https://repository.uaeh.edu.mx/revistas/index.php/icbi/issue/archive
Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI

Pädi

Publicación Semestral Pädi Vol. 7 No. Especial (2019) 14-18

Machine Learning para Robots, del Entrenamiento Virtual a la Tarea Real

Gabriel Sepúlvera Cervantes[*, a], Eduardo Vega-Alvarado[a], Edgar Alfredo Portilla-Flores[a]

[a] Centro de Innovación y Desarrollo Tecnológico en Computo del Instituto Politécnico Nacional

Resumen

El aprendizaje automático de máquinas ha tomado fuerza en las últimas décadas debido al avance de la tecnología, logrando que sistemas relativamente pequeños realicen tareas tales como reconocimiento de personas, clasificación de imágenes, diagnósticos médicos hasta control de robots de manera autónoma. La principal limitación que tienen los sistemas de aprendizaje autónomo para robots reales es el tiempo de entrenamiento, ya que durante dicha fase el sistema está sujeto a condiciones físicas tales como la aceleración, la gravedad y las colisiones con el entorno, todas estas situaciones con tiempos de ocurrencia muy grandes comparados con los tiempos de cómputo. Por lo anterior, en este trabajo se presenta una opción para el ML utilizando ambientes virtuales donde la velocidad de interacción física puede alterarse ya que el comportamiento físico es calculado por el procesador de la computadora, logrando acelerar el proceso y reduciendo los tiempos necesarios para el entrenamiento.

Palabras Clave: Robótica, aprendizaje automático, unidad.

1. Introducción

El aprendizaje automático de máquinas o *Machine Learning* (ML), se ha convertido en una herramienta eficaz en la solución de problemas, desde reconocimiento de imágenes y voz (Aggarwal, G., 2015), diagnóstico médico (Morita K., 2017), ciberseguridad (Xin, Y., 2018), hasta control autónomo de robots (Noble, F. 2017). Todas las actividades anteriores pueden realizarse dentro del ámbito computacional, es decir, empleando sólo computadoras y datos electrónicos, con excepción de la tarea de control de robots.

Para implementar un algoritmo de ML en el control de un robot físico, es necesario realizar una serie de entrenamientos de dicho algoritmo, para lo cual el robot controlado debe ser colocado en su posición inicial, con los estados de cada uno de sus actuadores en la configuración acorde a los parámetros de entrenamiento del algoritmo de ML. Ello implica un periodo considerable de tiempo entre un entrenamiento y el siguiente; por otro lado, la ejecución del entrenamiento está sujeta a los tiempos físicos reales de los sensores y actuadores del robot. Por lo anterior, existe un nicho de oportunidad a mejorar dentro del control de robots utilizando algoritmos de ML.

1. Machine Learning para robots

El aprendizaje automático de máquinas (ML) es una herramienta relativamente nueva debido a que la tecnología de las pasadas décadas no permitía su implementación de manera práctica. El ML puede utilizarse para resolver una amplia variedad de problemas, pero es fundamental contar con un canal de entrada de información al sistema para realizar los cálculos y la toma de decisiones adecuada para cada caso. De acuerdo al problema que se quiere resolver se utiliza un tipo específico de ML. El ML se divide en tres ramas principales: Aprendizaje Supervisado, Aprendizaje No Supervisado y Aprendizaje por Refuerzo (Lantz, B., 2015), (ver Figura 1).

Figura 1: Tipos de Aprendizaje de Máquina.

El aprendizaje supervisado es aquel en el cual el sistema a entrenar cuenta con una base de datos previamente etiquetados con el resultado correcto, para evaluar así el desempeño del sistema y corregir el error. Este tipo de aprendizaje se emplea en tareas de reconocimiento de imágenes, predicción de comportamientos, regresión de sistemas, entre otros.

En el aprendizaje no supervisado no se necesitan datos previamente etiquetados; el sistema recibe la información y genera una clasificación de la información en grupos (*clusters*) con coincidencias. Este tipo de aprendizaje se emplea en tareas de reconocimiento de patrones, clasificación de datos, auto codificadores, entre otros. Finalmente el aprendizaje por refuerzo es un tipo particular de aprendizaje a partir de la experiencia que se obtiene mediante la interacción del sistema con el mundo. Este tipo de aprendizaje se emplea en tareas como robots y coches autónomos, agentes para jugar juegos como ajedrez o videojuegos, entre otros.

2.1 Machine Learning con TensorFlow

TensorFlow es una biblioteca de código abierto para cálculo numérico desarrollada por la compañía Google en 2015 (Abadi, M., 2016). La forma de programación implementada en TensorFlow son los grafos. Cada nodo del grafo representa una

[*]Autor de correspondencia
Correo electrónico: gsepulvedac@ipn.mx (Gabriel Sepúlveda-Cervantes)
evega@ipn.mx, (Eduardo Vega-Alvarado)
aportilla@ipn.mx, (Adgar Alfredo Portilla-Flores)

operación matemática mientras que las conexiones entre nodos representan los conjuntos de datos multidimensionales o tensores, (ver Figura 2). Esta biblioteca permite la construcción y entrenamiento de redes neuronales, empleadas para detectar correlaciones y descifrar patrones de forma semejante al aprendizaje y razonamiento humano.

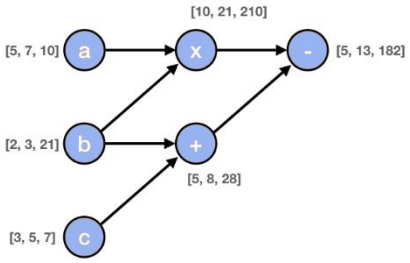

Figura 2: Grafo en Tensorflow

2.2 Aprendizaje por Refuerzo

Tensorflow permite la implementación de diversos algoritmos para aprendizaje de máquina, dentro de las tres áreas: supervisado, no supervisado y por refuerzo. Dado que el aprendizaje por refuerzo (Sutton, R. 2016) es el que más explota la interacción con el entorno del agente inteligente, este tipo de algoritmo se adapta mejor para ser entrenado utilizando Unity.

El aprendizaje por refuerzo es un enfoque computacional mediante el cual un agente trata de maximizar la cantidad total de recompensa que recibe al interactuar con un entorno complejo e incierto. Se distingue de otros algoritmos por su énfasis en el aprendizaje por parte del individuo mediante la explotación de la interacción directa con su entorno, sin depender de una supervisión ejemplar o un modelo completo.

Uno de los principales métodos basados implementados en Tensorflow para el aprendizaje por refuerzo es el de Optimización de Políticas Proximales (PPO), (Achulman, J., 2017). El PPO se basa en la creación de una función objetivo que permite múltiples actualizaciones derivadas de la interacción con el entorno, lo cual la hace preferible para el entrenamiento virtual de robots.

2. Unity3D

Unity es un motor para desarrollo de aplicaciones interactivas 2D y 3D, desarrollado por Unity Technologies en 2005. Unity además integra el motor de física Physics de la compañía Nvidia. En sus orígenes Unity fue utilizado para el desarrollo de videojuegos, pero con el paso del tiempo y la evolución del motor, se fueron extendiendo sus usos, hasta llegar actualmente al desarrollo de aplicaciones interactivas de Realidad Virtual, Realidad Mixta, creación de cortometrajes, aplicaciones móviles y llegando a los simuladores de vehículos (Yang, C.,2016). Así, una de las más recientes aplicaciones de este motor es el entrenamiento de agentes empleando ML, mediante la interconexión entre Unity y Tensorflow.

2.2 Integración Unity3D con Tensorflow

Unity se integra con Tensorflow mediante un kit de herramientas desarrollado por Unity Technologies llamado ML-Agents. Este kit permite que los juegos y simulaciones desarrolladas en Unity sirvan de entornos virtuales para entrenamiento y capacitación de agentes inteligentes (Juliani, A.,2018) (Ayad, S., 2018) (ML-Agents, 2019).

Los agentes pueden capacitarse mediante el aprendizaje por refuerzo, el aprendizaje por imitación, la neuroevolución u otros métodos de aprendizaje automático a través de una API de Python. El kit de herramientas ML-Agents (ver Figura 3) es útil tanto para los desarrolladores de juegos como para los investigadores de inteligencia artificial, ya que proporciona una plataforma central para evaluar los avances en IA en entornos dinámicos desarrollados en Unity, para su implementación en sistemas que se desenvuelvan en un entorno real.

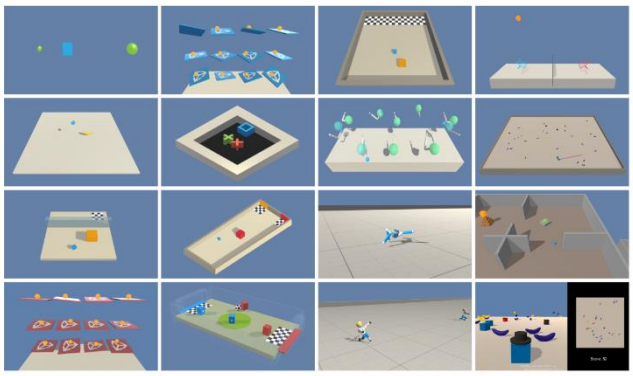

Figura 3: ML-Agents ejemplos incluidos en el Kit (ML-Agents, 2019).

3. Caso de estudio

Para validar la implementación se desarrolló un caso simple de estudio, el cual involucra un robot móvil de 2 ruedas incorporado con 3 sensores de proximidad de tipo ultrasónico, la electrónica de potencia y comunicación necesaria para su control y un sistema Arduino Leonardo como microcontrolador principal, (ver Figura 4).

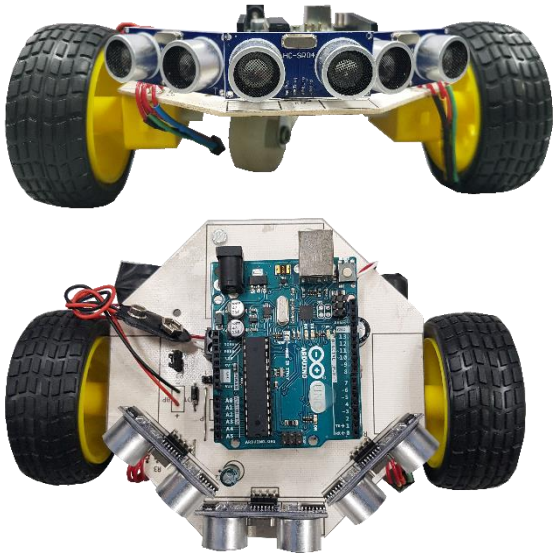

Figura 4: Robot móvil de 2 ruedas y 3 sensores de proximidad

El objetivo del robot móvil es avanzar dentro de una pista sin colisionar con las paredes de la misma; para esto se debe controlar la dirección de giro de cada llanta dependiendo de la información de los sensores de proximidad. Así, se puede determinar la descripción del agente en cuanto a la información

a su alcance y las acciones que puede tomar, como se muestra en las Tablas 1 y 2.

Tabla 1: Información de entrada al agente

Información	*Rango de valores*
Sensor Izquierda	0 , 1
Sensor Central	0 , 1
Sensor Derecha	0 , 1

Tabla 2: Acciones del agente

Acción	*Rango de valores*
Rotación Rueda Izquierda	-1 , 1
Rotación Rueda Derecha	-1 , 1

De esta forma se abstrae la información necesaria para realizar una simulación dentro de un ambiente virtual, ver (Figura 5). Empleando las herramientas del motor Unity se generan rayos de colisión para emular los sensores de proximidad y hacer uso del motor de física para recrear el movimiento producido por el giro de las ruedas. Así, dentro de la simulación se puede entrenar un agente cuyas entradas serán valores flotantes correspondientes al estado de cada sensor de proximidad, como muestran las líneas rojas y modificará el estado de su entorno al moverse a si mismo mediante el giro de las ruedas derecha e izquierda tanto en dirección a favor del reloj CW (valor flotante 1) como en contra del reloj CCW (valor -1), como indican las líneas azules.

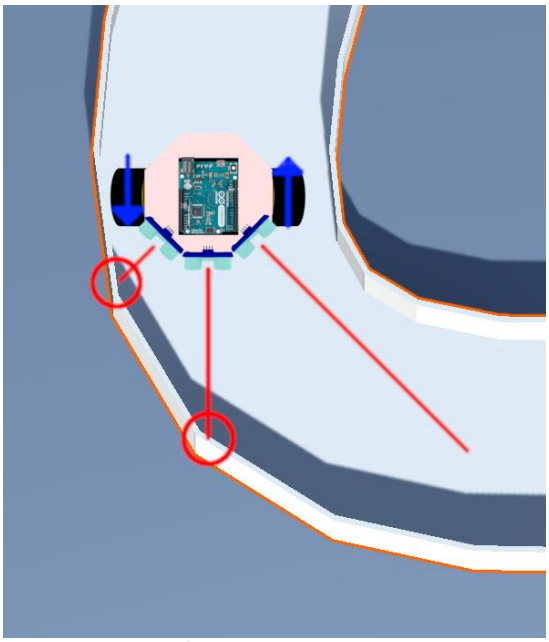

Figura 5: Representación del robot durante un recorrido por la pista

4. Entrenamiento

Para el entrenamiento de los agentes inteligentes se deben integrar las dos tecnologías anteriores: Tensorflow y Unity3D. Esta integración se logra a través de la Interfaz de Programación de Aplicaciones (API) ML-Agents (Juliani, A.,2018) (ML-Agents, 2019).

Dentro del Ambiente virtual desarrollado en Unity se incorporan instancias de los agentes inteligentes para ser entrenados. Cada instancia está conectada a un "cerebro" (objeto en código), el cual tiene como entradas y salidas la información que se muestra en las Tablas 1 y 2. La información recibida es enviada a través de un sistema de comunicación externa para la integración con Python y Tensorflow, ver figura 6. Así, el algoritmo de aprendizaje toma una decisión y comunica esta decisión de regreso hasta Unity. El cerebro ejecuta la acción determinada y recibe una recompensa de acuerdo al resultado de dicha acción. La recompensa está representada por un valor flotante, comúnmente se considera una buena recompensa un valor positivo y un castigo un valor negativo. El valor de la recompensa es enviado por el sistema de comunicación, para que el algoritmo de aprendizaje pueda almacenar el resultado de su acción y promoverla o relegarla para siguientes situaciones posibles. El proceso se repite hasta terminar el entrenamiento.

El resultado final del entrenamiento es una política de generación de acciones a partir del estado del sistema, en este caso el valor de los sensores, creada a partir de la búsqueda de maximizar la recompensa recibida. Esta política queda codificada en un archivo de datos que posteriormente puede ser empleado dentro de Unity ya sin necesidad de comunicación con Tensorflow.

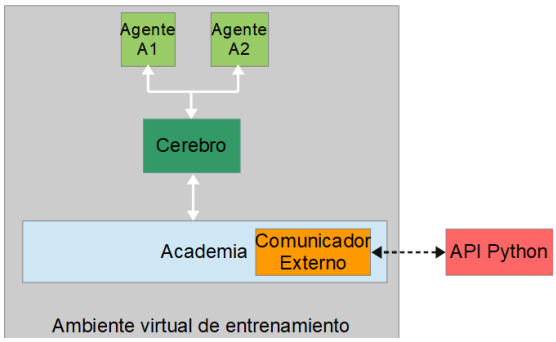

Figura 6: Grafo en Tensorflow

5.1 Ambiente virtual

Para el entrenamiento de los agentes inteligentes que controlarán al robot móvil se desarrolló una pista dentro del software de modelado 3D Blender. Esta pista inicia con cambios suaves para terminar con curvas pronunciadas, lo que permite evaluar al robot móvil desde un ambiente sencillo hasta uno complejo, (ver Figura 7). La pista se incorporó en Unity creando los componentes del motor de física necesarios para generar las interacciones físicas reales, tales como los componentes de colisión, gravedad y masas de los cuerpos.

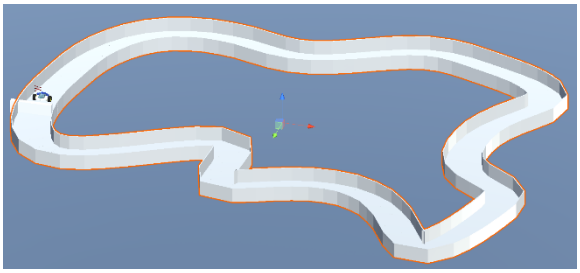

Figura 7: Pista de entrenamiento para agentes inteligentes

5.2 Entrenamiento virtual

Dado que Tensorflow permite el entrenamiento de un mismo cerebro empleando de manera simultánea múltiples agentes virtuales, dentro de Unity se generó un arreglo de cuarenta pistas para el entrenamiento simultáneo, (ver Figura 8). Esto permitió reducir el tiempo de entrenamiento en la misma proporción, es decir entrenar un cerebro que controla esos agentes en lugar de entrenar cuarenta veces el mismo cerebro controlando a un agente.

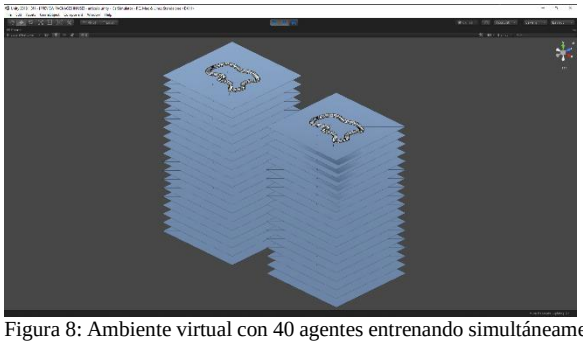

Figura 8: Ambiente virtual con 40 agentes entrenando simultáneamente

El entrenamiento virtual de agentes inteligentes presenta tres ventajas:

- Al estar el entrenamiento codificado y ejecutado en un sistema de cómputo, se pude reiniciar, configurar y alterar con muy poco esfuerzo y tiempo. Ejemplo de esto es colocar el robot en su posición inicial, cambiar la velocidad de movimiento, modificar la distancia de percepción de los sensores, etc.
- Debido a que la simulación física es calculada por un motor computacional el cual no necesita estar corriendo a la velocidad del mundo real, se puede incrementar la velocidad de simulación por encima del tiempo requerido en un ejercicio real. Lo anterior implica que se puede calcular el comportamiento de un objeto regido por leyes físicas en menor tiempo del que le lleva al mundo real realizar dicho comportamiento.
- Dados el ambiente virtual y el motor de física corriendo durante la simulación, se pueden entrenar de manera simultánea varios agentes, lo cual reduce el tiempo necesario para el entrenamiento.

5. Implementación física

Una vez entrenado el sistema y codificada la inteligencia adquirida en un archivo que puede ser utilizado en una aplicación en Unity, se procede a la generación de una aplicación móvil compilada para el sistema operativo Android. Esta aplicación permitirá la comunicación mediante Bluetooth con una tarjeta de desarrollo Arduino Leonardo. Ya establecida esta comunicación se pueden reemplazar las señales de información que utilizan los agentes inteligentes (Tabla 1 y 2) por valores reales de los sensores y actuadores que el sistema Arduino puede leer y escribir.

En resumen, el sistema se entrena utilizando el mundo virtual de Unity y el conocimiento adquirido se incorpora en una aplicación compilada con Unity en un archivo .apk para ser descargado en un celular Android, el cual se comunica vía Bluetooth para obtener el estado del sistema y tomar las acciones determinadas de manera autónoma, (ver Figura 9).

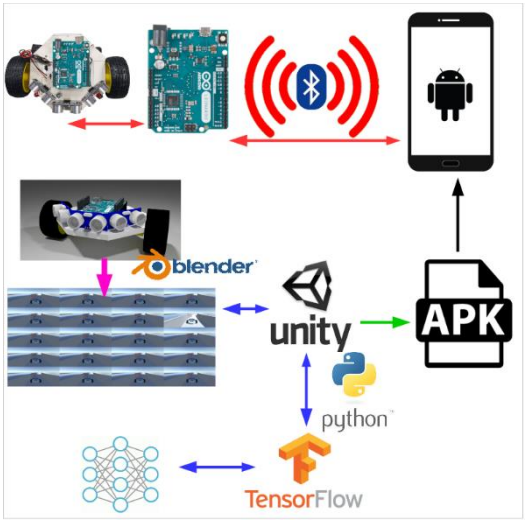

Figura 9: Marco de referencia y flujo de información de la propuesta

6. Conclusión

Se mostró un marco de referencia para entrenar un agente inteligente mediante un algoritmo de aprendizaje por refuerzo dentro de un ambiente virtual, tomar el conocimiento adquirido y embeberlo en una aplicación móvil para celular. La aplicación es capaz de sensar y controlar un robot móvil real basado en el conocimiento adquirido en el entrenamiento virtual.

English Summary

Machine Learning for Robots, from Virtual Training to Real Task

Abstract

Machine learning (ML) has gained strength in recent decades due to the advancement of technology, with relatively small systems performing tasks such as people recognition, classification of images, medical diagnostics, or even autonomous robot control. The main limitation of autonomous learning systems for real robots is the training time, since during the training phase the system is subject to physical conditions such as acceleration, gravity and collisions with the environment, all these situations with very large times of occurrence compared with the computation times. Therefore, this paper presents an option for the use of ML for robot control using virtual environments where the physical interaction speed can be altered since the physical behavior is calculated by the computer processor, accelerating the process and reducing the required training time.

Keywords:

Robotics, Machine Learning, Unity

Agradecimientos

Este trabajo ha sido realizado parcialmente gracias al apoyo del CONACyT México y al Instituto Politécnico Nacional a través de los proyectos SIP20190245 y SIP20195772.

Referencias

Juliani, A.,2018. Unity: A General Platform for Intelligent Agents, arXiv preprint arXiv:1809.02627v1.
Macedo, B. 2015, Evolving Finite-State Machines Controllers for the Simulated Car Racing Championship, Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)
Sita, E. 2017, ROS-Unity3D based system for monitoring of an industrial robotic process IEEE/SICE International Symposium on System Integration (SII)
Noble, F. 2017..A mobile robot platform for supervised machine learning applications, 24th International Conference on Mechatronics and Machine Vision in Practice
Morita K., 2017. Computer-aided diagnosis system for Rheumatoid Arthritis using machine learning, International Conference on Machine Learning and Cybernetics (ICMLC)
Xin, Y., 2018. Machine Learning and Deep Learning Methods for Cybersecurity, IEEE Access
Aggarwal, G., 2015. Characterization between child and adult voice using machine learning algorithm, International Conference on Computing, Communication & Automation.
Sutton, R. 2016, Reinforcement Learning: An Introduction, The MIT Press, Cambridge, Massachusetts, London, England
Lantz, B.,2015, Machine Learning with R, Birmingham-Mumbai: Packt Publishing
Abadi, M. 2016. TensorFlow: A System for Large-Scale Machine Learning. Proceedings of 12th USENIX Symposium on Operating Systems Design and Implementation. ISBN 978-1-931971-33-1
Yang, C.,2016, Unity 3D production and environmental perception vehicle simulation platform, International Conference on Advanced Materials for Science and Engineering
Ayad, S., 2018, Unity3D Based Control Method for a Robotic Ground Walking Platform in a Virtual Reality Environment, IEEE International Conference on Biomedical Robotics and Biomechatronics.
Achulman, J., 2017, Proximal Policy Optimization Algorithms, arXiv:1707.06347 v2.
ML-Agents, 2019, https://github.com/Unity-Technologies/ml-agents.