesos.



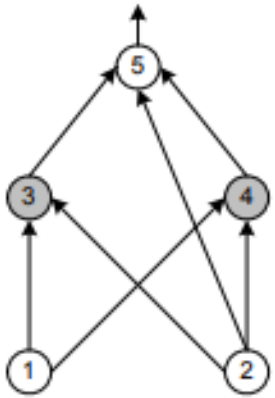
Desafíos en TWEANNs

1. ¿Cómo codificar una red neuronal?
2. ¿Cómo realizar cruzamiento entre estructuras con diferentes topologías para crear descendientes?
3. ¿Cómo crear convenciones (topologías de utilidad) competentes?
4. ¿Cómo proteger las innovaciones (topologías más complejas) de morir antes de permitirles suficiente tiempo para que sus pesos sean optimizados y así revelen su verdadero potencial?

Desafíos en TWEANNs

¿Cómo codificar una red neuronal?

- Directamente: Se especifica explícitamente cada neurona y conexión en la topología de red (fenotipo).
  - Codificación binaria: Una cadena de bits representa la matriz conectiva de las redes
  - Codificación por grafos: El número de nodos en un grafo bidimensional, representa la versión gráfica del genoma.



Codificación por grafos

	1	2	3	4	5
1	0	0	1	1	0
2	0	0	1	1	1
3	0	0	0	0	1
4	0	0	0	0	1
5	0	0	0	0	0

matriz de conectividad

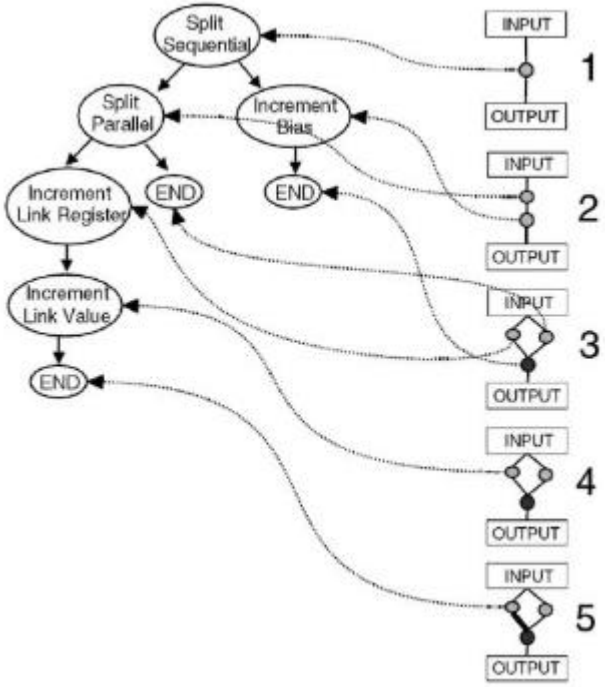
00110 00111 00001 00001 00000

Codificación binaria, utilizando 4 bits.

Desafíos en TWEANNs

¿Cómo codificar una red neuronal?

- Indirectamente: El genoma especifica indirectamente cómo se debería construir una red.
  - Esto permite una representación más compacta que la codificación directa, porque cada conexión y nodo no son especificados en el genoma, pero pueden derivarse de él.



Una forma de crear codificación indirecta es *Cellular Encoding* (CE). Un método que simula el crecimiento celular. Los genomas son programas escritos en un lenguaje especializado de transformaciones gráficas motivadas por divisiones celulares. Así, de una división celular pueden resultar diferentes tipos de conexiones, ya que existen varios tipos de divisiones celulares.

Desafíos en TWEANNs

¿Cómo realizar cruzamiento?

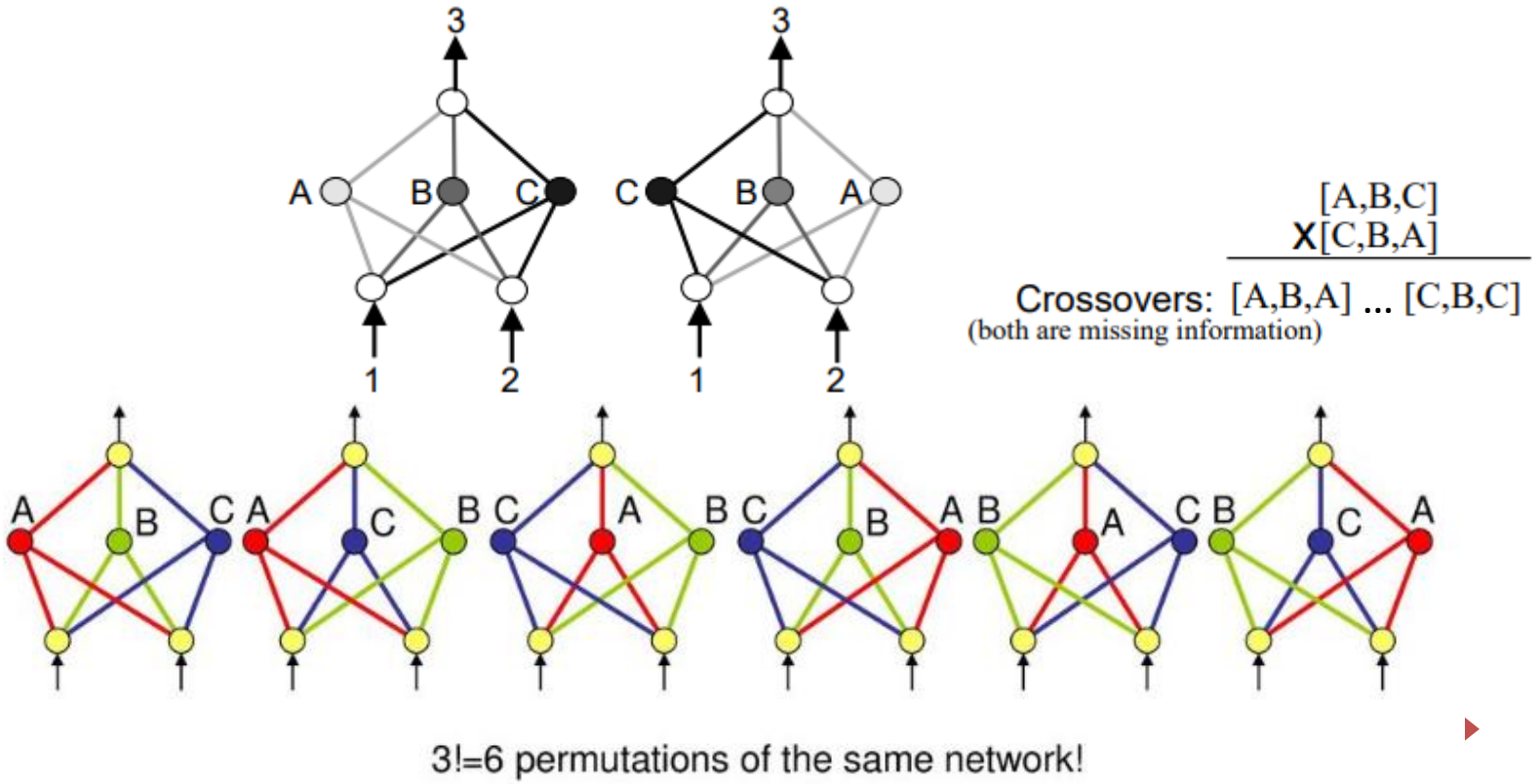
- Debido a que el cruce de redes con diferentes topologías a menudo puede conducir a una pérdida de funcionalidad, muchas investigadores renunciaron al cruzamiento. Sin embargo, investigaciones posteriores demostraron que en TWEANN no se necesita de un crossover para funcionar, solo alineamiento.

Desafíos en TWEANNs

¿Cómo crear Convenciones competentes?

- El problema de crear convenciones competentes, surge cuando hay más de una forma de representar la información en un fenotipo.

  Si, dos RNs, ambas codifican la misma función, no importa cómo se crucen, sus hijos siempre carecerán de información.



Desafíos en TWEANNs

¿Cómo proteger las innovaciones?

Las innovaciones (nuevas topologías) generalmente implican agregar una estructura más grande y con más conexiones a la red, mediante mutación.

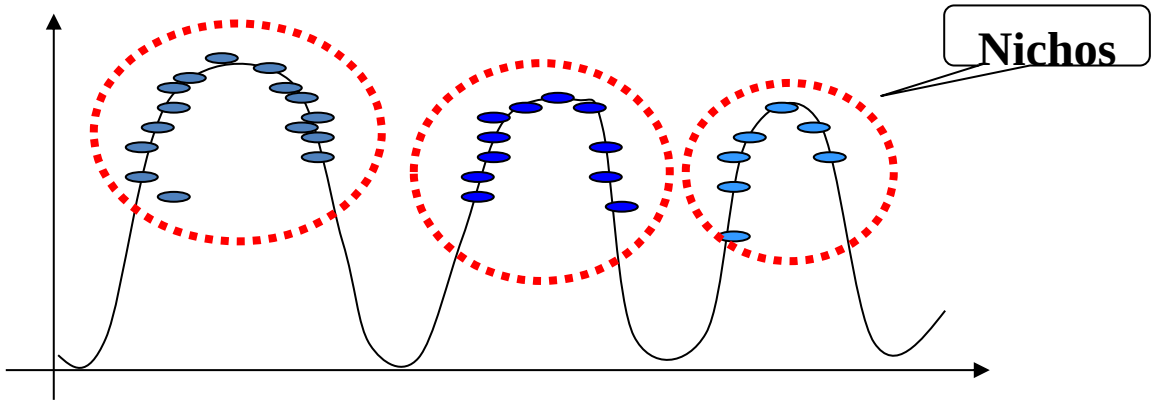
- Agregar una nueva conexión, implica:
  - Puede reducir el *fitness* de la red
  - Necesitará algunas generaciones más para optimizar la nueva estructura
- Agregar una nueva estructura , implica:
  - Las redes más grandes requieren más tiempo para ser optimizadas y no pueden competir con las más pequeñas
  - Debido a la pérdida inicial de *fitness* causada por la nueva estructura, es poco probable que la innovación sobreviva en la población el tiempo suficiente para ser optimizada.
- Una forma de proteger las innovaciones es mediante **especiación.**

Desafíos en TWEANNs

- Especiación (*niching*): En la naturaleza, es el proceso por el cual las especies se crean.

  La especiación es una técnica efectiva para evitar competencia y cruzamiento con redes muy diferentes.

  El objetivo es que redes con estructuras innovadoras se aíslen con sus propias especies (redes similares), ganando la oportunidad de optimizar sus estructuras antes de tener que competir con la población en general.



Innovación está implícitamente protegida dentro de un nicho

Desafíos en TWEANNs

Poblaciones iniciales e innovación topológica

- Las poblaciones iniciales son una colección de topologías aleatorias que garantizan la diversidad topológica desde el principio.
  - Inconvenientes:
    - Bajo muchos de los esquemas de codificación directa, existe la posibilidad de que una red no tenga una ruta desde cada una de sus entradas a sus salidas. Tales redes inviables toman tiempo para eliminarse de la población.
    - Comenzar con topologías aleatorias no conduce a encontrar soluciones mínimas deseables, ya que la población comienza con muchos nodos y conexiones innecesarias ya presentes. Ninguno de estos nodos o conexiones ha tenido que soportar una sola evaluación, lo que significa que no hay justificación para su configuración.

Desafíos en TWEANNs

Poblaciones iniciales e innovación topológica

- Soluciones:
  - Una forma de forzar topologías mínimas es incorporar el tamaño de la red en una función de penalización del *fitness*.
    - Así, las redes más grandes tienen su aptitud penalizada. Sin embargo es difícil saber qué tan grande la penalización debería ser, según el tamaño de red.
  - Comenzar con una población mínima, es decir, sin capas ocultas y hacer crecer la estructura solo si beneficia la solución.

NeuroEvolution of Augmenting Topologies (NEAT)

- Un enfoque que ha ganado considerable atracción al enfrentar estos desafíos es el algoritmo NEAT.
- NEAT fue presentado por Kenneth O. Stanley y Risto Miikkulainen[1] en el 2002. NEAT respondía a la pregunta de cómo obtener ventajas evolutivas en las topologías y pesos de las RNs.

Kenneth O. Stanley

https://www.cs.ucf.edu/~kstanley/

Risto Miikkulainen

http://nn.cs.utexas.edu/?risto

1. Stanley, K. O. & Miikkulainen, R. Evolving neural networks through augmenting topologies. *Evol. Comput.* **10**, 99–127 (2002).

NEAT – Importancia

NEAT aborda la solución a tres desafíos fundamentales planteados en TWEANNs :

1. ¿Cómo crear convenciones competentes (estructuras útiles) y como realizar cruzamiento?

   Esto fue resuelto creando marcas históricas: le dice al operador de cruce qué partes de dos redes neuronales son similares y por lo tanto, pueden intercambiarse.

2. ¿Cómo se puede proteger la innovación topológica que necesita pocas generaciones para ser optimizada, para que no desaparezca de la población prematuramente?

   NEAT protege la innovación estructural mediante especiación.

NEAT – Importancia

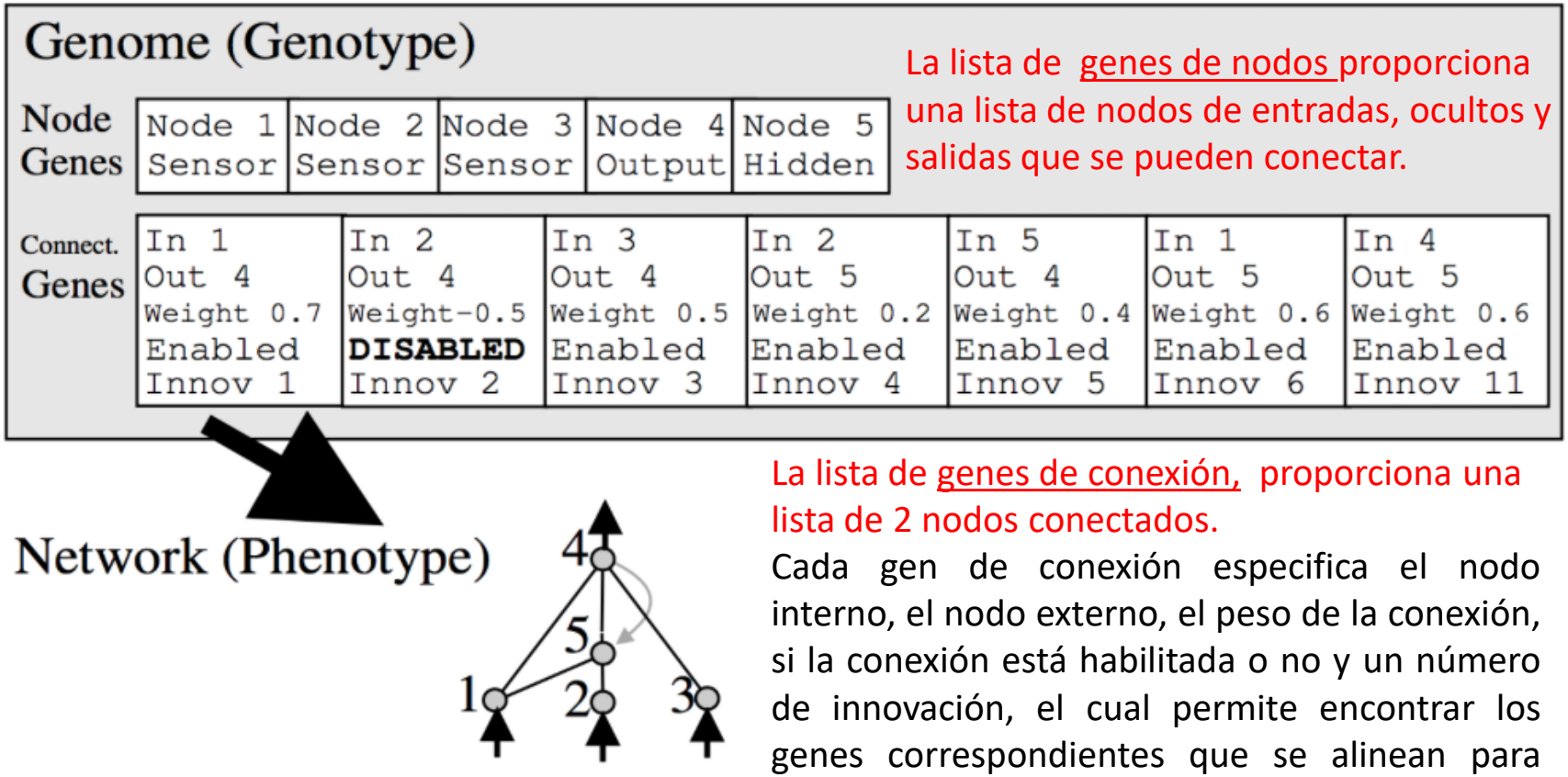
3. ¿Cómo se pueden minimizar la complejidad de las topologías a lo largo de la evolución?

   El potencial de NEAT está en comenzar una estructura de manera simple y luego aumentar la complejidad de la red durante las siguientes generaciones. Así, NEAT minimiza la dimensionalidad a través del crecimiento incremental de estructuras.

NEAT - Funcionamiento

- **Codificación genética en NEAT**

  Los genomas son representaciones lineales de la conectividad de la red



La lista de genes de nodos proporciona una lista de nodos de entradas, ocultos y salidas que se pueden conectar.

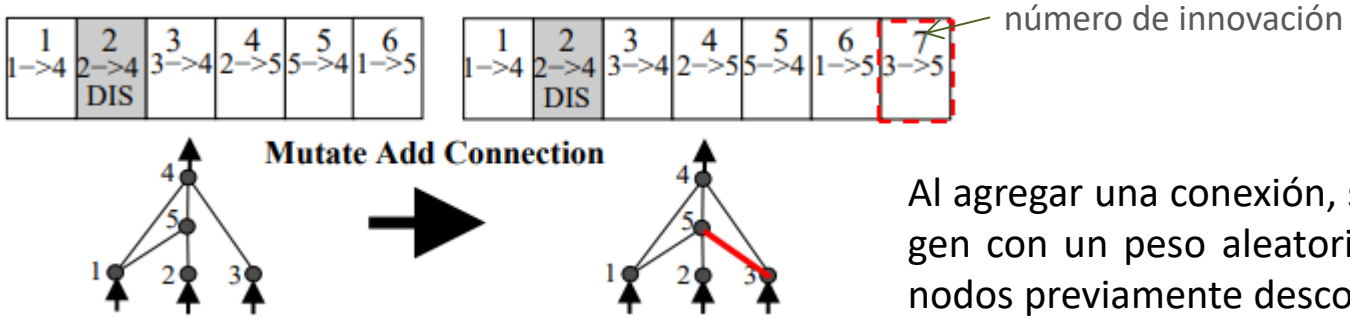
La lista de genes de conexión, proporciona una lista de 2 nodos conectados.

Cada gen de conexión especifica el nodo interno, el nodo externo, el peso de la conexión, si la conexión está habilitada o no y un número de innovación, el cual permite encontrar los genes correspondientes que se alinean para cruzamiento.

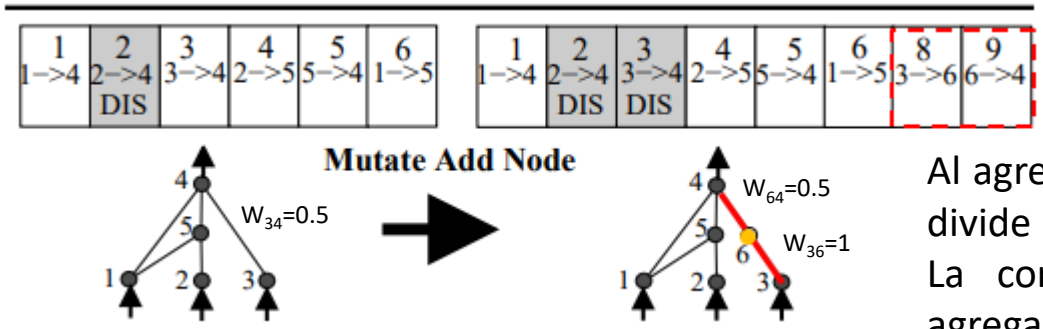
NEAT - Funcionamiento

- **Mutación**

  La mutación en NEAT puede cambiar los pesos en la conexión o cambiar las estructuras de la red.



Al agregar una conexión, se agrega un nuevo gen con un peso aleatorio que conecta dos nodos previamente desconectados.



Al agregar un nodo, una conexión existente se divide y el nuevo nodo se coloca en el medio. La conexión anterior se deshabilita y se agregan dos conexiones nuevas al genoma.
La conexión que conduce al nuevo nodo recibe un peso de 1, y la nueva conexión que sale recibe el mismo peso que la conexión anterior.

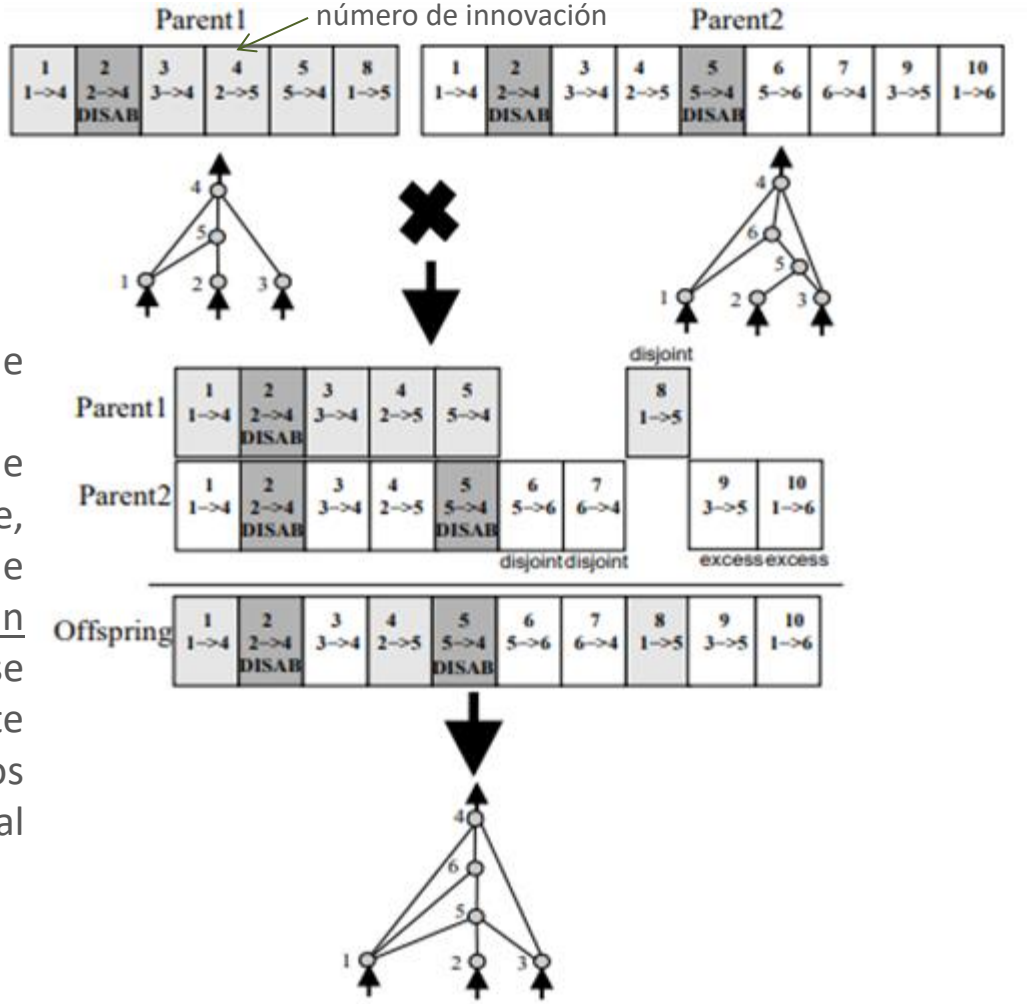
NEAT - Funcionamiento

- Seguimiento de genes a través de marcas históricas
  - Dos genes con el mismo origen histórico se espera que representen la misma estructura (aunque posiblemente con pesos diferentes), ya que ambos se derivan del mismo gen ancestral de algún punto en el pasado.
  - Por lo tanto, para saber qué genes se alinean se hace un seguimiento al origen histórico de cada gen en el sistema. Cada vez que aparece un nuevo gen (a través de la mutación), se incrementa un número de innovación global y se asigna a ese gen. Los números de innovación representan la cronología de la aparición de cada gen en el sistema.
  - Cuando se realiza el cruzamiento, los genes en ambos genomas con los mismos números de innovación son alineados.

NEAT - Funcionamiento

- Seguimiento de genes a través de marcas históricas

Ejemplo: **Cruzamiento** de genomas de diferentes topologías utilizando números de innovación.



La descendencia hereda los números de innovación.

Los genes coincidentes en sus números de innovación se heredan aleatoriamente, mientras que los genes disjuntos (los que no coinciden, en el medio) y los genes en exceso (los que no coinciden al final) se heredan del mejor progenitor. En este caso, se asume igual *fitness*, por lo que los genes disjuntos y en exceso se heredan al azar.

NEAT - Funcionamiento

- Protegiendo la innovación a través de la especiación
  - Redes similares son agrupadas restringiendo la competencia y el apareamiento dentro de un mismo espacio.
  - La idea es dividir la población en especies de manera que topologías similares se encuentren en la misma especie. Esta tarea parece ser un problema de coincidencia de topologías. Sin embargo, nuevamente resulta que las marcas históricas ofrecen una solución eficiente.
  - El número de genes en exceso y disjuntos entre un par de genomas es una medida natural de su distancia de compatibilidad. Cuanto más disjuntos son dos genomas, menos historia evolutiva comparten y por lo tanto, son menos compatibles.

NEAT - Funcionamiento

- Minimizando la dimensionalidad a través del crecimiento incremental a partir de una estructura mínima
  - NEAT inicia con una población de redes con cero nodos ocultos (es decir, todas las entradas se conectan directamente a las salidas).
  - Luego se introduce una nueva estructura de forma incremental a medida que ocurren mutaciones estructurales, y solo sobreviven aquellas estructuras que resultan útiles a través de evaluaciones del fitness.
  - Asi, minimizar la dimensionalidad le da a NEAT una ventaja de rendimiento en comparación con otros enfoques.

NEAT - Consideraciones

- NEAT es una contribución importante a los AGs porque muestra cómo es posible que la evolución desarrolle soluciones cada vez más complejas a lo largo de generaciones.
- NEAT soluciona los desafíos planteados por TWEANNs y esto hizo que la evolución de las topologías (a través de una serie de innovaciones) sean cada vez más complejas y más efectivas.
- NEAT aborda el problema de cruzar distintas topologías a través de marcas históricas (lo que le dice al operador de cruce qué partes de dos redes neuronales son similares y por lo tanto, pueden intercambiarse).
- NEAT evita la extinción prematura de estructuras innovadoras aumentadas a través de la especiación.

NEAT – Casos de éxito

- Robótica evolutiva:
- Un éxito destacado fue producir el primer perro robot Aibo de Sony[1].

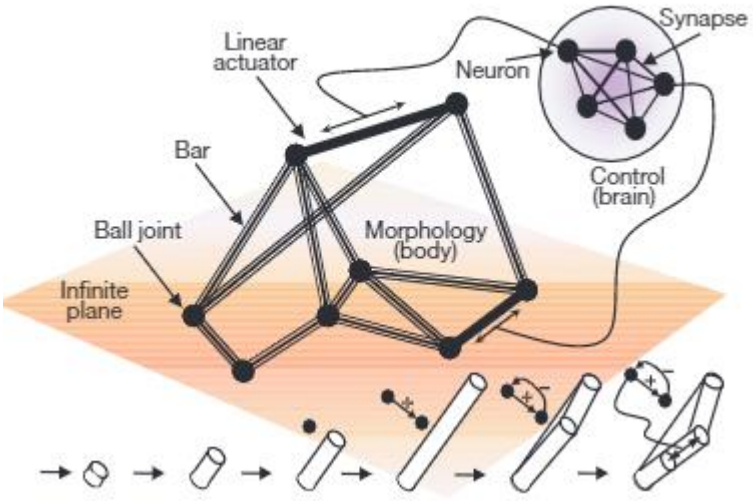
Actualmente está equipado con sensores para detectar sonidos e imágenes y cámaras para permitir que el perro corra hacia su dueño cuando se lo llama. Los movimientos de los ojos, orejas y cola ofrecer una amplia gama de "expresiones realistas y una variedad dinámica de movimientos".

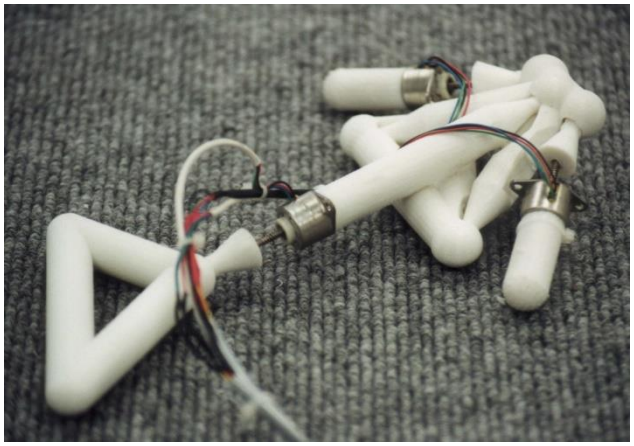
El nuevo Aibo estará equipado tecnología que le permitirá **aprender** trucos, **conocer su entorno** y **desarrollar un vínculo** con su propietario a través del reconocimiento facial y de voz.

1. Hornby, G *et al.* Evolving robust gaits with AIBO. In *Proc. IEEE Conference on Robotics and Automation* 3040–3045 (IEEE, 2000).

NEAT – Casos de éxito

- Otro éxito, fue evolucionar RNs y morfologías de robots que se imprimieron en 3D y podían moverse en el mundo real[2].

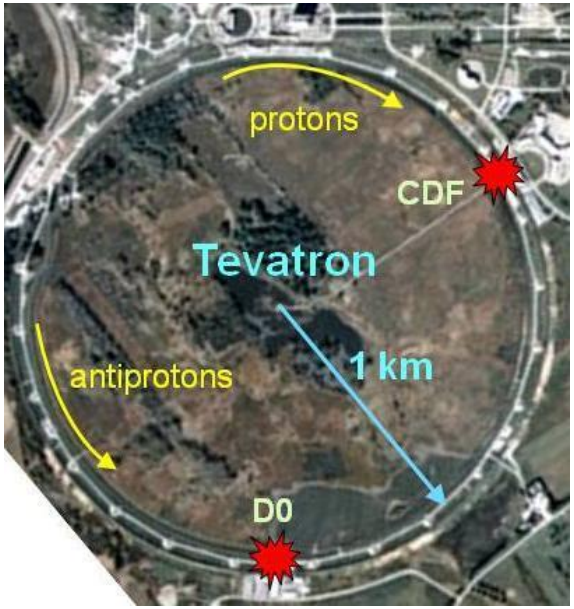




2. Lipson, H. & Pollack, J. B. Automatic design and manufacture of robotic lifeforms. *Nature* **406**, 974–978 (2000).

NEAT – Casos de éxito

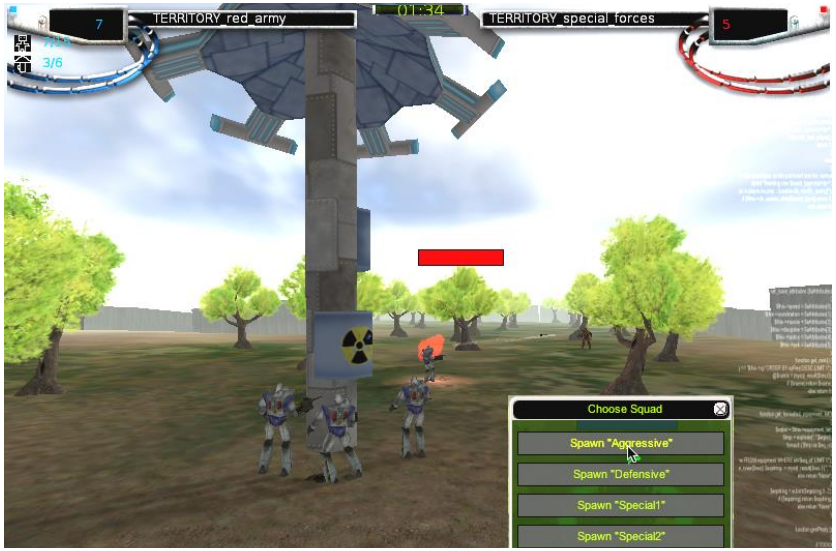
- Otros logros notables incluyen ayudar a descubrir a través de NEAT la medida más precisa hasta ahora de la masa del top quark, que se logró en el colisionador de partículas de Tevatron.



3. Aaltonen, T. et al. Measurement of the top quark mass with dilepton events selected using neuroevolution at CDF. *Phys. Rev. Lett.* **102**, 2001 (2009).

NEAT – Casos de éxito

- La neuroevolución en videojuegos:
- Permite la evolución en tiempo real de nuevos contenidos mientras se está jugango[3], o permite que el jugador entrene a personajes no jugadores como parte del juego a través de la neuroevolución [4].

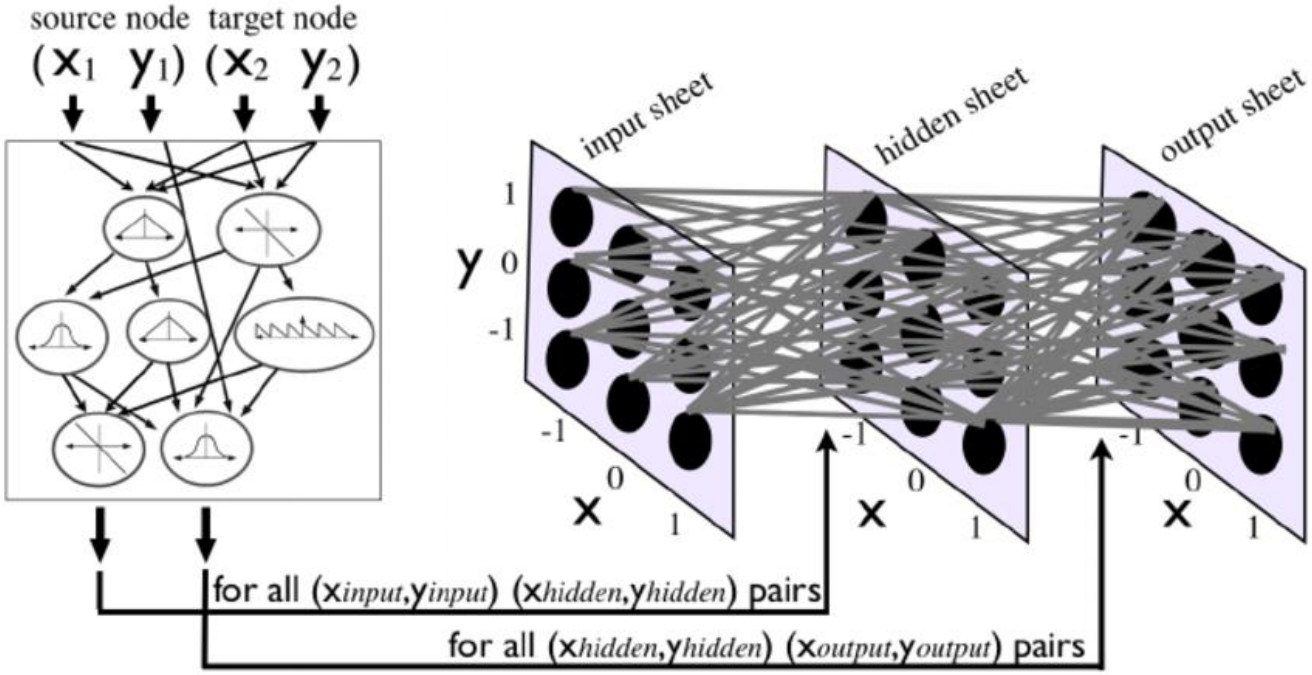


NERO video game
www.nerogame.org

3. Togelius, J., Yannakakis, G. N., Stanley, K. O. & Browne, C. Search-based procedural content generation: a taxonomy and survey. *IEEE Trans. Comput. Intell. AI Games* **3**, 172–186 (2011).
4. Stanley, K. O., Bryant, B. D. & Miikkulainen, R. Real-time neuroevolution in the NERO video game. *IEEE Trans. Evol. Comput.* **9**, 653–668 (2005).

Extensiones de NEAT

HyperNEAT (Hypercube-based NEAT): NEAT basado en hipercubos.
A través de esta técnica, se desarrollan RNs a gran escala, con cientos de miles a millones de conexiones.
En HyperNEAT, las RNs son codificados indirectamente y han demostrado ser útiles por ejemplo para la evolución de los movimientos de robots.



Source: An Empirical Analysis of HyperNEAT (van der Berg, 2013)

Extensiones de NEAT

- **rtNEAT:** una extensión de NEAT que permite que la evolución ocurra en tiempo real en lugar de a través de generaciones.

  La idea básica es poner a la población bajo evaluación constante con un temporizador de "vida" en cada individuo de la población. Cuando el temporizador de una red expira, se examina su fitness actual para ver si cae cerca de una solución sub-optima de la población, y si es así, se descarta y se reemplaza por una nueva red creada por dos padres de mayor fitness. Se establece un temporizador para la nueva red y se coloca en la población para participar en las evaluaciones en curso.

- **odNEAT :** odNEAT es una versión en línea y descentralizada de NEAT diseñada para sistemas multi-robot

Bibliografía

1. Stanley KO, Clune J, Lehman J, Miikkulainen R. Designing neural networks through neuroevolution. Nat Mach Intell 1:24–35 (2019) .
2. Togelius, J., Yannakakis, G. N., Stanley, K. O. & Browne, C. Search-based procedural content generation: a taxonomy and survey. *IEEE Trans. Comput. Intell. AI Games* **3**, 172–186 (2011).
3. Stanley, K. O. & Miikkulainen, R. Evolving neural networks through augmenting topologies. *Evol. Comput.* **10**, 99–127 (2002).
4. Hornby, G *et al.* Evolving robust gaits with AIBO. In *Proc. IEEE Conference on Robotics and Automation* 3040–3045 (IEEE, 2000).
5. Stanley, K. O., Bryant, B. D. & Miikkulainen, R. Real-time neuroevolution in the NERO video game. *IEEE Trans. Evol. Comput.* **9**, 653–668 (2005).
6. Lipson, H. & Pollack, J. B. Automatic design and manufacture of robotic lifeforms. *Nature* **406**, 974–978 (2000).