INTELIGENCIA ARTIFICIAL

APRENDIZAJE AUTOMÁTICO II

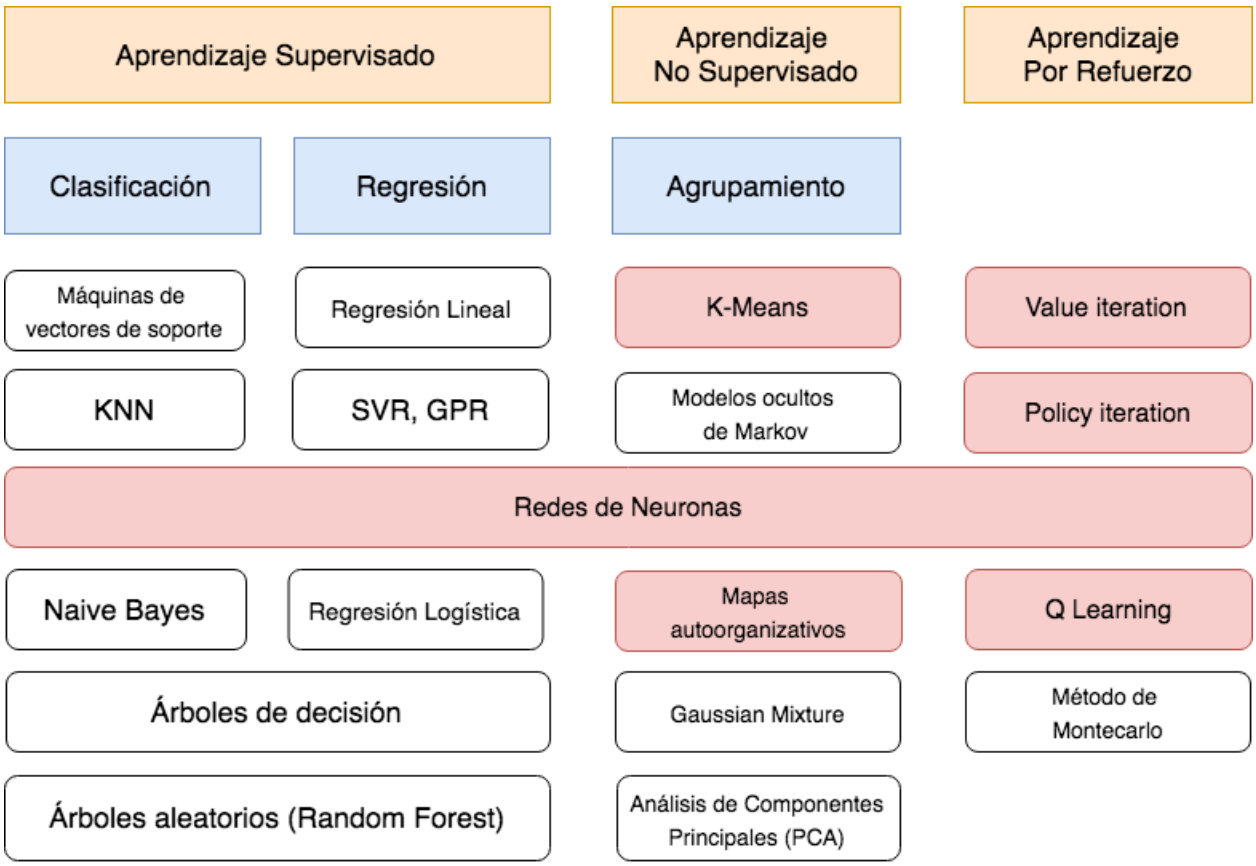
Structuralia

INDICE

1. INTRODUCCIÓN

En el tema anterior se describió de forma global el concepto de Aprendizaje Automático focalizando en los métodos de Aprendizaje Supervisado los cuales utilizan información etiquetada para aprender diferentes modelos basados en clasificación y regresión. Lamentablemente el número de conjuntos de datos correctamente etiquetados es bastante pequeño lo cual produce que la potencia que ofrecen los diferentes algoritmos de Aprendizaje Supervisado desaparezca debido a la complejidad que supone conseguir buenos conjuntos de datos que estén correctamente etiquetados. Además el proceso de etiquetado suele presentar algunos problemas que aumentan su complejidad o disminuyen su efectividad. Esto es debido a que este proceso (manual o automático) puede haberse realizado de manera incorrecta haciendo que los datos sean totalmente inservibles o los datos puede que estén sesgado o que existen situaciones en la cuales no es posible realizar un etiquetado de datos debido a la naturaleza a la naturaleza, cantidad o complejidad de los datos. Este y otros problemas derivados del funcionamiento de los algoritmos los investigadores en Inteligencia Artificial decidieron intentar crear otros tipos de métodos que no necesitaran de información correctamente etiquetada.

Los intentos por resolver estos problemas dieron lugar a la aparición de métodos de Aprendizaje no supervisado los cuales eran capaces de identificar características comunes de las instancias de entrenamiento que les permiten agrupar la información en conjuntos o clases de manera automática. Además los diferentes estudios en otras áreas del proceso de aprendizaje de los seres vivos dieron lugar a la aparición de los algoritmos de Aprendizaje por Refuerzo los cuales intentaban imitar el proceso de aprendizaje de los seres vivos los cuales son capaces de aprender conceptos mediante la utilización de recompensas que permitían reforzar aquella información que debía ser aprendida. A pesar de los avances de estos dos nuevos tipos de familias de métodos seguían existiendo limites al proceso de aprendizaje automático debido principalmente a la complejidad que suponía extraer características útiles, tanto de forma manual como de forma automática, para identificar ciertos elementos que permitan aprender conceptos más complejos, lo que supuso una disminución en la producción de nuevas técnicas de aprendizaje automática. Esta congelación temporal en la aparición de nuevas técnicas de aprendizaje finalizó en 2006 cuando el aprendizaje profundo demostró que ciertas arquitecturas de redes de neuronas eran capaces de extraer características de manera automática y aprender modelos complejos.



En la Figura se presentan diferentes tipos de algoritmos de Aprendizaje Automático distribuidos por familia (Supervisado, No supervisado y Por refuerzo) y por técnica (Regresión, Clasificación y Agrupamiento). En esta figura sólo se presentan algunos de las técnicas más utilizados, pero existen muchas más e incluso diferentes versiones de cada una de ellas. En rojo se marcan los diferentes técnicas que serán descritas de forma detallada en este tema.

2. APRENDIZAJE NO SUPERVISADO

El aprendizaje no supervisado [13] es una de las familias de métodos de aprendizaje donde el proceso de aprendizaje se realiza mediante la utilización de instancias (ejemplos) no etiquetadas. Este tipo de modelo de aprendizaje consiste en definir la información del entorno mediante sólo información de entrada de manera que el resultado de salida (clase o valor) de todas las instancias del conjunto de entrenamiento es definida en base a las diferentes similitudes que pueden ser identificadas entre los diferentes conjuntos de entrenamiento. Esto implica que los valores y/o el número de las posibles salidas de los modelos generados por los métodos de este

de métodos es totalmente desconocido. Dependiendo del método a utilizar puede que el número de posibles salidas sea definido a priori por el usuario. Los diferentes elementos utilizados (categorías, instancias, conjunto de entrenamiento, algoritmo, etc.) en el proceso de aprendizaje de este tipo de métodos es exactamente igual que en el aprendizaje supervisado con la salvedad de que no suele existir una fase de test debido a que no se conocen los posibles valores de salida que van a obtenerse tras el proceso de aprendizaje.

2.1 Algoritmos K-medias

El algoritmo K-medias o algoritmo de Lloyd [1] es un método de aprendizaje no supervisado basado en el concepto de agrupamiento o "clustering" de instancias similares en base a su distribución espacial en el espacio de las variables (características) que definen la estructura de las diferentes instancias. El proceso de agrupamiento consiste en construir o identificar un número finito de conjuntos, grupos o "clusters" (Celdas de Voronoi) en los cuales se encuentran distribuidos un conjunto de instancias donde las instancias de cada uno de los conjuntos son muy similares entre sí y diferentes a las instancias asignadas a los otros conjuntos. Para conocer si una determinada instancia pertenece o no a un grupo, se utiliza una media similitud en base a la distancia (normalmente se utiliza la distancia euclidea) entre el elemento que representa al cluster denominado centroide y el ejemplo. El centroide es un vector de características que representa el punto medio del cluster. Cada uno de estos conjuntos o "clusteres" son las diferentes clases en las que se agruparán las instancias de entrenamiento y que definirán las futuras predicciones del modelo generado. De forma general, se puede decir que el objetivo de este método consiste en aprender los valores de los atributos (variables) que representan a los centrodiodes de cada uno de los grupos o clústeres en los cuales se distribuirán las instancias de entrenamiento. Este algoritmo se aplica sobre conjuntos de instancias con atributos (variables) de tipo continuo, debido a que la mayor parte de las funciones de distancias utilizadas para medir la pertenencia de una instancia a un cluster están basadas en la distancia euclidea o algún otro tipo de distancia de tipo espacial. En el caso de que existan atributos de tipo discreto estos deben ser transformados en variables de tipo continuo con el fin de poder aplicar la función de distancia. En la Figura se presenta un ejemplo de un posible modelo construido mediante la utilización del algoritmo K-medias. En este caso se ha seleccionado un valor de k=3, lo que producirá que el número de clusters generados por el algoritmo sea tres utilizando un conjunto de instancias de entrenamiento compuestas por dos atributos de tipo continuo.

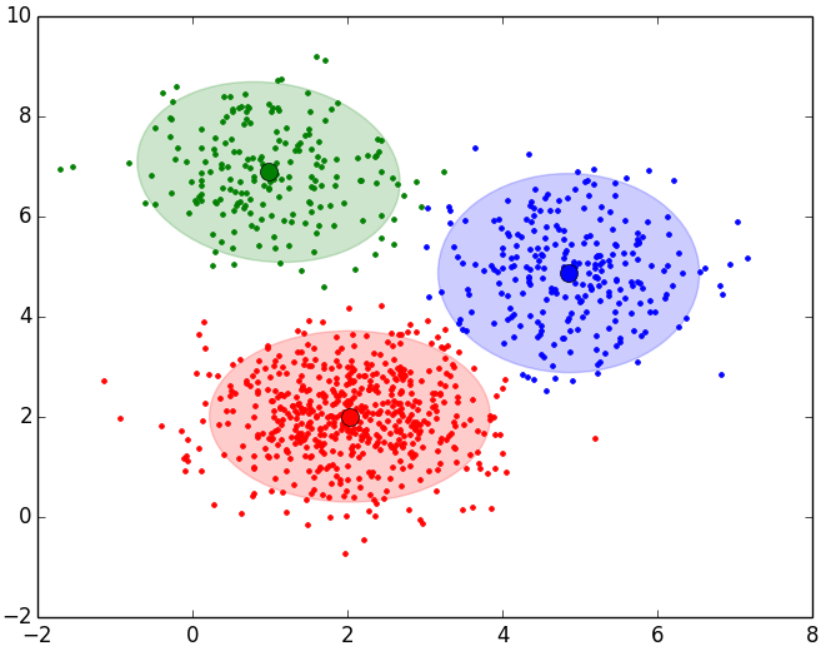


Figura 1: Ejemplo de modelo generado por el algoritmo K-medias utilizando un valor de K igual 3.

2.1.1. Funcionamiento del proceso de auto organización

Como indicamos anteriormente el proceso de aprendizaje del algoritmos K-medias consistente en calcular los valores de los atributos de los centroides que representan a cada uno de los clústeres. Es decir el modelo generado por este algoritmo consiste en un conjunto de vectores cada uno de los cuales corresponde con uno de los centroides. Cada vez que se quiera predecir la clase de una nueva instancia en base a este modelo se calculará la distancia a cada uno de los centroides seleccionándose la clase que representan al centroide que se encuentra más cerca. El proceso de aprendizaje es un proceso de refinamiento iterativo debido a que el valor de los centroides se va refinando en base a las instancias de entrenamiento. Este proceso se encuentra dividido en tres fases:

- Inicialización: El algoritmo K-medias no es capaz de seleccionar de forma automática el número de clústeres en los cuales se deben dividir las diferentes instancias, por lo que es necesario indicar al algoritmo cual será el número de clústeres en los cuales deben dividirse las instancias. Una vez definido este valor, se utiliza un algoritmo de particionamiento aleatorio [2] mediante el cual se seleccionan de forma aleatoria k observaciones del conjunto de las instancias de entradas que son utilizados como

centroides iniciales. La distribución de los centroides da lugar a un diagrama de Vorononi [3]. Esta fase sólo se realiza al comienzo del algoritmos el cual ejecutará las dos fases siguientes durante una serie de iteraciones.

- Asignación: A continuación se realiza un proceso de asignación de las diferentes instancias del conjunto de entrenamiento a su correspondiente cluster. Para ello se calcula la distancia entre cada instancia y los centroides de cada cluster asignándose la instancia al cluster cuyo centroide está más cerca. El cálculo de la distancia suele realizarse mediante la utilización de la distancia euclidea pero es posible utilizar otra medida de distancia.
- Actualización: Una vez realizara la asignación de las instancias a cada grupo se produce una actualización de los valores del centroide en base a las diferentes instancias que han sido asignadas al cluster. Los nuevos valores del cluster se calculan en base a la siguiente ecuación

$$c_i^{t+1} = \frac{1}{|S_i^t|} \sum_{x_j \in S_i^t} x_j$$

donde i es el índice del centroide que se está recalculando en el instante de tiempo t+1, S_i^t es el conjunto de instancias que pertenecen al centroide i en el instante t de tiempo y x_j representa un determinad ejemplo del conjunto S_i^t.

Las fases de asignación y actualización se ejecutarán hasta que cada uno de los centroides converja a un óptimo global. Es decir, hasta que no se produzca ninguna actualización en ninguno de los centroides en base a las diferentes instancias del conjunto de entrenamiento. Debido a que el algoritmo k-medias es un algoritmo heurístico, guiado por una medida de distancia, no existen ningún tipo de garantía de que se converja a un óptimo local que provocara que se finalizara el proceso de aprendizaje. Con el fin de evitar este problema, se introduce una conducción de parada mediante un número máximo de iteraciones de las fases asignación y actualización con el fin de parar el proceso de aprendizaje en caso de que el algoritmo no sea capaz de converger.

2.2 Mapas autoorganizativos

Los mapas auto-organizados (Self-Organizing Maps, SOM en sus siglas en inglés) o mapas de auto-organizativos de Kohonen [4] es un método de aprendizaje no supervisado de tipo competitivo que construye una red de neuronas artificial con una estructura específica. Este método de aprendizaje normalmente es considerado como un algoritmo de agrupamiento ya que agrupa la información en clases en base a sus característica, aunque algunos investigadores lo consideran como un algoritmo de clasificación aunque las clases no se conozcan a priori. El proceso de construcción de esta red no utiliza ningún tipo de conocimiento de control que indique si la red neuronas se están construyendo (configurando) de forma correcta debido a que las instancias del conjunto de entrenamiento no poseen ningún tipo de información objetivo (salida) que indique como debería configurarse a la red. La red auto-organizada debe descubrir características similares, regularidades, correlaciones o categorías en las instancias de entrada que serán incluidos a la estructura interna de conexiones de las diferentes neuronas. Este proceso de auto-organización de las conexiones entre las neuronas en función de los estímulos procedentes de la información de las instancias del conjunto de entrenamiento es el que da nombre a la técnica. En el proceso de aprendizaje competitivo las neuronas compiten unas con otras con el fin de realizar una determinada tarea. Es decir cuando se introduce un valor de entrada a la red sólo una de las neuronas de salida (o un grupo de ellas localizadas espacialmente en la misma área de la cuadricula que define la salida) se deben activar. Este comportamiento produce que las neuronas compitan por activarse, quedando finalmente una única neurona como vencedora y anulándose el resto de ellas, las cuales son forzadas a sus valores de respuesta mínimos. El objetivo de este proceso de aprendizaje es categorizar los datos que se introducen en la red de forma que cuando se detectan valores similares en la misma categoría se debe activar la misma neurona de salida. Las diferentes clases o salidas deben ser definidas por la propia red mediante las correlaciones entre los datos de entrada. Es decir, los mapas auto-organizativos se pueden definir como un algoritmos para clasificar las diferentes observaciones que existen en el conjunto de entrenamiento.

2.2.1. Estructura de los mapas auto-organizativos

El modelo resultante de esta técnica consiste en una red de neuronas compuesta por dos capas cuyas conexiones son hacia delante. La capa de entrada compuesta por N neuronas, una por cada característica de las instancias del conjunto de entrenamiento, encargada de transmitir la información de entrada a la capa de salida. La capa de salida compuesta por M neuronas es la encargada de procesar la información y formar el mapa de categorías. La estructura de la capa

de salida está distribuida como una cuadrícula o mapa bidimensional, donde las diferentes neuronas de la cuadrícula representan a cada categoría (pesos, según la notación de Kohonen) estén correlacionados espacialmente, de modo que los puntos más próximos en la rejilla sean más parecidos entre sí que los que estén muy separados. La estructura de capa de salida suele corresponderse con rectángulo o con un hexágono.

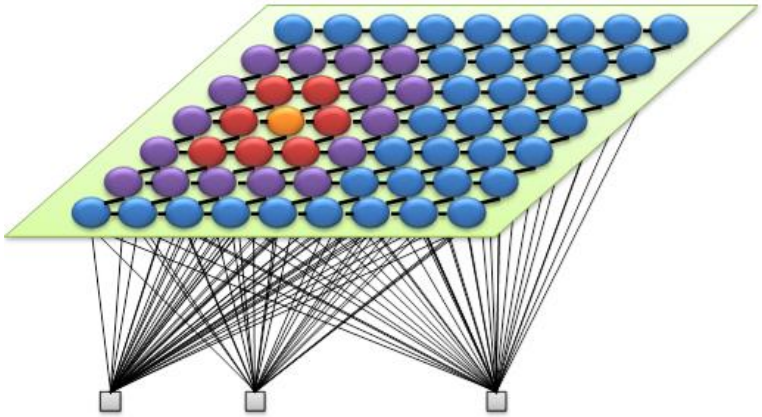


Figura 2: Estructura de la red de neuronas de un mapa auto-organizativo para cuatro posibles clases de salida

La **Figura 2** presenta una ejemplo de un mapa auto-organizativo donde tenemos la capa de entrada está formada por tres neuronas que se corresponden con las tres características presentes en la instancias de entrada y la capa de salida formada por una cuadrícula de 9x7 neuronas que son capaces de predecir 4 clases diferentes. Cada una de las neuronas de la capa de entrada está conectada con cada una de las neuronas de la capa de salida mediante un conexión que posee un peso. De manera que cada una de las neuronas de la capa de salida tienen asociado un vector de pesos, denominado vector de referencia, en el cual se almacenan los valores aproximados de las características que identifican la categorías representadas por la neurona de salida. Esto hace que exista algún tipo de conexión implícita de excitación e inhibición (vecindad), aunque no estén realmente conectadas, entre las neuronas que están juntas debido a que los valores del vector de referencia son muy similares.

2.2.2. Funcionamiento del proceso de auto organización

El proceso de aprendizaje de los mapas auto-organizativos consistente en calcular los valores de cada uno de los vectores de las neuronas de la capa de salida en base a las instancias del conjunto de entrenamiento. El proceso de aprendizaje está dividido en dos fases:

- Inicialización: El proceso de inicialización comienza mediante la selección de forma aleatoria de el número de neuronas que formarán parte de la capa de salida, aunque en algunas implementaciones es posible configurar el tamaño. A continuación se inicializan los valores de los vectores de referencia (pesos) de las neuronas del mapa. Este proceso de inicialización se puede realizar mediante la generación de valores aleatorios o mediante un muestreo aleatorio en base a los valores de dos instancias de entrada. El segundo método de inicialización suele acelerar el proceso de "auto-organización" debido a que los pesos iniciales son una aproximación de los posibles pesos que utilizará el mapa.
- Entrenamiento: El proceso de entrenamiento consta a su vez de dos fases las cuales se aplican sobre cada una de las instancias del conjunto de entrenamiento un número de iteraciones que debe ser definida a priori. La primera de estas dos fases consiste en calcular la distancia de uno de los vectores de entrada, seleccionado de manera aleatoria, a cada uno de los vectores de referencia de las neuronas de la capa de salida (mapa). Normalmente se utiliza la distancia euclidea, pero puede utilizarse otra medida de distancia en base a la naturaleza de los atributos de las diferentes instancias de entrenamiento. A continuación se identifica la neurona más similar (Best Matching Unit, BMU en sus siglas en inglés) a la instancia, en base a la medida de distancia utilizada, y se realiza un proceso de actualización de los pesos de todas la neuronas del mapa con respecto a la neurona BMU. Normalmente sólo se producirán cambios en la neurona BMU y en aquella que se consideren cercanas, es decir aquellas que se encuentran en su vecindad. Durante el proceso de aprendizaje se va produciendo un decremento de la tasa de aprendizaje con el fin de conseguir que cada neurona se especialice y el ratio de propagación de los cambios sea cada vez menor. Es decir, si se permiten una propagación uniforme en el mapa durante el proceso de aprendizaje se perdería el concepto de especialización de las neuronas del mapa eliminando la posibilidad de que aparecieran clases muy residuales, de forma que el algoritmo incluye la tasa de aprendizaje la cual van acercándose a cero a lo largo del proceso de aprendizaje

permitiendo la especialización de las diferentes neuronas del mapa. La formula utilizada para realizar la actualización de los pesos de los vectores de referencia es la siguiente:

$$W_v(t+1) = W_v(t) + V(bmu, v, s)\ \alpha(v)(T(\text{t}) - W_v)$$

Donde $W_v(t)$ es el vector de pesos de la neurona que se quiere actualizar, t es el índice de la iteración en el proceso de aprendizaje, BMU es el índice de la neurona BMU que ha sido identificada como la más cercana al ejemplo de entrenamiento, $\alpha(v)$ es la tasa monótonamente decreciente de aprendizaje, s es índice, diferente de v, de todas las otras neuronas del mapa y t es el índice del ejemplo de entrenamiento en el conjunto T (conjunto de entrenamiento).

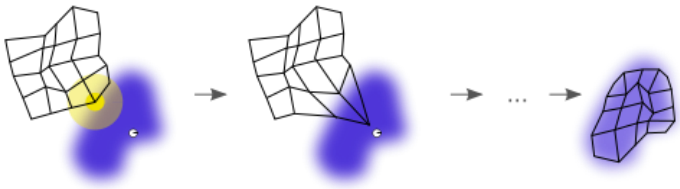


Figura 3 - Ejemplo de funcionamiento del proceso de actualización de los pesos del mapa. © Wikipedia

La función V es la función de vecindad que indica si las neuronas con índice bmu y v forman parte de la misma vecindad. En su versión más sencilla esta función devuelve el valor 1 a todas las neuronas suficientemente cerca a la neurona BMU y 0 a las otras, pero es más común elegir una función gaussiana que define de forma más precisa el distanciamiento que se va produciendo entre la neuronas. Observando la formula se puede apreciar que según se va produciendo el distanciamiento de las neuronas el ratio de cambio en el valor de los pesos es menor produciendo la especialización de las neuronas hacia las diferentes clases. Este proceso de distancia se produce en base a la tasa de aprendizaje que va acercándose a 0 y al resultado de la función de distancia. En la Figura 3 se presenta la forma en la cual se va auto-organizando el mapa durante el proceso de aprendizaje, se puede considerar que el mapa es como una red elástica que se va plegando dentro del espacio de las variables que representan las características de las instancias, por lo que aunque la topología de la capa de salida sea representada

mediante un rectángulo o un hexágono la estructura real mapa depende del proceso de aprendizaje y se parece más a la estructura de la Figura 3.

3. APRENDIZAJE PROFUNDO

El Aprendizaje Profundo o Deep Learning es un termino que apareció por primera vez en 2006[12] para describir un conjunto de nuevas arquitecturas de redes de neuronas artificiales que eran capaces de aprender mucho mejor que las arquitecturas tradicionales o planas. Este nuevo tipo de redes de neuronas se diferencian principalmente de las arquitecturas tradicionales en la utilización de un elevado número de capas de tipo oculto que aplican diferentes tipo de operaciones no lineales que permiten transformar la información con el fin de extraer nuevas características. Además la utilización de un número tan elevado de capas permite representar la información utilizando diferentes niveles de abstracción simplificando el proceso de clasificación y/o permitiendo la creación de nuevas características que permiten aprender ciertos conceptos imposibles para las arquitecturas tradicionales. Este nuevo tipo de arquitecturas basadas en la utilización de múltiples combinaciones de capas supuso un salto cualitativo en la creación de modelos que eran capaces de manejar conjuntos de datos con una elevadísima dimensionalidad y/o representaciones paramétricas no lineales. Debido a este tipo de arquitecturas comenzó a ser posible identificar, de manera hasta cierto punto eficiente, objetos, personas en base a características muy específicas mediante la utilización de imágenes en bruto (redes de neuronas profundas convolucionales) o procesar lenguaje tanto hablado como escrito con el fin de crear sistemas de traducción en tiempo real (redes de neuronas profundas recurrentes). A continuación se describe de forma detallada el funcionamiento de las redes de neuronas convoluciones las cuales son probablemente el tipo de red de neuronas de Aprendizaje Profundo más utilizadas.

3.1 Redes de neuronas convolucionales

Una red de neuronas convolucional (Convolutional Neural Networks CNN, en sus siglas en inglés) es un tipo de red de neuronas artificial que permite construir modelos mediante la utilización de aprendizaje supervisado. Este tipo de red de neuronas intenta imitar la estructura y funcionamiento del cortex visual del ojo humano con el fin de poder extraer información de la imágenes en bruto y conseguir detectar objetos, personas, etc. Para ellos las redes convolucionales utilizan diferentes tipos de capas ocultas especializadas en identificar ciertos tipos de información siguiente un flujo específico en el tratamiento de la información desde características más generales acerca de la imagen (línea, bordes, colores, etc.) hasta

características más específicas (rasgos faciales, caras, patrones en el rostro, etc.) construidas por combinación de las más generales. En la **Figura 4** se presenta un ejemplo de las diferentes capas de una red profunda de neuronas convolucional formada por 19 capas donde se aplican diferentes operaciones que nos permiten extraer características de las imágenes insertadas como entrada y utilizar todas estás características para construir una clasificación utilizando las dos ultimas capas denominadas Fully Connected (FD). En una red convolucional se pueden identificar dos procesos: (1) uno de extracción de características; y (2) otro de clasificación.

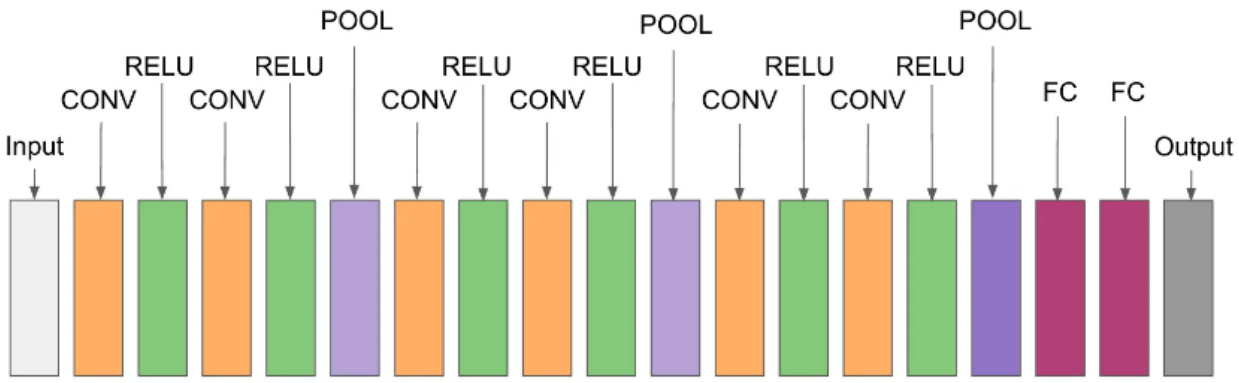


Figura 4: Distribución de las diferentes capas de una red de neuronas convolucional

3.1.1. Extracción de características

El proceso de extracción de características permite extraer información de la imágenes comenzando por la extracción de información general que es común a cualquier tipo de imagen, como por ejemplo líneas, colores, trazos, bordes hasta características más específicas que están más cercanas a los diferentes elementos que queremos identificar en las imágenes, como por ejemplo, caras, formas específicas, símbolos, etc. Para poder extraer este tipo de información se introducen en la red una serie de capas que permiten extraer este tipo de información mediante la aplicación de operaciones matemáticas las cuales permiten focalizarse en el proceso de extracción con el fin de identificar ciertos patrones específicos. Las capas más comunes utilizadas en una red de neuronas profunda de tipo convolucional son las capas convoluciones y de subsampling.

Representación de la información

La primera etapa del proceso consiste en transformar una imagen de entrada en neuronas de la capa entrada para ello se utiliza la estructura en forma de matriz de píxeles de la propia imagen.

Es decir, cada pixel de la imagen se asigna a una neurona de entrada. Para la explicación de los diferentes procesos vamos a utilizar una imagen de 7x7x1, es decir, nuestra imagen tendrá 49 píxeles y un canal de color (escalada de grises) lo que supondría que tendríamos 49 neuronas en la capa de entrada. En el caso de que estuviéramos utilizando una imagen en color con tres canales (Rojo, Verde y Azul) tendríamos 7x7x3 = 147 neuronas en la capa de entrada. En la **Figura 5** se presenta como las imágenes en color se dividen en tres imágenes cada una correspondiente a uno de los tres canales del RGB. Cuando se utilizan este tipo de imágenes como entrada de la red de neuronas el número de neuronas de la capa e entrada debe multiplicarse por tres.

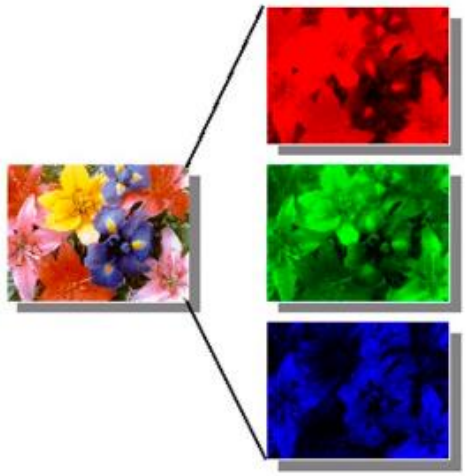


Figura 5: Ejemplo de cómo una imagen se divide de las imágenes correspondientes a los tres canales de color (RGB)

Capa convolucional

Las capas convolucionales son las capas que producen la verdadera extracción de características en este tipo de redes. Para ellos se utilizan operaciones de tipo matricial, denominadas convoluciones, que producen modificaciones en los píxeles con el fin de resaltar cierto tipo de información en la imagen con el fin de extraer nuevas características. Una convolución es una transformación de tipo local que se aplica sobre una matriz de pixeles donde el valor del pixel resultantes tras la aplicación de la convolución depende de la vecindad local de cada pixel, la cual viene definida por el filtro (kernel) que se utiliza para definir la vecindad y producir la convolución.

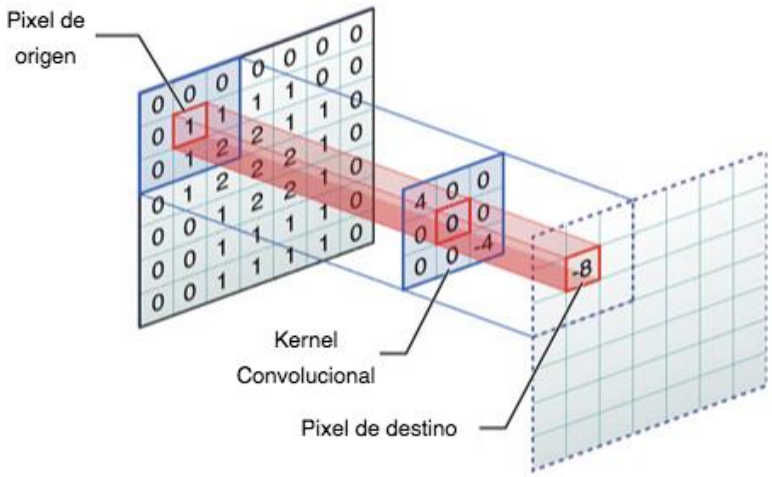


Figura 6: Ejemplo de funcionamiento de un convolución aplicada mediante un kernel 3x3.

En la **Figura 6** se presenta un ejemplo de un proceso de convolución sobre una imagen a la que se le aplica un filtro de tamaño 3x3. Este proceso consiste en transformar el valor de cada uno de los píxeles mediante la aplicación de un filtro que realiza un producto escalar. Esta operación se aplica sobre todas las neuronas de entrada con el fin de generar una nueva matriz de salida. Aunque realmente las capas de convolución no sólo aplican un único filtro sobre su neuronas de entrada sino que aplican un conjunto de filtros que producen la generación en un conjunto de imágenes cada una de las cuales ha sufrido un proceso mediante el cual se han resaltados ciertos aspectos de la imagen. Es decir, si tuviéramos una imagen de 7x7x1 (ancho, alto y canal) y aplicáramos un conjunto de 32 filtros generaríamos 32 nuevas imágenes que supondrían 1568 neuronas en la siguiente capa oculta. Además este ejemplo se ha descrito sobre sólo un canal. Las imágenes están formadas por tres canales (RGB) de forma que al aplicar un filtro deberíamos aplicarlo sobre los tres canales de la imagen de forma que el filtro 3x3 de la **Figura 6** se convertiría en un filtro 3x3x3.

Capa de subsampling

Las capas convolucionales incrementan el número de neuronas debido al proceso de resaltado que provocan los filtros, si continuáramos incluyendo más capa convolucionales incrementariamos el número de neuronas hasta un punto que sería incontrolable, por lo que con el fin de reducir el número de neuronas tras la aplicación de varias capas convolucionales se suele aplicar una capa de subsampling. Este tipo de capa intenta reducir el número de neuronas mediante un proceso de combinaciónes que intentan mantener las características que han sido

resaltadas durante las convoluciones pero utilizando un número menor de neuronas. Las capa de subsampling más utilizadas son la cada de tipo Pooling (max, mediun, etc.).

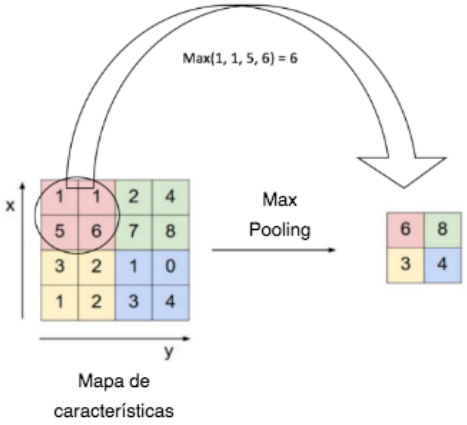


Figura 7: Ejemplo de funcionamiento de un operación de max-pooling aplicada mediante un kernel 2x2.

En la figura se presentan una operación de Max-polling mediante la utilización de un filtro de 2x2. Este filtro se aplica sobre todas las matrices (imágenes) generadas en la capa anterior transformando 4 neuronas en una. En el ejemplo se aplica una filtro de tipo Max que selecciona aquel pixel que tiene el mayor valor. Este proceso de convolución y sampling se suele repetir múltiples veces con el fin de extraer la mayor cantidad de la imagen.

3.1.2. Clasificación

Tras el proceso de extracción de características se produce un aplanamiento de la información que transforma las neuronas de la última capa de extracción en una única capa de neuronas mediante una capa de tipo Fully Connected la cual puede estar conectada a otro tipo de capa o directamente a la salida mediante una capa de tipo Softmax que se usa normalmente para conectar la última capa oculta con la capa de salida cuyo número de neuronas será igual al número de clases entre las cuales estamos clasificando. En la Figura 8 se presenta el funcionamiento completo de una red convolucional formada por los dos procesos descritos previamente.

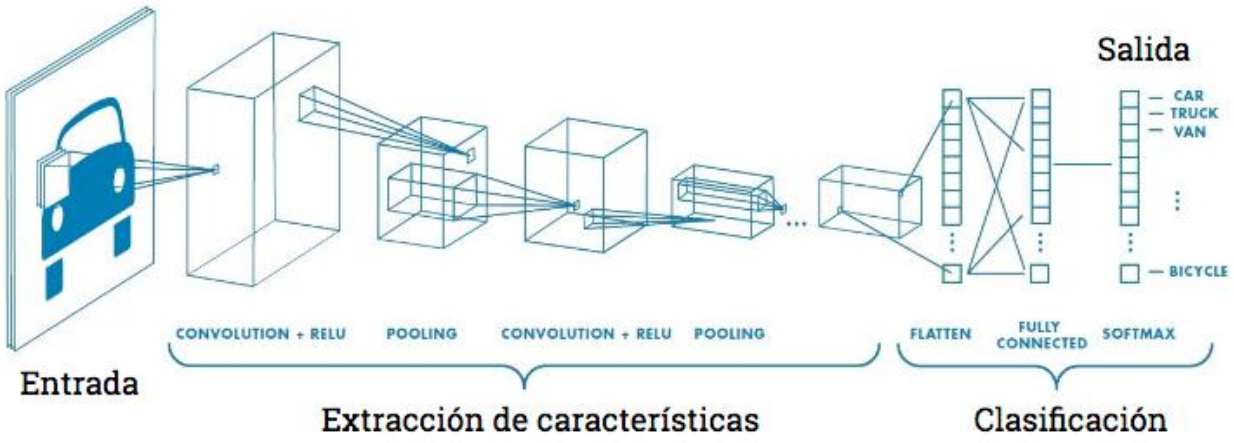


Figura 8: Funcionamiento completo de una red convolucional para la clasificación de imágenes de diferente tipos de vehículos

4. APRENDIZAJE POR REFUERZO

El aprendizaje por refuerzo [5] [13] es una de las familias del aprendizaje automático basado en el aprendizaje conductista que permite construir modelos de aprendizaje basados en políticas. Es decir, este tipo de algoritmos tratan de identificar la secuencia de acciones que debe escoger un agente en base a la información del entorno maximizando algún tipo de recompensa que se obtendrá tras la aplicación de las acciones. Es decir, este funcionamiento recuerda el proceso de aprendizaje que utilizamos lo seres vivos para aprender o enseñar ciertos conceptos relacionados con la interacción con el entorno de manera automática. Como por ejemplo, cuando los humanos aprendemos a caminar de manera bípeda donde recibimos un “refuerzo negativo” cuando nos caemos debido a que no hemos conseguido alcanzar nuestra posición objetivo, o en el peor caso nos hemos caído y hemos sufrido un golpe, o un “refuerzo positivo” cuando hemos sido capaces de movernos correctamente hasta la posición objetivo. El proceso de aprendizaje va adaptando nuestro organismo de forma que seamos capaces de mantener el equilibrio y mover de forma correcta las articulaciones del cuerpo para desplazarnos de la forma correcta. Como se puede observar el modo de funcionamiento de esta familia de métodos de aprendizaje difiere de los métodos de tipo supervisado y no supervisados a los cuales se les proporciona todo el conjunto de instancias de entrenamiento a priori con el fin de construir el modelo. En cambio los métodos de aprendizaje por refuerzo obtienen las diferentes instancias de entrenamiento de manera activa interactuando con el entorno en diferentes iteraciones. En la **Figura 9** se presenta un diagrama que describe el funcionamiento básico de un sistema basado

en Aprendizaje por Refuerzo el cual está formado por dos entidades básicas, el agente y el entorno, que interactúan entre sí por medio de las acciones que ejecuta el agente en el entorno.

- Agente: El agente es la entidad principal del proceso de Aprendizaje por Refuerzo que interactua con el entorno obteniendo información y produciendo variaciones en el mediante la aplicación de acciones.
- Acción: Es la acción que es ejecutada por el agente en el entorno con el fin de producir una variación en el. Las lista de posibles acciones depende del estado del entorno, debido a que no es posible aplicar cualquier acción en cualquier estado. En el caso de un videojuego bidimensional como el Pacman, la lista de posibles acciones puede incluir moverse hacia la derecha, hacia la izquierda, hacia arriba, hacia abajo o quedarse quieto.

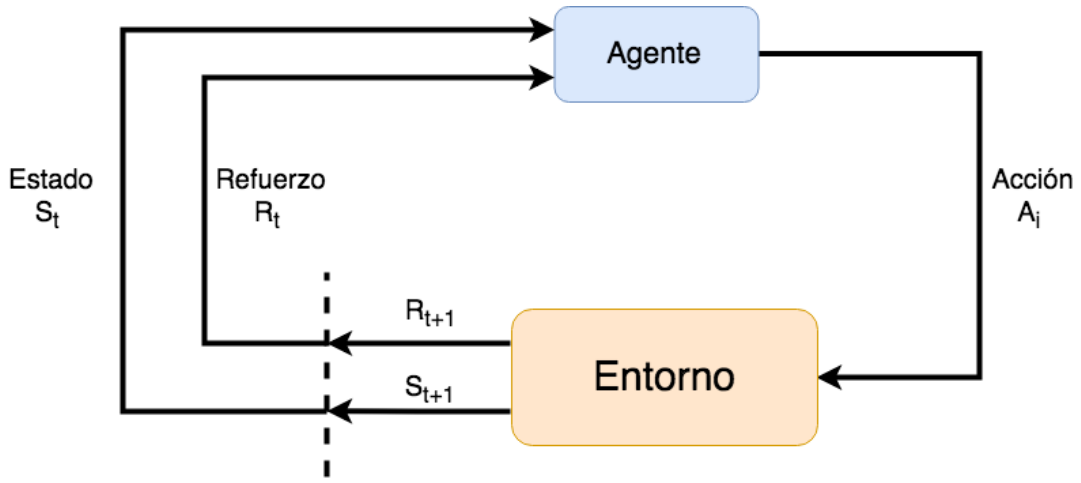


Figura 9: Arquitectura básica de un proceso de Aprendizaje por Refuerzo

- Estado: El estado es la representación completa del entorno en un instante específico de tiempo. Esta información incluye toda la información referente al entorno, así como la información que representa al agente.
- Refuerzo: El refuerzo es un valor numérico de tipo real que se obtiene tras la ejecución de una acción en el entorno. Este valor es utilizado como medida para evaluar si la ejecución de una acción ha sido positiva o negativa para el agente dependiendo del valor de refuerzo obtenido.
- Entorno: Es la representación virtual del mundo en la cual interactua el agente. El entorno almacena el estado actual el cual es modificado por las acciones (entradad) ejecutadas

por el agente generando a continuación como salida el nuevo estado del entorno y la recompensa obtenida tras la ejecución de la acción.

El objetivo de un proceso de aprendizaje por refuerzo consiste en resolver un problema de decisión secuencial que implica la selección de un conjunto de acciones (decisiones) cuyo resultado (refuerzo) no se conoce hasta el final. Para ello el sistema de aprendizaje deberá construir un modelo en forma de política $\pi(s, a)$ que deberá sugerir la mejor acción a ejecutar en el entorno. El cual suele ser modelado mediante un Proceso de Decisión de Markov [6] (Markov Decission Process, MDP en sus siglas en inglés) completamente observable si es posible obtener información completa del entorno, pero existe algún tipo de incertidumbre asociada al resultado de la acción, es decir existe una probabilidad de que la acción no se ejecute de manera correcta y produzca un estado diferente al esperado. Aunque existe la posibilidad de que el entorno sea parcialmente observable, lo que producirá un Proceso de Decisión de Markov Parcialmente Observable [7] (Partially Observable Markov Decission Process, POMDP en sus siglas en inglés) cuando además de incertidumbre en la ejecución de las acciones existe también incertidumbre en la información referente a los estados, es decir el agente no es capaz de observar de manera completa el estado, sólo es capaz de obtener información parcial.

4.1 Procesos de decisión de Markov

Un proceso de decisión de Markov (Markov Decision Process, MDP en sus siglas en inglés) es un tipo de problema de decisión secuencial aleatoria dependiente del tiempo en el cual se cumple la propiedad de Markov [8]. Este propiedad define que las recompensas que se pueden obtener en el futuro mediante la aplicación de acciones en un determinado estado sólo depende de dicho estado y no de los anteriores. Es decir, los refuerzos que vayamos a obtener tras la aplicación de las diferentes acciones disponibles para un estado sólo dependerán del estado actual del entorno y no de los estados anteriores por los que se ha transitado hasta llegar al estado actual. Un proceso de decisión de Markov se puede definir de manera formal como una tupla MDP = (S, A, T, R):

- S es el conjunto finito de estados, donde $s_t \in S$ es el estado en el que se encuentra el agente en el instante temporal t. De manera formal el conjunto de estados S del entorno se define como un conjunto finito $S = \{s_1, s_2, \ldots, s_n\}$, siendo n el tamaño del espacio de estados. La forma en la que se modela el entorno depende de cada problema, por lo que no hay una representación específica como ocurría con el aprendizaje supervisado y no

supervisado donde la información se modelada como un vector de atributos. En cambio cuando aplicamos aprendizaje por refuerzo podemos modelar la información de la manera que más se adapta al problema a resolver. Normalmente la información suele modelarse como una matriz multidimensional ya que muchos de los problemas tratados consisten en aplicar una serie de acciones en un entorno multidimensional.

- A es un conjunto finito de acciones, cada una de las cuales sólo puede ser aplicada en un determinado estado. Es decir sea $a_t \in A(s_t)$ el conjunto de acciones que puede ser aplicadas en el estado s_t. Es posible que existan estados en los cuales no es posible aplicar ningún tipo de acciones, siendo estos estados considerados como sumideros o estados muertos (dead-end), ya que una vez alcanzados es imposible resolver el problema.
- La función $T(s_t, a, s_{t+1})$ es una función de probabilidad tal que $\mathrm{T}: \mathrm{S} \times \mathrm{A} \rightarrow \mathrm{P}(\mathrm{S})$ que define la probabilidad de que se produzca una transición del estado s_t al estado s_{t+1} tras las aplicación de la acción a_t.
- R es una función de refuerzo tal que $\mathrm{R}: \mathrm{S} \times \mathrm{A}$ donde $\mathrm{R}(s_t, a_t)$ es el refuerzo recibido tras ejecutar la acción a_t en el estado s_t.

Las funciones de transición y recompensa son los elementos que define el modelo del MDP. Un MDP puede ser representado mediante un diagrama de transición de estados donde cada nodo se corresponde con un estado y los arcos (dirigidos) denotan las diferentes acciones que permiten transitar de un estado a otro. Esta representación gráfica normalmente incluye información referente a la recompensa que se obtiene al transitar a un estado y de la probabilidad de cada transición. En la **Figura 10** se presenta un ejemplo del diagrama de un MDP. Cada uno de los arcos muestra el identificador de la acción que produce la transición y dos valores numéricos. El primero representa el refuerzo que se obtiene al transitar de un estado a otro utilizando esa transición y el segundo representa la probabilidad de que se produzca la transición. Por el ejemplo, la aplicación de la acción a1 en estado S2 da lugar a dos transiciónes donde la probabilidad de ambas tiene que ser uno. En este caso con probabilidad igual a p = 0,2 (20%) se transitaría al estado S0 y con probabilidad p = 0,8 (80%) se transitaría al estado S1. Normalmente los MDP poseen al menos un estado denominado terminal en su conjunto de estados que normalmente es utilizado como meta u objetivo de forma que se pueda resolver el problema que representa el modelo.

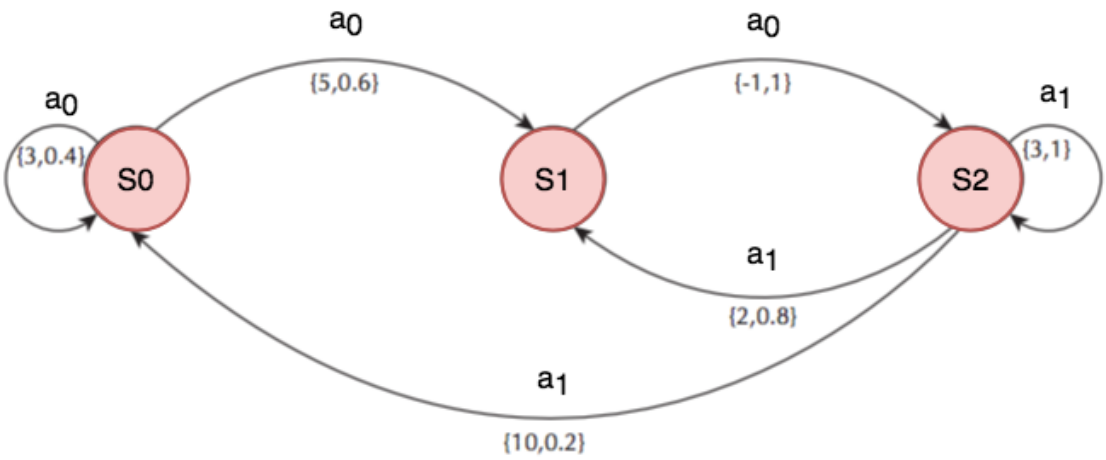


Figura 10: Representación gráfica de un Proceso de Decisión de Markov no determinista

La solución a un MDP suele consistir en identificar la política que permite alcanzar uno de los posibles estados terminales o metas de manera optima. De forma que el objetivo en este tipo de modelos consiste en identificar dicha política. Una política es una función $\pi : S\ x\ A$ que realiza un mapeo entre los estados y las acciones mediante sus distribuciones de probabilidad.

4.2 Funciones de valor

El objetivo al intentar resolver un problema modelado mediante un MPD consiste en encontrar la política óptima o aquella que se encuentre lo más cerca posible de esta. Para conseguir encontrar la política óptima es necesario definir algún tipo de función que permita estimar como la calidad de los estados desde la perspectiva del agente, es decir, necesitamos algún tipo de medida que nos permita medir como de bueno es para el agente llegar a un determinado estado en el entorno. Para calcula ese valor de calidad es posible utilizar dos funciones:

- La función Q, denominada función de valor estado-acción, que estima como de bueno es aplicar una acción en un determinado estado. De manera formal se puede definir la función Q de una política π, tal que $Q^{\pi} : S \times A \rightarrow R$, es igual a la recompensa obtenida cuando a partir de un determinado estado s y una determinada acción a, se sigue la política π:

$$Q^{\pi}(s,a) = \sum_{i=0}^{\infty} \gamma^{i} R(s_i, a_i) = R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \cdots$$

Esta ecuación puede ser refinada con el fin de obtener otra expresión que es la que realmente se utiliza en los diferentes algoritmos. De modo que la función Q puede ser expresada también en base a la siguiente ecuación:

$$Q^{\pi}(s,a) = R(s,a) + \gamma Q^{\pi}(s', a')$$

- La función V, denominada función de valor estado, estima como de bueno es que el agente se encuentre en un determinado estado. De manera formal se puede definir la función V de una política π, tal que $V^{\pi} : S \rightarrow R$, es igual a la recompensa obtenida cuando a partir de un determinado estado s se sigue la política π:

$$V^{\pi}(s) = \sum_{i=0}^{\infty} \gamma^{i} R(s_i, \pi(s_i))$$

A pesar de cada una de esta funciones de valor se centra en un aspecto diferentes del modelo, estas pueden ser relacionadas entre sí en base a la política que sigue, que es el elemento común de ambas, mediante la siguiente ecuación:

$$V^{\pi}(s) = Q^{\pi}(s, \pi(s))$$

Ambas ecuaciones de valor nos permiten calcular una política óptima. Aunque en el caso de la función de valor V^{π} es necesario conocer el modelo del MDP de forma completa, es decir es necesario conocer las funciones de transición y de recompensa. Esto es debido a que la función de valor V^{π} sólo contiene información sobre la calidad de los estados, por lo que para poder determinar la acción óptima es necesario utilizar información que es provista por las funciones de transición y recompensa. En cambio, la función Q^{π} incluye información sobre los estados y las acciones lo que permite calcular la política óptima sin necesidad de conocer el modelo del MDP. Debido a esta importante diferencia entre la funciones de Q y V hace que la mayoría de

los algoritmos de Aprendizaje por Refuerzo utilicen funciones de valor de estado-acción debido a que asumen que el modelo del MDP es total o parcialmente desconocido. A continuación describiremos los primeros algoritmos que se desarrollaron para utilizar las funciones de valor y calcular la política óptima del Modelo de Decisión de Markov. Cabe destacar que en la descripción de estas funciones hemos asumido del el Modelo de Decisión de Markov es determinista, es decir las acciones que se aplican sobre los estados siempre producen la misma transición. En el caso que necesitemos trabajar con sistemas no deterministas será necesarios aplicar una serie de variaciones en las ecuaciones matemáticas con el fin de poder controlar la falta de determinismo en las acciones. Pero en este curso no vamos a tratar con MDPs no deterministas. A continuación se presentan diferentes algoritmos que tratan de calcular la política óptima en situaciones en las cuales tienen información parcial o total del modelo descrito por el MDP.

4.3 Algoritmos tradicionales

Uno de los principales problemas que existen a la hora de resolver un problema modelado mediante un MDP o un POMDP es el conocimiento completo o parcial del modelo. Dependiendo de que tipo de conocimiento se posea, el problema podrá ser resuelto mediante la utilización de unas u otras técnicas. En el caso de que se tenga un conocimiento completo del modelo, es decir, se conocen la función de transición y la función de recompensa, es posible utilizar técnicas de programación dinámica. Las técnicas de programación dinámica no suelen son considerados como técnicas de Aprendizaje por Refuerzo debido a que tiene conocimiento completo del modelo y no necesitan aprenderlo, pero fueron el germen de los verdaderos algoritmos de aprendizaje por refuerzo por lo que es necesario describir como funcionan para entender cuales son las bases matemáticas de los diferentes algoritmos de aprendizaje por refuerzo los cuales se basan en la utilización de funciones de valor de tipo Q. Además los diferentes métodos que serán descritos a continuación asumen que los conjuntos de estados y acciones son finitos y lo suficientemente pequeños para ser almacenados en algún tipo de estructura de datos.

4.3.1. Iteración de políticas

El algoritmo de iteración de políticas[9] (Policy Iteration, PI en sus siglas en inglés) es un algoritmo de programación dinámica que realiza variaciones en la política en cada iteración con el fin de mejorarla hasta que la política converja. El funcionamiento de este algoritmo se basa en tres fases:

- Inicialización: En esta fase se produce la inicialización de las estructuras en las cuales se almacena la política generándose una política aleatoria o basada en algún tipo de algoritmo de generación que minimiza el proceso de mejora de la política ya que parte de una política más cercana a la política óptima. Esta fase sólo se ejecuta en la primera iteración.
- Evaluación: Es esta fase se produce una evaluación de la política mediante el cálculo de la función de valor. Este proceso se denomina como predicción de la función de valor y para ello se utilización la ecuación de Bellman.

$$Q_{t+1}^{\pi}(s,a) = R(s,a) + \gamma\, Q_t^{\pi}(T(s,a), \pi(T(s,a)))$$

- Mejora: En esta fase se aplican una serie de variaciones en política con el fin de mejorarla mediante la utilización de la función de valor calculada en la fase anterior. Esta serie de mejoras tratan de generar una nueva política con un comportamiento más cercano al de la política óptima. Esta proceso suele realizarse mediante la utilización de un algoritmo avaricioso (greedy) que escoge para cada estado la acción con mayor valor de la función Q^{π}. Si consideramos que en la iteración anterior t se ha evaluado la política π_t, la nueva política mejorara se generaría en base a la siguiente expresión:

$$\pi_{t+1}(s) = \arg\max_{a} Q^{\pi_t}(s,a)$$

Si la nueva política generada mediante el proceso de mejora no presenta ninguna mejora respecto a la anterior se considera que la política ha convergido y por lo tanto es la política óptima. En caso contrario se almacena aquella política de entre las dos cuyo comportamiento sea más similar a la política óptima y se vuelve a la fase dos del algoritmo.

4.3.2. Value Iteration

El algoritmo de iteración de valor[10] (Value Iteration, VI en sus siglas en inglés) es un algoritmo de programación dinámica que realiza variaciones en la función de valor en cada iteración con el fin de mejorarla hasta que encontrar la función de valor óptima. Mientras que el algoritmo PI

utilizaba la ecuación de Bellman para evaluar una política que posteriormente era mejorada, el algoritmo VI utiliza la ecuación de optimalidad de Bellman para intentar calcular directamente la función de valor óptima. El funcionamiento de este algoritmo se basa en tres fases:

- Inicialización: En esta fase se produce la inicialización de las estructuras en las cuales se almacena la función de valor generándose un función de valor aleatoria. Esta fase sólo se ejecuta en la primera iteración.
- Mejora: En esta fase se aplican una serie de variaciones en la función de valor actual con el fin de mejorarla mediante la utilización ecuación de optimalidad de Bellman. Esta serie de mejoras tratan de encontrar una nueva función de valor que está más cercana al función óptima. Si consideramos que en la iteración anterior t se ha evaluado la política π_t, la nueva política mejorara se generaría en base a la siguiente expresión:

$$Q_{t+1}(s,a) = R(s,a) + \gamma \max_{a} Q_t\,(T(s,a),a')$$

4.4 Q-Learning

El algoritmo Q-Learning[14] es una algoritmo de Aprendizaje por Refuerzo que trata de aprender una política mediante la utilización de experiencias que son generadas mediante un proceso de exploración del entorno. A diferencia de los algoritmos descritos anteriormente este no necesita tener conocimiento completo del modelo, es decir no necesita conocer las probabilidades de éxito de las acciones. Las diferentes instancias de entrenamiento son experiencias las cuales se representan como una tupla e = (s, a, s', r) donde: s es el estado actual del entorno; a es la acción que será aplicada sobre el entorno; s' es el estado resultante tras la aplicación de la acción a; y r es el refuerzo obtenido tras la aplicación de la acción a en el estado s. Estas experiencias de entrenamiento son utilizadas para construir la tabla Q que almacenada la política aprendiza durante el proceso de entrenamiento.

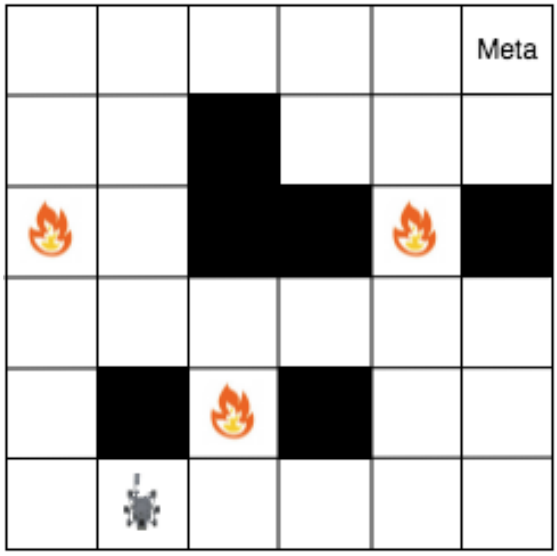


Figura 11: Estado inicial de problema a resolver mediante el aprendizaje por refuerzo

Esta tabla es un mapa donde se almacena el máximo valor de refuerzo que puede ser obtenido al aplicar una acción en un estado. De forma sencilla esta tabla nos indica cual es la mejor acción que puede ser aplicada en cada estado. En la **Figura 11** se presenta el estado inicial de un posible entorno en el cual un robot se debe mover hasta la localización denominada como meta. Para ello el robot tendrá que desplazarse por el entorno evitando las zonas de fuego y los muros con el fin de llegar a la meta. En este caso el objetivo del proceso de aprendizaje es crear la tabla Q que permita al robot alcanzar la meta sin caer en ninguna de las casillas que están ardiendo. En este caso la tabla Q está formada por un conjunto de 30 estados que se corresponde con las diferentes posiciones a las cuales se puede desplazar el robot y 4 posibles acciones que permitirán al robot moverse hacia arriba, abajo, izquierda y derecha, por lo que tendremos una tabla de 30x4 posibles valores de refuerzo. Como se puede observar en la imagen, existen estados en los cuales no es posible aplicar todas las acciones lo que supondrá que el valor de refuerzo de esos pares estado-acción tendrán un valor 0 y no deberían ser seleccionados nunca.

El proceso de aprendizaje

El proceso de aprendizaje del Algoritmo Q-Learning consiste en conjunto de episodios cada uno de los cuales se corresponde con una ejecución completa del algoritmo con el fin de actualizar el mayor número posible de casillas de la tabla Q consiguien que los valores de esta converjan de la manera más rápida posible. Cada uno de estos episodios ejecuta un algoritmo de 5 fases:

- Inicialización: Esta es la primera fase del algoritmo se inicializa la tabla Q, la tasa de aprendizaje y el factor de descuento. Normalmente el valor de inicio es 0, pero existen

configuraciones en la cuales se asigna un valor aleatorio entre 0 y 1. Además también es posible utilizar una tabla Q generada previamente con el fin de acelerar el proceso de aprendizaje. Esta fase sólo se ejecuta una vez al inicio de cada episodio.

- Selección: Esta fase consiste en seleccionar la mejor acción de la tabla Q y ejecutarla, debido a que el comienzo del proceso de aprendizaje todas los pares estado-acción tienen valor 0 por lo que se produce una selección aleatoria de las acciones. Este proceso de selección aleatoria es la que se conoce como proceso de exploración ya que el agente realiza una exploración del entorno debido a que no tiene información sobre el. A medida que se va incluyendo información sobre el entorno en la tabla Q, la exploración se convierte en explotación debido a que comienza a explorarse sólo aquellas acciones que tienen unos recompensas mayores.
- Ejecución: En esta fase se realiza la ejecución de la acción seleccionada en el entorno generándose un nuevo estado como consecuencia de la ejecución de la acción ejecutada en estado previo.
- Calculo: En esta fase se realiza el calculo del nuevo valor de refuerzo mediante la función de valor de tipo Q cuya información es posteriormente almacenada en la tabla Q.

$$\underset{1}{Q(s_t,a)} = \underset{2}{Q(s_t,a)} + \underset{3}{\alpha}\left[\underset{4}{R(s_t,a)} + \underset{5}{\gamma}\underset{6}{\max Q(s_{t+1},a)} - \underset{2}{Q(s_t,a)}\right]$$

Figura 12: Ecuación de actualización de la tabla Q

1. Nuevo valor en la tabla Q para el estado y la acción seleccionada.
2. Es el antiguo valor almacenado en la table Q para el estado y la seleccionada.
3. La tasa de aprendizaje (learning rate) es una variable cuyo valor está comprendido entre 0 y 1 e indica cuando vamos a tener en cuenta lo que aprendamos en la nueva experiencia producida por la ejecución de la acción a en estado s_t. Cuando su valor es cercano a 1 significa que vamos a tener en cuenta las nuevas experiencias y cuando el valor es cercano a 0, no se va a producir casi ningún cambio en los valores de la tabla Q, debido a que el agente no está aprendiendo sólo seleccionando aquellas acciones con mejor valor en la tabla Q.
4. Refuerzo que se obtiene por ejecutar la acción a en el estado s_t.

5. El factor de descuento (discount rate) es una variable cuyo valor está comprendido entre 0 y 1 e indica la importancia de las acciones ejecutadas a largo plazo. Cuando su valor es cercano a cero 0 el algoritmo sólo tiene en cuenta los refuerzo inmediatos, en cambio si el valor es próximo a 1 algoritmo se centra más en lo que pueda ocurrir a largo plazo.
6. Es el maximo valor de recompensa que se obtendrá al transitar al estado s_{t+1}. Este valor se calcula como el maximo de todos los valores de recompensa que se encuentran almacenados en la tabla Q para el estado s_{t+1}.

- Actualización: La fase actualización modifica los valores de la tabla Q en base a la ecuación descrita en la **Figura 12** y se vuelva a la fase 2.

Este modelo de funcionamiento se producirá durante un numero controlado de iteraciones debido a que es muy complicado identificar cuando se han conseguido obtener los valores óptimos de la tabla. Es posible identificar cuando se han obtenido los valores óptimos de la tabla Q analizando los cambios que se producen en cada una de las iteraciones, si durante un número determinado de iteraciones no se produce ningún cambio en la tabla, se podría intuir que el proceso de aprendizaje ha terminado. Pero en casos en los cuales la tabla Q a generar es de un tamaño muy elevado es posible que sea muy complicado identificar este tipo de situaciones, por lo que lo este tipo de estrategias no son muy utilizadas. Además el algoritmo Q-Learning tiene dos parámetros cuyos valores pueden cambiar el proceso de aprendizaje por lo que es interesante aprender con diferente valores de estos parámetros.

4.5 Deep Reinforcement Learning

La aparición de las técnicas de aprendizaje profundo supuso una revolución debido a que permitían crear arquitectura muchos más complejas que permitían resolver de complejidad muy elevada relacionadas con los procesos de visión y habla. Pero esta revolución no se detuvo en la utilización de redes de neuronas profundas para la construcción de modelos basados en aprendizaje supervisado o no supervisado, sino que ha terminado por alcanzar al aprendizaje por refuerzo. Uno de los principalmente problemas de los algoritmos basados en aprendizaje por refuerzo es la complejidad del modelo el cual puede estar formando por millones de estados y acciones lo que supone en el caso del algoritmo Q-Learning la necesidad de una tabla Q que tendrá un tamaño excesivo. La necesidad de mantener en memoria una estructura de datos de una tamaño tan elevado supone un gran problema a la hora de construir modelos en entorno

muy complejos. La posible solución a este problema consiste en sustituir la tabla por algún otro tipo de estructura que tenga un tamaño menor o utilizar algún tipo de algoritmo que no tenga la necesidad de un estructura de datos con grande. Una de las posibles alternativas es la utilización de las redes de neuronas, las cuales pueden ser utilizadas para construir un modelo que permita predecir el refuerzo de todas las posibles acciones que pueden ser ejecutadas en un determinado estado que se incluye como entrada a la red.

En base a esta idea es posible construir una red de neuronas profunda que actué como un predictor del refuerzo de la acciones, es decir, la red utilizaría como entrada un representación del estado del entorno y generaría como salida un vector donde cada uno de los elementos se correspondería con el valor de refuerzo de cada una de las acciones que pueden aplicarse en el estado. La **Figura 13** presentan dos posibles arquitecturas utilizadas para la construcción de este tipo de redes de neuronas. La estructura que se encuentra a la izquierda funcionaría de manera similar a un regresor, dada un entrada compuesta por el estado del mundo y una posible acción que ser ejecutada se obtiene un valor numérico que se corresponde con la recompensa que se obtendría al ejecutar esta acción. Este tipo de configuración es muy útil para problemas con un número muy elevado de acciones donde sólo un pequeño conjunto de ellas pueden ser aplicadas en el estado. El modelo construido mediante una red con esta arquitectura permitiría obtener el valor de refuerzo que se obtendría por la ejecución de una acción específico, por lo que sería necesario un sistema que utilizara la red para obtener el valor de cada una de ellas y a continuación elegir la acción que obtuvo el mejor valor. En cambio, la arquitectura presentada en a la derecha de la **Figura 13** funcionaría de manera similar a un clasificador ya que generaría como salida un array con todos las posibles acciones que pueden ser ejecutadas en el juego generando un valor de refuerzo para cada una de ellas de forma que se elegiría aquella acción el valor de refuerzo más grande. Este tipo de redes son muy útiles para resolver problemas en los cuales no existe un número muy elevado de posibles acciones o donde no es posible identificar cuales las acciones que pueden ejecutarse en un determinado estado. El modelo construido mediante una red con esta arquitectura permitiría obtener el valor de refuerzo de todas las acciones que pueden ejecutarse en el entorno, por lo que sería necesario un sistema que seleccionara la acción con mayor valor.

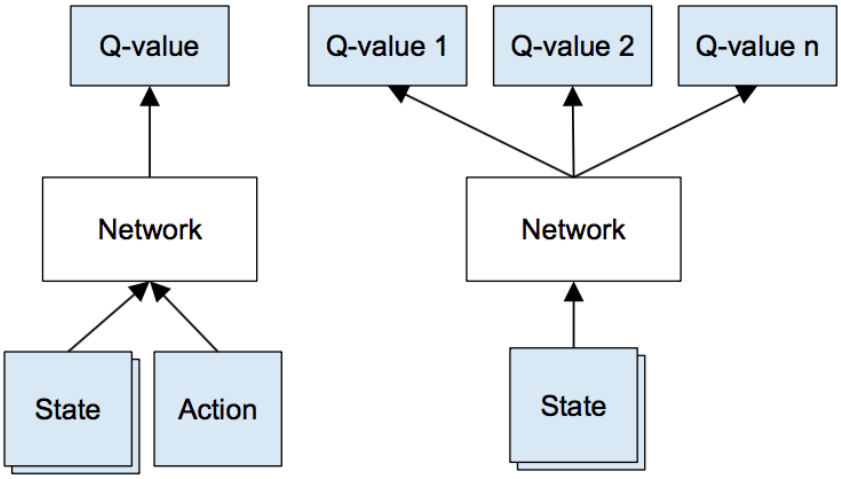


Figura 13: Ejemplos de las posibles estructuras de la redes de neuronas basadas en Aprendizaje Automático

Este tipo de redes de neuronas han sido denominadas como Redes de Neuronas Profundas de Aprendizaje por Refuerzo (Reinfocement Learning Deep NN) ya que permiten aprender un modelo que funciona de manera similar a como lo haría un algoritmo de aprendizaje por refuerzo pero utilizando una red de neuronas que actúa como un aproximador del refuerzo. La mayoría de las aproximaciones que han utilizado este tipo de técnica intentaban construir un agente que fuera capaz de jugar en un videojuego lo que le facilitaba la representación de la información ya que consistía en captar una imagen de la pantalla del juego y utilizar una variación de una red de neuronas profunda de tipo convolucional que realiza un preprocesado de la imagen con el fin de extraer características que permitan definir cual ese el posible refuerzo de cada uno de las posibles acciones de forma que se pueda elegir aquella con mayor refuerzo. La primera implementación desarrollada por los ingenieros de la empresa Deep Mind Technologies [11] donde demostraron como una máquina era capaz de aprender a jugar a los videojuegos de una Atari 2600 mediante el análisis de los pixeles de la imagen que representaban el estado del juego. Uno de los elementos más destacables de esta implementación es que había sido entrenada para jugar a siete diferentes videojuegos que utilizaban el mismo interfaz de control. La **Figura 14** presenta una representación parcial de la de una red similar a la utilizada para jugar con la videoconsola Atari 2600 la cual dispone una serie de capas convolucionales que extraen características de la imagen y una serie de capas fully connected que clasifican la información con el fin de generar la salida.

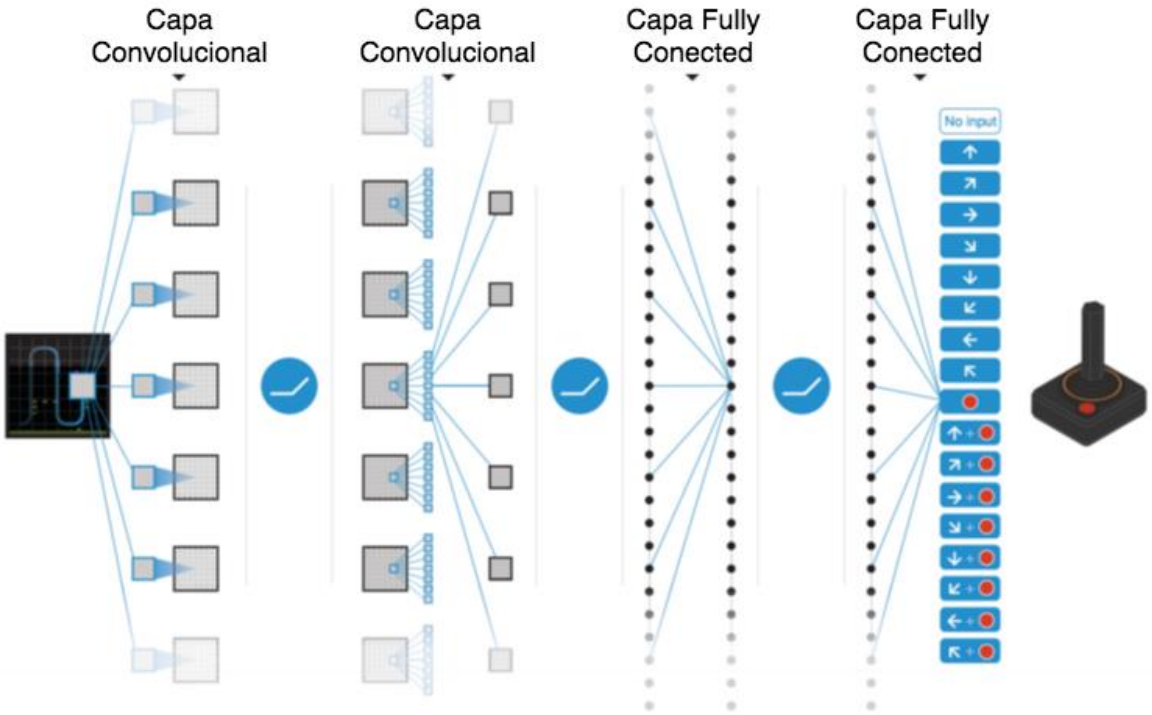


Figura 14: Ejemplo de la estructura de una Red de Neuronas de Aprendizaje Profundo. © DeepMind

5. CONCLUSIONES

En este tema se han presentan las tres familias restantes de métodos de aprendizaje automático. Uno de ellos utilizando conjuntos de datos muy complejos a partir de los cuales es muy complicado identificar características que permitan construir modelos (Aprendizaje Profundo); y dos que ofrecen diferentes algoritmos que son capaces de aprender modelos utilizando conjuntos de datos de entrenamiento no etiquetados (Aprendizaje no Supervisado) o mediante la interacción en tiempo real con el entorno (Aprendizaje por Refuerzo). Cada una de estas técnicas está especializada en aprender modelos en base a diferentes representaciones en la información desde la más básica definida como un vector de atributos o características hasta los pixeles de una imagen. En general, podemos indicar que todas estas familias ofrecen una amplia gama de métodos para la construcción de modelos de aprendizaje los cuales en combinación con las nuevas técnicas de extracción y manipulación de información de manera masiva van a permitir la resolución de muchísimos problemas a los cuales no podíamos enfrentarnos con la tecnología de hace unos años. En el próximo tema se describirá como funciona el ciclo de vida

de los datos y como las diferentes técnicas descritas en estos dos temas son utilizadas para construir modelos mediante la utilización masiva de datos gracias al Big Data.

6. REFERENCIAS

[1] MacQueen, J. B. (1967). Some Methods for classification and Analysis of Multivariate Observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. University of California Press. pp. 281-297.

[2] Hamerly, G. and Elkan, C. (2002). Alternatives to the k-medias algorithm that find better clusterings. Proceedings of the eleventh international conference on Information and knowledge management (CIKM).

[3] de Berg M., Cheong O., Van Kreveld M., and Overmars M. (2008). Computational Geometry: Algorithms and Applications (Tercera edición). Springer-Verlag Berlín Heidelberg.

[4] Kohonen, T. (1982). Self-Organized Formation of Topologically Correct Feature Maps. Biological Cybernetics. Volumen 43(1), pp 59-69.

[5] Richard S. S. and Andrew G. B. (1998) Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning). The MIT Press.

[6] Bellman, R. (1957). A Markovian Decision Process. Journal of Mathematics and Mechanics. Volumen 6.

[7] Åström, K. J. (1965). Optimal control of Markov processes with incomplete state information. Journal of Mathematical Analysis and Applications. Volumen 10. Pp 174-205.

[8] Richard B. (1958). Dynamic programming and stochastic control processes. Information and Control. Volumen 1(3). pp 228-239.

[9] Howard R.A. (1960). Dynamic Programming and Markov Processes. MIT Press, Cambridge, Massachusetts.

[10] Bellman R. (1957). Dynamic Programming. Princeton University Press.

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. (2013). Playing Atari with Deep Reinforcement Learning. NIPS Deep Learning Workshop.

[12] Goodfellow I., Bengio Y. and Courville, Aaron. (2016). Deep Learning. MIT Press.

[13] Mitchell T. M. (1997). Machine Learning. McGraw-Hill, New York.

[14] Dayan P. (1992) Technical note q-learning. Machine Learning, Volumen 292(3). Pp 279–292.