UNIVERSIDAD POLITÉCNICA DE VICTORIA

IMPLEMENTACIÓN DE UN ALGORITMO DE APRENDIZAJE REFORZADO CON PYTHON Y LA BIBLIOTECA TENSORFLOW

REPORTE DE INVESTIGACION
DE ESTANCIA II
INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

PRESENTA:
FROYLÁN MELQUIADES WBARIO MARTÍNEZ
(1530006)

A CARGO:
DR. ALBERTO GARCÍA ROBLEDO

CIUDAD VICTORIA, TAMAULIPAS, ABRIL DE 2018

Resumen

En la actualidad se estudian diferentes metodos para resolver problemas computacionales complejos, para los cuales la complejidad espacial y temporal juegan un papel fundamental las cuales los programadores han ideado nuevo metodos para poder optimizarlos y hacer soluciones posibles cuando esto no es viable.

La inteligencia artificial es una de estas nuevas ramas de la ciencia computacional la cual estudia y emulan los comportamientos inteligentes de los seres vivos, asi como problemas tratados en el parrafo anterior.

En este documento se propone una solucion a un entorno en donde se aplica inteligencia artificial apoyado por algoritmos de aprendizaje automatico y mas puntualmente algoritmos de aprendizaje por refuerzo, los cuales debido a su complejidad son necesarias tecnicas mas complejas como lo son las redes neuronales artificial para aprovechar su potencia y de estas forma idear una solucion optima.

Para la realizacion del proyecto se necesito de la configuracion y entorno de Python 3.6 con las librerias numpy, OpenAI gym y Tensorflow.

Ademas de esto de vio a la tareas de investigar los diferentes metodos que existen en el aprednizaje automatico, asi como la funcion de las redes neurnales artificiales y su integracion con los algoritmos de aprendizjae por refuerzo.

Palabras claves: Redes Neuronales Artificiales, Aprendizaje Por Refuerzo, Q-Learning.

Summary

At present, different methods are being studied to solve complex computational problems, for which spatial and temporal complexity play a fundamental role and for which programmers have devised new methods to optimize them and make possible solutions when this is not feasible.

Artificial intelligence is one of these new branches of computational science which studies and emulates the intelligent behaviors of living beings, as well as problems discussed in the previous paragraph.

In this document we propose a solution to an environment where artificial intelligence is applied supported by automatic learning algorithms and more punctually learning algorithms by reinforcement, which due to their complexity are necessary more complex techniques such as artificial neural networks to take advantage of their power and thus ideas an optimal solution.

For the realization of the project, the configuration and environment of Python 3.6 with the numpy, OpenAI gym and Tensorflow libraries was required.

In addition to this he saw the task of investigating the different methods that exist in the automatic learning, as well as the function of artificial neural networks and their integration with the algorithms of learning by reinforcement.

Keywords: Artificial Neural Networks, Reinforcement Learning, Q-Learning.

Índice

Resumen II

Summary III

Índice IV

1. Introducción 1
- 1.1. Antecedentes . . . 1
- 1.2. Justificación . . . 1
- 1.3. Objetivo General . . . 1
- 1.4. Objetivo Particular . . . 1
- 1.5. Descripción y Alcances del Proyecto . . . 2

2. Marco Teorico 3
- 2.1. Inteligencia Artificial . . . 3
- 2.2. Aprendizaje Automatico . . . 4
- 2.3. Aprenizaje Supeervisado . . . 5
- 2.4. Aprendizaje no supervisado . . . 6
- 2.5. Aprendizaje Semisupervisado . . . 6
- 2.6. Aprenizaje Por Refuerzo . . . 7
- 2.7. Algoritmo Q-Learning . . . 8
- 2.8. Redes Neuronales . . . 9
- 2.9. TensorFlow . . . 10
- 2.10. OpenAI gym . . . 12

3. Desarrollo del problema 13
- 3.1. Entorno FrozenLake-v0 . . . 13
- 3.2. Q-Table: Solucion a FrozenLake-v0 con la ayuda de tablas de decision . . . 13
- 3.3. Q-Network: Solucion al entorno FrozenLake-v0 aplicando redes neuronales artificiales . . . 16
- 3.4. Entorno NChain-v0 . . . 17
- 3.5. Solucion al entorno NChain-v0 aplicando redes neuronales . . . 18

4. Resultados 20
- 4.1. Resultados de Q-Table . . . 20

5. Conclusiones 21

1. Introducción

En las ultimas decadas el crecimiento en la investigacion y aplicacion de la inteligencia artificial ha evolucionado la forma en que se programan las computadoras, y ademas la constante busqueda de nuevos metodos eficientes para llevara a cabo dicha tarea.

Desde 1956 que se acuño este termino, los ingenieros e investigadores se han dado a la tarea de encontrar nuevos modelos para realmente lograr un avance significativo. Con ello se crearon diversas ramas de investigacion para cubrir todo el campo de esta materia.

Una de estas ramas es el Aprendizaje Automatico (Machine Learning ML) que en los ultimos años a tenido un gran crecimiento en las investigaciones e incluso ha llegado a implementarse esta rama junto con las redes neuronales artificiales las cuales dan una mayor potencia y eficiencia a los metodos previamente ideados.

1.1. Antecedentes

El ser humano desde que tiene memoria ha buscado la manera de mejorarse y aprender de sus acciones. A esto podria decirse que es llamado el aprendizaje de prueba y error. Es precisamente este concepto que a lo largo de los años el humano ha querido utilizar para desarrollar nueva y mejor tecnologia adaptable al entorno.

1.2. Justificación

El motivo por el que se desarrolla esta investigacion es para implementar algoritmos de aprendizaje por refuerzo, mas precisamente en los algoritmos de Q-Learning, aplicando una variante de ellos implementando metodos con redes neuronales.

1.3. Objetivo General

Desarrollar la solucion a un entorno de entrenamiento de OpenAI gym aplicando Q-Learning y mejorando el modelo junto con redes neuronales tipo FeedForward para la eficacia y eficiencia al momento de que el agente tome las decisiones.

1.4. Objetivo Particular

- Investigacion de Inteligencia Artificial.
- Investigacion de Machine Learning y algoritmos Q-Leaning
- Investigacion de las Redes Neuronales y sus tipos
- Instalacion del entorno TensorFlow y OpenAI gym
- Imeplementacion de un modelo de Q-Learning y redes neuronales para resolver un entorno de Open gym

1.5. Descripción y Alcances del Proyecto

Este proyecto solo esta enfocado a redes neuronales y problemas simples en ambientes de pocos estados y controlados por la bilbioteca OpenAI. Aplicando redes neuronales de una capa y simples funciones de activacion binarias, asi como el uso de la biblioteca de TensorFlow para administrar el entorno de la red.

2. Marco Teorico

2.1. Inteligencia Artificial

La Inteligencia Artificial es un campo de las ciencias computacionales enfocado en el desarrollo de computadoras capaces de hacer cosas que son normalmente hechas por personas en particular, cosas asociadas con personas actuando inteligentemente.

Un investigador de la Universidad de Standford acuñó el termino en 1956 durante la que hoy se conoce como la conferencia de Dartmouth, en la cual el nucleo de la IA fue definido:

"Se intentará encontrar la manera de hacer que las máquinas utilicen el lenguaje, formen abstracciones y conceptos, resuelvan problemas que ahora están reservados a los humanos y se mejoren a sí mismas. Creemos que se puede lograr un avance significativo en uno o más de estos problemas si un grupo cuidadosamente seleccionado de científicos trabajan juntos durante un verano."

La difinicion precisa y el significado de la palabra inteligencia, y más aun de la inteligencia artificial, es el tema de mucha discusion. Un diccionario solo, por ejemplo da cuatro definiciones de Inteligencia Artificial:

- Un area de estudio en el campo de la informatica. La inteligencia artificial se ocupa del desarrollo de computadoras capaces de participar en procesos de pensamiento similares a los humanos, como el aprendizaje, el razonamiento y la autocorrecion.
- El concepto de que las maquinas pueden ser imitadas asume algunas capacidades normalmente pensadas como la inteligencia humana como el aprendizaje, la adaptacion, la autocorrecion, etc.
- La extension de la inteligencia humana a traves del uso de computadoras, como en tiempos pasados, el poder fisico se extendi a traves del uso de las herramintas mecanicas.
- En un sentido restringido, el estudio de las tecnicas para utilizar las computadoras de formas mas eficaz mediante tecnicas de programacion mejoradas

(The New International Webster's Comprehensive Dictionary of the English Language, EncyclopedicEdition)

Hoy en dia la comunidad de Inteligencia Artificial ha tratado de imitar comportamientos inteligentes con programas computacionales. Esto no es una tarea facil porque un programa de computadora debe se capaz de hacer muchas maneras en orden para ser llamado inteligente.

Hay muchas mas definiciones para la Inteligencia Artificial, pero la mayoria se pueden clasificar en cuatro categorias:

- Sistemas que piensan como humanos
- Sistemas que actuan como humanos
- Sistemas que piensan racionalmente
- Sistemas que actuan racionalmente

2.2. Aprendizaje Automatico

El Aprendizaje Automatico es la ciencia de la programacion de las computadoras en la cual purden aprender de deteccion automatizada de patrones significativos en datos.

Una ligera definicion:

El aprendizaje automatico es el campo de estudio que le da a las computadoras la habilidad de aprender sin ser explicitamente programadas

-Arthur Samuel, 1959

Una definicion mas en terminos de Ingenieria:

Un programa computacional es dicho para aprender de la experiencia E con respecto a alguna tarea T y algo de medida de rendimiento P, si su rendmiento en T, por medio de P, mejorada con la experiencia E.

-Tom Mitchell, 1997

Aplicaciones del Aprendizaje Automatico:

- Ranqueo de paginas: Esto es, el proceso de enviar una consulta a un motod de busquea, que luego encuentra las paginas web relevantes para la consulta y que las devuelve en su orden de relevancia. El aprendizaje automatico en lugar de se utiliza para automatizar el proceso de diseño de un buen motor de busqueda.

- Filtracion Colaborativa: Las compañias electronicas que ofrecen musica y pelicuas en streamming como Amazon o Netflix usan la informacion extensamente para convencer a los usuarios de comprar contenido en base a su informacion.

- Traduccion automatica de documentos: En un extremo, nosotros podemos enfocar un completo entendimiento de un texto antes de traducirlo usando un conjunto de reglas seleccionadas por un linguista computacional bien versado en los dos idiomas que nos gustaria traducir

- Muchas aplicaciones de seguridad: por ejempo el control de acceso, usando patrones de reconocimiento como uno de sus componentes. ESto es dando una foto (o grabando un video) de una persona, reconociendo quien es esta persona. En otras palabras, el sistema necesita clasificar las caras dentro de una de muchas categorias (Alicia, Bob, Carlos) o decidir que es una cara desconocida.

Aprendizaje Automatico es bueno en:

- Problemas para los que las soluciones existentes requieren de mucho ajuste manual o largas listas de reglas: un algoritmo de Aprendizaje Automatico a menudo puede simplificar el codigo y rendir mejor

- Problemas complejos para los cuales no existe una buena solucion utilizando un enfoque tradicional: las mejores tecnicas de Aprendizaje Automatico pueden encontrar una solucion

- Entornos fluctantes: un sistema de Aprendizaje Automatico puede adaptarse a nuevos datos

- Obtener informacion sobre problemas complejos y grandes cantidades de datos

Tipos de sistemas de Aprendizaje Automatico:
Hay varios tipos de diferentes sistemas basados en apendizaje automatico que se pueden clasificar en las siguiente categorias:

- Si estan o no entrenados con supervision humana (supervisados, no supervisados, semisupervisados y aprendizaje por refuerzo)
- Si pueden o no apredner gradualmente sobre la marcha (aprendizaje en linea versus aprendizaje por lotes)
- Ya sea que trabajen simplemente comparando nuevos puntos de datos con puntos de datos conocidos, o en su lugar detecten patrones en los datos de entrenamineto y contruyan un modelo predictivo, de manera muy parecida a como lo hacen los cientificos (aprendizaje basado en instancias versus aprendizaje basado en modelos)

Los sistemas de aprendizaje automatico pueden ser clasificados de acuerdo a la cantidad y tipos de supervision que reciben durante el entrenamiento. Hay cuatro principales categorias: aprendizaje supervisado, aprendizaje no supervisado, aprenizaje semisupervisado y aprendizaje por refuerzo.

2.3. Aprenizaje Supeervisado

En el aprenizaje supervisado, los datos de entrenamiento que alimentan al algoritmo incluyen las soluciones deseadas llamadas etiquetas.

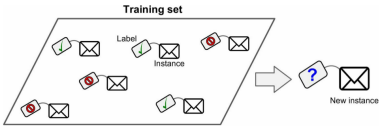


Figura 1: Aprendizaje supervisado

Una tipica tarea en el aprendizaje supervisado es la clasificacion. El filtrador de spam del correo electronico es un buen ejemeplo de esto: este se entrena con muchos ejemplos de correos electronicos junto con su clase (spam o no spam), y debe aprender a clasificar los nuevos correos electronicos.
Algoritmos de Aprendizaje Supervisado

- K-Nearest Neighbors
- Regresion Lineal
- Refresion Logica
- Máquinas de Vectores de Soporte
- Arboles de decision y bosques aleatorios
- Redes Neuronales

2.4. Aprendizaje no supervisado

En el aprendizaje no supervisado, los datos de entrenamiento no estan etiquetados. Los sistemas intentan aprender sin un maestro.

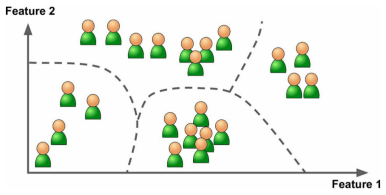


Figura 2: Aprendizaje No Supervisado

Algoritmos de Aprendizaje No Supervisado

- Agrupamiento
  - K-Means
  - Hirerchical Cluster Analysis (HCA)
  - Expectation Maximization
- Visualizacion y reduccion de la dimensionalidad
  - Principal Component Analysis (PCA)
  - Kernel PCA
  - Locally-Linear Embedding (LLE)
  - t-distributed Stochastic Neighbor Embedding (t-SNE)
- Aprendizaje de las reglas de asociacion
  - Apriori
  - Eclat

2.5. Aprendizaje Semisupervisado

Algunos algoritmos pueden tratar con datos de entrenamiento parcialmente etiquetados, normalmente muchos datos sin etiquetar y un poco de datos etiquetados. Esto se llama aprendizaje semisupervisado. Algunos servicios de alojamiento de fotos, como Google Photos, son buenos ejemplos de ello. Una vez que cargues todas tus fotos familiares en el servicio, reconocerá automáticamente que la misma persona A aparece en las fotos 1, 5 y 11, mientras que otra persona B aparece en las fotos 2, 5 y 7. Esta es la parte no supervisada del algoritmo (clustering). Ahora todo lo que el sistema necesita es que le digas quiénes son estas personas. Sólo una etiqueta por persona, 4 y es capaz de nombrar a todos en cada foto, lo que es útil para buscar fotos.

La mayoría de los algoritmos de aprendizaje semisupervisados son combinaciones de algoritmos supervisados y no supervisados. Por ejemplo, las redes de creencias profundas (DBN,

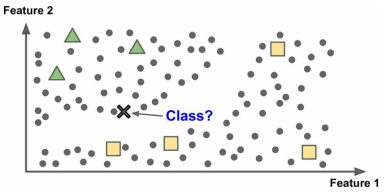


Figura 3: Aprendizaje Semisupervisado

por sus siglas en inglés) se basan en componentes no supervisados llamados máquinas Boltzmann restringidas (RBM, por sus siglas en inglés) apiladas unas encima de otras. Los mecanismos para encuadernación con anillos se capacitan secuencialmente de manera no supervisada, y luego se perfecciona todo el sistema utilizando técnicas de aprendizaje supervisado.

2.6. Aprenizaje Por Refuerzo

El Aprendizaje de Refuerzo (RL) se refiere a un tipo de método de Aprendizaje Automático en el cual el agente recibe una recompensa retardada en el siguiente paso del tiempo para evaluar su acción previa. Se utilizaba sobre todo en juegos (por ejemplo, Atari, Mario), con un rendimiento igual o incluso superior al de los humanos. Recientemente, a medida que el algoritmo evoluciona con la combinación de redes neuronales, es capaz de resolver tareas más complejas, como el problema del péndulo.

Típicamente, una configuración de RL está compuesta de dos componentes, un agente y un entorno:

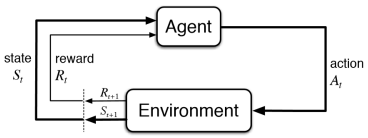


Figura 4: Configuracion del Aprendizaje por refuerzo

Entonces el entorno se refiere al objeto sobre el que el agente está actuando (por ejemplo, el propio juego en el juego de Atari), mientras que el agente representa el algoritmo. El entorno comienza enviando un estado al agente, que luego se basa en su conocimiento para tomar una acción en respuesta a ese estado. Después de eso, el entorno envía un par de estado siguiente y la recompensa de vuelta al agente. El agente actualizará sus conocimientos con el premio devuelto por el medio ambiente para evaluar su última acción. El bucle continúa hasta que el entorno envía un estado terminal, que termina en episodio.

La mayoría de los algoritmos de RL siguen este patrón.

Elementos dentro de un algoritmo de aprendizje por refuerzo:

- **Accion (A):** Todos los movimientos posibles que el agente puede hacer
- **Estado (S):** Situación actual devuelta por el medio ambiente.

- **Recompenza (R):** Un retorno inmediato desde el entorno para evaluar la última acción.
- **Politica (π):** La estrategia que el agente emplea para determinar la siguiente acción basada en el estado actual.
- **Valor (V):** El rendimiento esperado a largo plazo con descuento, a diferencia de la recompensa a corto plazo R. Vπ(s) se define como el rendimiento esperado a largo plazo de la actual política de extinción del estado π.
- **valor Q or accion-valor (Q):** El valor Q es similar al valor V, excepto que toma un parámetro extra, la acción actual a. Qπ(s, a) se refiere al retorno a largo plazo del estado actual s, tomando la acción a bajo la política π.

Libre de Modelo vs Basado en Modelo:

El modelo representa la simulación de la dinámica del entorno. Es decir, el modelo aprende la probabilidad de transición T(s1—(s0, a)) del par de estado actual s0 y la acción a al siguiente estado s1. Si la probabilidad de transición se aprende con éxito, el agente sabrá la probabilidad de entrar en un estado específico dado el estado actual y la acción. Sin embargo, los algoritmos basados en modelos se vuelven poco prácticos a medida que crece el espacio de estado y el espacio de acción (S * S * A, para una configuración tabular).

Por otro lado, los algoritmos sin modelos se basan en pruebas y errores para actualizar sus conocimientos. Como resultado, no requiere espacio para almacenar toda la combinación de estados y acciones. Todos los algoritmos discutidos en la siguiente sección caen dentro de esta categoría.

Con-Politica vs Sin-Politica

Un agente con-politica aprende el valor basado en su acción actual a derivada de la política actual, mientras que su contraparte sin-politica lo aprende basado en la acción a* obtenida de otra política. En Q-learning, tal política es la política codiciosa.

2.7. Algoritmo Q-Learning

Q-Learning es un algoritmo de Aprendizaje Por Refuerzo libre de politicas y modelos basado en la conocida ecuacion de Bellman:

$$v(s) = \mathbb{E}[R_{t+1} + \lambda v(S_{t+1})|S_t = s]$$

E en la ecuación anterior se refiere a la expectativa, mientras que λ se refiere al factor de descuento. Podemos reescribirlo en forma de valor Q:

$$Q^{\pi}(s,a) = \mathbb{E}[r_{t+1} + \lambda r_{t+2} + \lambda^2 r_{t+3} + \ldots | s, a]$$

$$= \mathbb{E}_{s'}[r + \lambda Q^{\pi}(s', a')|s, a]$$

El valor óptimo de Q, indicado como Q*, puede expresarse como:

$$Q^*(s,a) = \mathbb{E}_{s'}[r + \lambda \max_{a'} Q^*(s', a')|s, a]$$

El objetivo es maximizar el valor Q.

2.8. Redes Neuronales

Una red neuronal artificial (RNA) es un modelo computacional basado en la estructura y funciones de las redes neuronales biológicas. La información que fluye a través de la red afecta a la estructura de la RNA porque una red neuronal cambia -o aprende, en cierto sentido- basándose en esa entrada y salida.

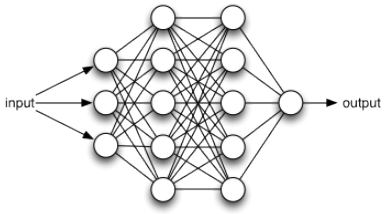
Las RNA se consideran herramientas de modelado de datos estadísticos no lineales en las que se modelan las complejas relaciones entre entradas y salidas o se encuentran patrones.

La RNA también se conoce como red neuronal.

Una RNA tiene varias ventajas, pero una de las más reconocidas es el hecho de que puede aprender de la observación de conjuntos de datos. De esta forma, la RNA se utiliza como una herramienta de aproximación de función aleatoria. Estos tipos de herramientas ayudan a estimar los métodos más rentables e ideales para llegar a las soluciones mientras se definen las funciones o distribuciones de computación. ANN toma muestras de datos en lugar de conjuntos de datos completos para llegar a soluciones, lo que ahorra tiempo y dinero. Las RNA se consideran modelos matemáticos bastante simples para mejorar las tecnologías de análisis de datos existentes.

Las RNA tienen tres capas interconectadas. La primera capa consiste en neuronas de entrada. Esas neuronas envían datos a la segunda capa, que a su vez envía las neuronas de salida a la tercera capa.

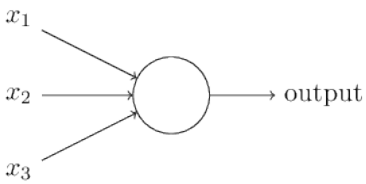
Entrenar una red neural artificial implica elegir entre modelos permitidos para los que existen varios algoritmos asociados.



Perceptron

Existen varios modelos para representar una red neuronal. Estos modelos difieren también en los modelos matemáticos. El modelo inicial y más sencillo de entender es el conocido como Pecepron.

Los perceptrones fueron desarrollados en las décadas de 1950 y 1960 por el científico Frank Rosenblatt. Un perceptrón toma varias entradas binarias, $x1, x2, ...$, y produce una sola salida binaria:



En el ejemplo mostrado el perceptrón tiene tres entradas, $x1, x2, x3$. En general podría tener más o menos entradas. Rosenblatt propuso una regla simple para calcular el resultado.

Introdujo pesos, $w1, w2, ...$, números reales que expresan la importancia de las respectivas entradas a la salida

La salida de la neurona, 0 o 1, se determina por si la suma ponderada $\sum_j w_j x_j$ es menor o mayor que algún valor umbral. Al igual que los pesos, el umbral es un número real que es un parámetro de la neurona. Para ponerlo en términos algebraicos más precisos:

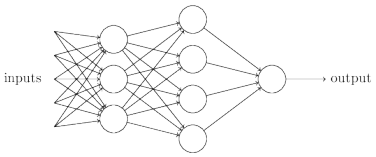
$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$

Simplifiquemos la forma en que describimos los preceptrones. La condición $\sum_j w_j x_j > threshold$ es engorroso, y podemos hacer dos cambios notacionales para simplificarlo. El primer cambio es escribir $\sum_j w_j x_j$ como un producto punto, $w * x \equiv \sum_j w_j x_j$ donde w y x son vectores los cuales componen los pesos y las entradas, respectivamente. El segundo cambio es mover el umbral al otro lado de la desigualdad, y reemplazarlo por lo que se conoce como el sesgo del perceptrón, $b \equiv -threshold$. Usando el sesgo en lugar del umbral, la regla del perceptrón puede ser reescrita:

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

Se puede pensar en el sesgo como una medida de lo fácil que es conseguir que el perceptrón emita un 1. O para decirlo en términos más biológicos, el sesgo es una medida de lo fácil que es conseguir que el perceptrón se dispare. Para un perceptrón con un sesgo realmente grande, es extremadamente fácil para el perceptrón emitir un 1. Pero si el sesgo es muy negativo, entonces es difícil para el perceptrón emitir un 1

En la red de abajo, la primera columna de percepciones - lo que llamaremos la primera capa de percepciones - es tomar tres decisiones muy simples, sopesando la evidencia de entrada. ¿Qué hay de las percepciones en la segunda capa? Cada una de esas percepciones está tomando una decisión sopesando los resultados de la primera capa de la toma de decisiones. De esta manera un perceptrón en la segunda capa puede tomar una decisión a un nivel más complejo y abstracto que los perceptrones en la primera capa. Y el perceptrón de la tercera capa puede tomar decisiones aún más complejas. De esta manera, una red de múltiples capas de percepciones puede participar en la toma de decisiones sofisticadas.



2.9. TensorFlow

TensorFlow es una librería de software de código abierto para computación numérica usando gráficas de flujo de datos. Fue desarrollado originalmente por el equipo de Google

Brain dentro de la organización de investigación Machine Intelligence de Google para el aprendizaje automático y la investigación de redes neuronales profundas, pero el sistema es lo suficientemente general como para ser aplicable en una amplia variedad de otros dominios también.

TensorFlow es multiplataforma. Funciona en casi todo: GPUs y CPUs -incluyendo plataformas móviles y embebidas- e incluso unidades de procesamiento tensorial (TPUs), que son hardware especializado para realizar cálculos tensores.

Modelos de Ejecucion de TensorFlow: Ejecucion de Grafos computacionales

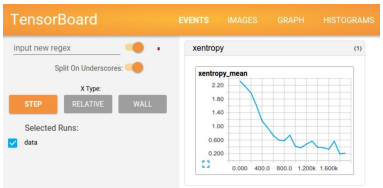
El aprendizaje automático puede volverse complejo rápidamente, y los modelos de aprendizaje profundo pueden hacerse grandes. Para muchos gráficos de modelos, necesita capacitación distribuida para poder iterar dentro de un marco de tiempo razonable. Con Tensorflow puede escribir código para construir un gráfico de cálculo, luego ejecutarlo. El gráfico es una estructura de datos que describe completamente el cálculo que se desea realizar.

Tiene las siguientes ventajas:

- Es portátil, ya que el gráfico puede ejecutarse inmediatamente o guardarse para su uso posterior
- Puede funcionar en múltiples plataformas: CPUs, GPUs, TPUs, móviles, embebidos.
- Es transformable y optimizable, ya que el gráfico puede ser transformado para producir una versión más óptima para una plataforma determinada. Además, se pueden realizar optimizaciones de memoria o de cálculo y realizar compensaciones entre ellas.
- Soporte para ejecución distribuida

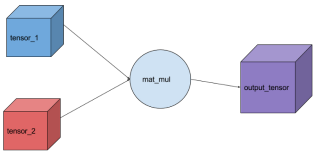
TensorBoard

Tensorboard es un conjunto de aplicaciones web para inspeccionar, visualizar y comprender las ejecuciones y gráficos de TensorFlow. Puede usar TensorBoard para ver las gráficas de su modelo TensorFlow y acercarse a los detalles de las subsecciones de las gráficas. Puede trazar métricas como la pérdida y la precisión durante una ejecución de entrenamiento; mostrar visualizaciones de histogramas de cómo un tensor está cambiando con el tiempo; mostrar datos adicionales, como imágenes; recopilar metadatos de tiempo de ejecución para una ejecución, como el uso total de memoria y las formas de tensores para los nodos; y más.



Computacion basado en Grafos:

Tensorflow es un sistema de programación en el que los cálculos computacionales se representan en forma de gráfos. Los nodos en el grafo se llaman ops (abreviatura de operaciones) - Una operación toma cero o más tensores, realiza una cierta operación computacional y produce cero o más tensores. Un tensor es un conjunto multidimensional y estandarizado.



Los programas Tensorflow suelen estructurarse en una fase de construcción y una fase de ejecución que utiliza una sesión para ejecutar operaciones en el gráfico.

Por ejemplo, es común crear un gráfico para representar y entrenar una red neuronal en la fase de construcción, y luego ejecutar repetidamente un conjunto de operaciones entrenadas en la fase de ejecución.

Para construir un gráfo, se parte de los ops que no requieren ninguna entrada (source ops), como una constante, y pasa su salida a otro ops para realizar el cálculo computacional.

El constructor de operaciones en la librería Python devuelve los objetos que se mantienen para la salida de las operaciones construidas. Puede pasarlos a otras operaciones construidas para usarlos como entradas.

La librería Python para TensorFlow tiene una gráfica por defecto que añade nodos a los builders ops. El gráfico por defecto es suficiente para muchas aplicaciones.

2.10. OpenAI gym

Gym es un juego de herramientas para desarrollar y comparar algoritmos de aprendizaje de refuerzo. No hace suposiciones sobre la estructura de su agente, y es compatible con cualquier biblioteca de cálculo numérico, como TensorFlow o Theano.

La biblioteca gym es una colección de problemas de pruebas - entornos - que puede utilizar para elaborar sus algoritmos de aprendizaje de refuerzo. Estos entornos tienen una interfaz compartida que permite escribir algoritmos generales.

3. Desarrollo del problema

Para la aplicacion de lo visto en el marco teorico se veran ahora tres aplicaciones del algoritmo de Q-Learning con dos diferentes problemas que son parte de los entornos ofrecidos por la biblioteca OpenAI los cuales se describen a continuacion:

3.1. Entorno FrozenLake-v0

FrozenLake-v0 es un entorno estocastico con un espacio de 16 estados y 4 posibles acciones.

El agente controla el movimiento de un personaje en un mundo de cuadrícula. Algunas fichas de la cuadrícula son accesibles para caminar, y otras conducen a que el agente caiga al agua. Además, la dirección de movimiento del agente es incierta y solo depende parcialmente de la dirección elegida. El agente es recompensado por encontrar un camino transitable a una ficha de meta.

El entorno esta descrito con el siguiente enunciado:

El invierno esta aqui. Tú y tus amigos estaban lanzando un frisbee en el parque cuando hiciste un lanzamiento salvaje que dejó el frisbee en el medio del lago. El agua está mayormente congelada, pero hay algunos agujeros donde el hielo se derritió. Si entras en uno de esos agujeros, caerás en el agua helada. En este momento, hay una escasez internacional de frisbee, por lo que es absolutamente imprescindible que navegues por el lago y recuperes el disco. Sin embargo, el hielo es resbaladizo, por lo que no siempre te moverás en la dirección que deseas.

El tablero esta descrito usando una regilla como la siguiente:

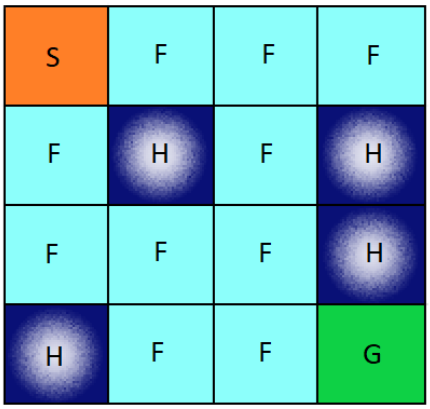


Figura 5: Tablero del FrozenLake

S: Punto inicial, seguro. F: Superficie congelada, seguro. H: Hoyo, cai en las profunidades. G: Meta, donde el frisbee esta localizado.

3.2. Q-Table: Solucion a FrozenLake-v0 con la ayuda de tablas de decision

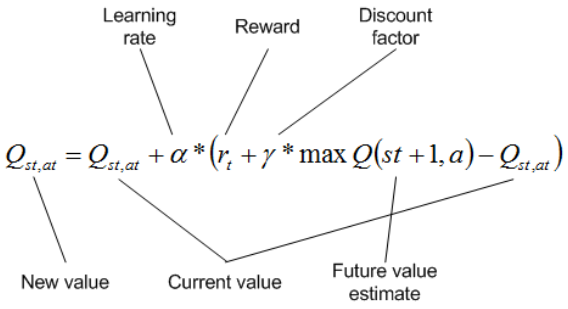
El algoritmo Q-Learning es un algoritmo perteneciente a la familia de los algoritmos de aprendizaje por refuerzo, es uno de los mas usados en esta rama siendo fuera de politica y de

metodo, por ello su flexibilidad al momento de elegir un metodo para elegir la mejor accion a tomar por el agente. El algoritmo tiene algunas variantes dependiendo en su aplicacion siendo dos de estas 1) por medio de tablas de decision que son llenadas y refinidas conforme para el tiempo de ejecucion y 2) aplciando complejos modelso como los son las redes neuronales, en esos problemas en los cuales almacenar valores en las tablas no es una accion factible ni escalable.

Una tabla Q esta dada por los estados: los posibles estados que puede tomar el entorno y por el numero de acciones que puede realizar.

Siendo asi en el problema de FrozenLake-v0 teniendo una tabla Q de 16x4 en la cual se guardan valores que deciden que accion tomara el agente para llegar hasta su objetivo.

La formula que se aplica en este tipo de problemas para llenar o actualizar la tabla en cada uno de los episodios es la siguiente:



Desarrollo algoritmico del problema:

Para la resolucion de este se utilizo el legnuaje de programacion Python 3.6 junto con las biblitecas de Numpy y OpenAI gym, ademas de Matplotlit para graficar los resultados.

```
1  import gym
2  import numpy as np
3  import random
4  import matplotlib.pyplot as plt
5  import time
6  import os
7
8  env = gym.make('FrozenLake-v0')
9
10 Q = np.zeros([env.observation_space.n,env.action_space.n])
11
12 lr = .8
13 y  = .95
14 num_episodes = 2000
15
16 jList = []
17 rList = []
18
```

```
19 for i in range(num_episodes):
20     s = env.reset()
21     rAll = 0
22     d = False
23     j = 0
24
25
26     while j < 99:
27         j+=1
28
29         a = np.argmax(Q[s,:] + np.random.randn(1,env.action_space.n)*(1./(i+1)))
30         s1,r,d,_ = env.step(a)
31         Q[s,a] = Q[s,a] + lr * (r + y * np.max(Q[s1,:]) - Q[s,a])
32         rAll += r
33         s = s1
34
35         env.render()
36
37         if d == True:
38             break
39
40     jList.append(j)
41     rList.append(rAll)
42
43 print("Score over time: " +  str(sum(rList)/num_episodes))
44 print("Final Q-Table Values")
45 print(Q)
```

- Lineas 1-6: Se encargar de importar los paquetes necesarios de las bilbiotecas
- Linea 8: Prepara el entorno FrozenLake el cual establece una matriz de 4 x4
- Linea 10: declara el array Q el cual es inicializado con ceros y el tamaño de la matriz mencionada anteriormente
- Linea 12-14: Configura las constantes que son el factor de aprenizaje (lr) y el factor de descuento (y), asi como el numero total de episodios que son obtenido por el entrenamiento
- Las listas declaradas en las lineas 16 y 17 se usan para almacenar el total de pasos que son realizados en cada episodio y el total de recompenzas en cada episodio
- Lineas 19-42: Denota el ciclo atraves el cual son puestos en marcha los episodios
- Lineas 20-23: El estado del entorno es reseteado a su posicion original, la variable rAll en una variable acumuladora la cual agrega todas las recompenzas ganadas en cada episodio, la variable d es una variable booleana la cual indica si el agente cayo o llego a su destino y la variable j es una variable contadora la cual cuenta el numero de pasos en cada episodio
- Lineas 26-38: Indica el espacio en el tiempo en donde el agente da 100 pasos para moverse alrededor hasta que llegue al destino o caiga dentro de un hoyo
- Linea 29: aplica un ruido al valor de accion

- Linea 30: Llama a la funcion step(), la cual retorna cuatro valores los cuales son: el siguiente estado, la recompenza dada por esa accion, un valor booleano el cual indica si el agente cao dentro de un hoyo o llego al objetivo, y un valor que indica informacion extra para debugging
- Linea 31: se usa la formula de Q-Learning usando las variables previamente afectadas por la siguiente accion tomada
- Linea 32: acumula la recompenza ganada
- Linea 33: El viejo estado es rempelzado con el nuevo
- Linea 35: Muestra el entorno y el movimiento del agente graficamente
- Linea 37: Compara si la variable indicando que el episodio terminado es verdadera
- Linea 40 y 41: Agrega informacion colectada de los pasos y las recompenzas al final de cada episodio
- Lineas 43-45: imprime los resultados finales

3.3. Q-Network: Solucion al entorno FrozenLake-v0 aplicando redes neuronales artificiales

La tabla Q para almacenar valores que influyen en la toma de desciones por parte del agente es una buena aplicacion, pero la desventaja que existe es que en problemas/entornos mas complejos y algunos de la vida real el numero de posibles acciones aumenta drasticamente y en ello una tabla de valores no es una manera viable y ademas poco escalable para resolver ese tipo de problemas. Por ello en temas complejos existen metodos compeljos como lo son las redes neuronales que son las que predicen una buena accion basado en experiencia anterior y consumen menos recursos de memoria lo cual con las tablas queda corto.

Para este problema se tiene el mismo problema, entorno. pero en esta ocasion se utiliza una red neuronal gracias a la biblioteca TensorFlow. Tensorflow provee y adiciona al entorno mucha mas capacidad gracias a su computacion en grafo.

El objetivo de la red es aprende a como mapear los valores Q sin la necesidad de guardar estos en una tabla.

El grafico de la red neuronal a utilizar se puede replesentar en la siguiente figura:

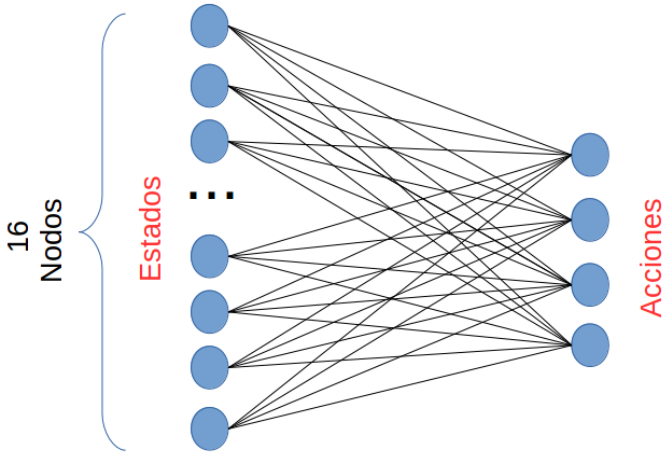
Se utiliza una red neuronal de 16 nodos en la capa de entrada y 4 nodos en la capa de salida.

Un vector con codificacion one-hot toma los estados en la capa de entrada, este vector tiene una medidad de 1x16

Un vector de salida de 1x4 se designa para las salidas de la red y los cuales cada nodo corresponde a una accion posible a tomar por el agente.

Los pesos de la red funcionan como las celdas de la tabla Q teniendo una matriz de 16x4 como resultado de la asignacion de cada nodo de entrada y nodo de salida.

Ademas el metodo de actualizacion es diferente comparado con el Q-Table que se usa una formula para actualizar el valor, en lugar de actualizar directamente la tabla, con la red



neuronal se usa backpropagation y una funcion de perdida (la funcion de perdida es la suma de los cuadrados perdidos)

La funcion de perdida sera la suma de los cuadrados de la perdida, donde la diferencia entre el valor Q actual predicho y el valor objetivo es computado y la gradiente pasa a traves de la red.

En el aprendizaje de backpropagation cada vez que se presenta un vector de entrada de una muestra de entrnamiento, el vector de salida se compara con el valor deseado.

El resultado dado de esta comparacion nos dice cuan lejos estamos del valor deseado para una entrada en particular. El objetivo de la backpropagation es minimizar la suma de el error para todas las muestras de entrenamiento, de modo que la red se comporte de una manera mas deseable ajustando los pesos de la red.

3.4. Entorno NChain-v0

NChain-v0 es otro entorno de la bilbioteca de OpenAI gym el cual cuanta con un espacio de 5 estados y 2 posibles acciones.

Descripcion:

- Se tiene una cadena de 5 estados
- Los arcos son rotulados con las acciones que causa la transicion de estado y su recompenza asociada
- Visualmente la accion abstracta 1 hace que la accion a en el entorno se lleve a cabo
- La accion abstracta 2 causa la accion en el entorno b
- Con posibilidad de 0.2, el agente "se deslizaz su accion tiene el efecto opuesto
- El comportamiento optimo es elegir siempre la accion 1 (aunque esto a veces da como resultado las transiciones etiquetadas con b)

```
from __future__ import division
import gym
import numpy as np
import random
import tensorflow as tf
import matplotlib.pyplot as plt

env = gym.make('FrozenLake-v0')

tf.reset_default_graph()

inputs1 = tf.placeholder(shape=[1,16], dtype=tf.float32, name="x")
W = tf.Variable(tf.random_uniform([16,4],0,0.01), name="w")
Qout = tf.matmul(inputs1,W)
predict = tf.argmax(Qout,1)

nextQ = tf.placeholder(shape=[1,4],dtype=tf.float32)
loss = tf.reduce_sum(tf.square(nextQ - Qout))
trainer = tf.train.GradientDescentOptimizer(learning_rate=0.1)
updateModel = trainer.minimize(loss)

init = tf.global_variables_initializer()

y = 0.99
e = 0.1
num_episodes = 2000
logs_path = '/tmp/tensorflow_logs/example/'

jList = []
rList = []
with tf.Session() as sess:

```

- Una vez que se alcanza el estado 5, se recibe una recompenza de 10 antes que el agente resbale y regrese al estado 1
- cada que cae la accion 2 el agente vuelve al estado 1 del entorno

Este problema requiere de una exploracion efectiva y una estimacion precisa de la recompenza descontada

Acciones:

- a o 1.- Hacia adelate
- b o 2.- Hacia atras

No. Estados:

- 5

Recompenzas:

- +0 accion hacia adelante
- +2 si se resbala y empeza en el estado 1
- +10 si llega al estado 5

Graficamente el entorno se veria asi:

3.5. Solucion al entorno NChain-v0 aplicando redes neuronales

```
with tf.Session() as sess:

    sess.run(init)
    summary_writer = tf.summary.FileWriter(logs_path, graph=tf.get_default_graph())

    for i in range(num_episodes):
        s = env.reset()
        rAll = 0
        d = False
        j = 0

        while j < 99:
            j+=1

            a,allQ = sess.run([predict,Qout],feed_dict={inputs1:np.identity(16)[s:s+1]})

            if np.random.rand(1) < e:
                a[0] = env.action_space.sample()

            s1,r,d,_ = env.step(a[0])

            Q1 = sess.run(Qout,feed_dict={inputs1:np.identity(16)[s1:s1+1]})

            maxQ1 = np.max(Q1)
            targetQ = allQ
            targetQ[0,a[0]] = r + y * maxQ1

            _,W1 = sess.run([updateModel,W],feed_dict={inputs1:np.identity(16)[s:s+1],nextQ:targetQ})
            rAll += r
            s = s1

            if d == True:
                e = 1./((i/50.) + 10)
                break

        jList.append(j)
        rList.append(rAll)

print("Percent of succesful episodes: " + str(sum(rList)/num_episodes) + "%")
plt.plot(rList)
plt.bar(range(len(rList)), rList, color="blue" )
plt.show()
```

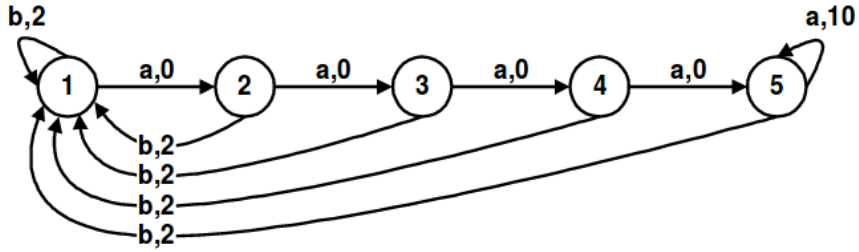


Figura 6: Tablero del NChain

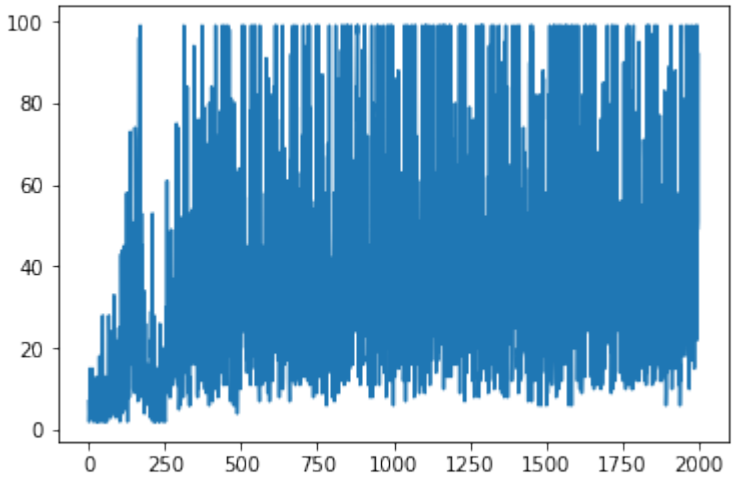
4. Resultados

4.1. Resultados de Q-Table

La tabla Q al final de 200 episodios quedo de la siguiente manera:

```
Final Q-Table Values
[[7.77898393e-02 1.06986406e-02 1.11113480e-02 1.07800744e-02]
 [2.50462922e-04 2.16185731e-04 1.68311083e-03 7.01763680e-02]
 [8.02558221e-03 5.38535479e-02 1.94010733e-03 6.78926257e-03]
 [2.17585492e-04 2.12080398e-04 5.65861034e-05 5.86373225e-02]
 [1.89542941e-01 1.01557483e-03 3.29774533e-03 1.01537032e-03]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [6.85248143e-04 1.37889074e-08 1.18450955e-02 9.95851663e-07]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [8.79437001e-04 5.80516842e-04 9.43828045e-04 2.16870227e-01]
 [2.21680268e-03 3.26185169e-01 7.84763038e-04 0.00000000e+00]
 [8.39476827e-02 2.11686858e-04 2.50229541e-04 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [1.86062292e-03 0.00000000e+00 7.05976132e-01 1.33568562e-03]
 [0.00000000e+00 0.00000000e+00 9.53275649e-01 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]]
```

El rendimiento el algoritmo esta representado por el siguiente grafico:



5. Conclusiones