Computación, Machine Learning

Historia de las Matemáticas

DANIEL LÓPEZ GARCÍA
LOTHAR SOTO PALMA
ELENA TORO PÉREZ
Universidad de Granada
10 de enero de 2017

Índice

Índice de figuras

1. Introducción

Machine learning es una disciplina científica del ámbito de la Inteligencia Artificial que crea sistemas que aprenden automáticamente.

Pero, ¿qué significan las palabras aprender y automáticamente, en este contexto?

Aprender en este contexto quiere decir identificar patrones complejos en millones de datos. La máquina que realmente aprende es un algoritmo que revisa los datos y es capaz de predecir comportamientos futuros.

Automáticamente, también en este contexto, implica que estos sistemas se mejoran de forma autónoma con el tiempo, sin intervención humana.

Hoy en día, podemos observar la aplicación concreta de Machine Learning en varios campos que nos rodean, como son:

- **Automóviles:** Sistemas de piloto automático, Detección de fallos por reconocimiento externo de vibraciones...
- **Bancos:** Lectura de cheques y otros documentos, Detección de fraude en transacciones...
- **Electrónica:** Reconocimiento de voz, visión artificial...
- **Finanzas:** Asesoría de préstamos (aceptación/denegación de un crédito), Identificación de falsificaciones, Interpretación y reconocimiento de firmas...
- **Internet:** Compras on-line, Anuncios on-line, Selección de clientes potenciales basándose en comportamientos en las redes sociales, interacciones en la web...
- **Manufactura:** Inspección de calidad mediante sistemas visuales, Control de la producción y del proceso, Análisis y diseño de productos....
- **Medicina:** Análisis de células portadoras de cáncer mamario, Diseño de prótesis, Diseño de prediagnósticos médicos basados en síntomas del paciente...
- **Recursos Humanos:** Predicción de qué empleados serán más rentables el año que viene.
- **Robótica:** Control dinámico de trayectoria, Sistemas ópticos...
- **Seguridad:** Reconocimiento de huellas digitales, Criptografía...
- **Telecomunicaciones:** Compresión de datos e imágenes, Automatización de servicios de información, Decidir cuál es la mejor hora para llamar a un cliente...
- **Transporte:** Sistemas de rutas y seguimiento de flotas, Diagnóstico de frenos en camiones...
- **Voz:** Reconocimiento de voz, Transformación de texto escrito a voz...

Un ejemplo concreto de aplicación de esta técnica puede ser el siguiente:

Una empresa de telefonía quiere saber qué clientes están en “peligro” de darse de baja de sus servicios para hacer acciones comerciales que eviten que se vayan a la competencia.

¿Cómo puede hacerlo? La empresa tiene muchos datos de los clientes, pero seguramente los usa solo para facturar y para hacer estadísticas.

¿Qué más puede hacer con esos datos? Se pueden usar para predecir cuándo un cliente se va a dar de baja y gestionar la mejor acción que lo evite.

Los datos históricos del conjunto de los clientes, debidamente organizados y tratados en bloque, generan una base de datos que se puede explotar para predecir futuros comportamientos (favorecer aquellos que mejoran los objetivos de negocio y evitar aquellos que son perjudiciales).

Esa cantidad inmensa de datos son imposibles de analizar por una persona para sacar conclusiones y menos todavía para hacer predicciones. Los algoritmos en cambio sí pueden detectar patrones de comportamiento contando con las variables que le proporcionamos y descubrir cuáles son las que han llevado, en este caso, a darse de baja como cliente.

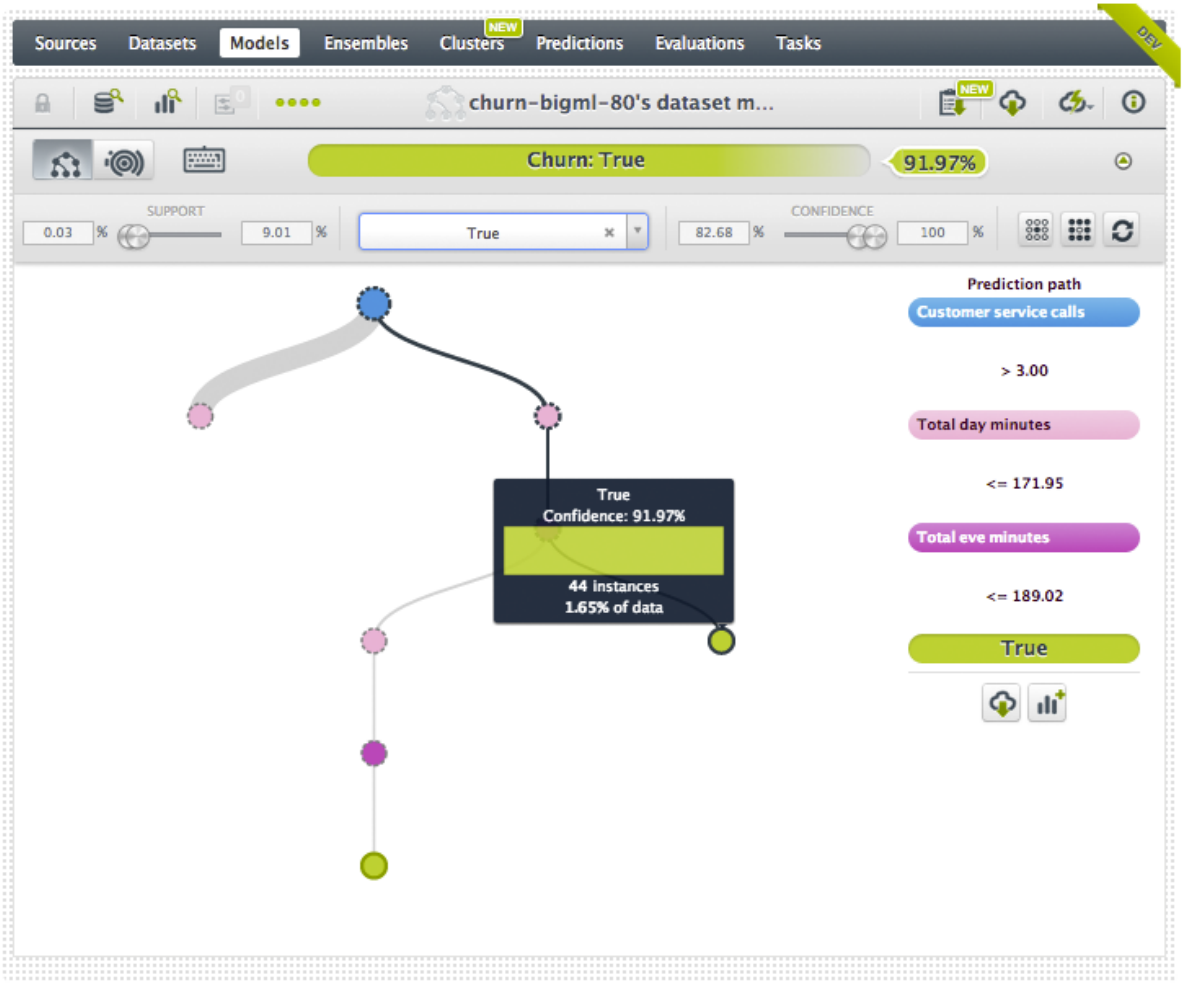

Figura 1: Ejemplo de una predicción simplificada basada en datos de una compañía de telefonía ficticia, pero usando una herramienta de Machine Learning real

La visualización en árbol (en esta imagen está simplificado) permite ver los patrones que han seguido ciertos clientes que se han dado de baja.

En este caso está resaltada una de las ramas centrales, que indican un patrón en el que el cliente:

- Tiene más de 3 llamadas al servicio de atención al cliente.
- Llama menos de $171,95$ minutos al día.
- Las llamadas en horario nocturno son inferiores a $189,02$ minutos.

Por tanto, la predicción sería: *Si los clientes que tienen estas características ya se han dado de baja de la compañía, es previsible que los que todavía son clientes y tienen este mismo comportamiento*

estén en riesgo de irse. Según este modelo predictivo, es bastante probable que esto suceda (se dice que la predicción tiene una confianza, en este caso, del $91,97\,\%$*).*

Si el departamento de marketing tuviera esta información, podría proponerles a dichos clientes un cambio de plan de tarificación o podría revisar por qué han llamado al servicio de atención al cliente para intentar mantenerlos.

2. Tipos de aprendizaje

Los diferentes formas de aprender se pueden clasificar en función de la salida que produzca y de su forma de tratar con los ejemplos. Los principales tipos son:

- **Aprendizaje supervisado:** Aprende, a partir de un conjunto de instancias pre-etiquetadas un método para predecir(Ejemplo, clasificación:la clase a que pertenece una nueva instancia)
- **Aprendizaje no supervisado**: No hay conocimiento a priori sobre el problema, no hay instancias etiquetadas, no hay supervisión sobre el procedimiento.(Ejemplo, clustering:Encuentra un agrupamiento de instancias "natural" dado un conjunto de instancias no etiquetadas)
- **Aprendizaje semisupervisado**: Este tipo de algoritmos combinan los dos algoritmos anteriores, teniendo en cuenta ejemplos clasificados y no clasificados.
- **Aprendizaje por refuerzo**: Los objetivos reaccionan a las acciones del agente mediante una función(reward) que informa al agente de la bondad de la acción. Por lo tanto, el sistema aprende a base de ensayo-error, reforzando las acciones que reciben una respuesta positiva.
- **Aprendizaje multi-tarea**:Métodos que usan conocimiento previamente aprendido para resolver problemas parecidos. Se basa en la teoría de que nuestro cerebro tiene que aprender no lo hace de forma aislada, sino que utiliza todo el conocimiento previamente aprendido para ayudarse en este nuevo aprendizaje.

3. Antecedentes estadísticos

Para que los sistemas puedan aprender por ellos mismos, se utilizan una serie de técnicas y algoritmos capaces de crear modelos predictivos, patrones de comportamiento, etc. Este proceso está basado en la teoría estadística. A continuación vamos a ver 3 de los métodos más importantes:

- **Teorema de Bayes:**. En 1812 **Pierre-Simon Laplace** publica *Théorie Analytique des Probabilités*, en la que expande la obra de Bayes y define lo que ahora se conoce como Teorema de Bayes.

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)},$$

Figura 2: Teorema de Bayes

  La teoría bayesiana es muy usada en machine learning. Su idea básica es que todo lo que has aprendido es incierto, así que tienes que calcular la probabilidad de que sea incorrecto, e ir actualizándola según las pruebas. De acuerdo a estos principios se crearon los primeros filtros de correo spam.

- **Método de los mínimos cuadrados**: Adrien-Marie Legendre describió en **1805** el *"méthode des moindres carrés"* conocida como el método de los mínimos cuadrados.

  Su aplicación más importante es en ajuste de datos. El mejor ajuste en el sentido de los mínimos cuadrados minimiza la suma de los residuos cuadrados, siendo un residuo la diferencia entre un valor observado y el valor ajustado proporcionado por un modelo.
- **Cadenas de Markov:** En **1948 Claude Shannon** publicó un artículo, *The Mathematical Theory of Communication*, en el que creó el campo de la teoría de la información, introduciendo el concepto de entropía a través del modelado de Markov del inglés. Estos modelos idealizados pueden capturar muchas de las regularidades estadísticas de los sistemas. También permiten una estimación efectiva del estado y reconocimiento de patrones.

4. Máquina de aprendizaje de Turing

Alan Turing (1912-1954) fue un matemático, criptógrafo, filósofo y un teórico de la computación que, además de trabajar en el equipo que descifró los códigos Enigma de Alemania, fue un pionero en el campo de la inteligencia artificial publicando uno de los primeros trabajos sobre esta materia.

En **1950**, Alan Turing publicó uno de sus artículos más importantes para la revista *Mind* (y que está considerado como una de las piedras angulares de la inteligencia artificial), *Computing Machinery and Intelligence*, donde propuso un experimento: **el Test de Turing**.

El Test de Turing se basa en una hipótesis positivista que parte del principio de que "si una máquina se comporta en todos los aspectos como inteligente, entonces, dicha máquina debe ser inteligente".

El artículo de Turing comenzaba con una frase que era toda una declaración de intenciones de lo que evaluaría el test:

> "Propongo considerar la siguiente cuestión: ¿Pueden pensar las máquinas?"

Por lo que el resultado que se obtiene de este test es intentar medir si una máquina puede ser inteligente con un método que, aún hoy, sigue estando vigente.

La incógnita sobre la capacidad de las máquinas de pensar tiene una larga historia, esta se divide entre las perspectivas materialista y dualista de la mente.

Veamos algunos antecedentes al Test de Turing:

- **René Descartes** tuvo ideas similares a la prueba de Turing en su texto *Discurso del Método* de **1637** donde escribió: "*Cuántos autómatas diferentes o máquinas móviles se pueden construir por la industria del hombre [. . .] Podemos comprender con facilidad el hecho de que una máquina sea diseñada para pronunciar palabras, incluso responder ante acciones tangibles que produzcan un cambio en sus órganos; por ejemplo, si se le tocara de una manera particular no preguntaría qué le queremos decir; si se le toca en otra parte puede decir que se le está lastimando entre otras cosas. Pero nunca sucede que pueda ordenar su habla en distintas maneras para responder apropiadamente ante todo lo que se diga en su presencia, de la misma manera que la clase más baja de humanos puede hacerlo.*"

  Descartes reconoce que los autómatas son capaces de reaccionar ante interacciones humanas, pero argumenta que tal autómata carece de la capacidad de responder adecuadamente ante lo que se diga en su presencia, de la misma manera en la que un humano podría.

Por lo tanto, Descartes abre las puertas para la prueba de Turing al identificar la insuficiencia de una respuesta lingüística apropiada, lo cual separa al humano del autómata. Descartes no llega a considerar que una respuesta lingüística apropiada puede ser producida por un autómata del futuro y, por lo tanto, no propone el test de Turing como tal, aunque ya razonó los criterios y el marco conceptual.

- **Denis Diderot** también planteó, en su *Pensées philosophiques*, un criterio de la prueba de Turing: "*Si se encuentra un loro que puede responder a todo, se le consideraría un ser inteligente sin duda alguna.*"

  Aunque esto no signifique del todo que él esté de acuerdo con el posterior Test de Turing, sí muestra que ya era un argumento común usado por los materialistas de la época.

- En **1936**, el filósofo **Alfred Ayer** consideraba la pregunta filosófica típica sobre otras mentes: "*¿Cómo sabemos que otras personas experimentan el mismo nivel de conciencia que nosotros?.*"

  En su libro *Lenguaje, Verdad y Lógica*, Ayer propuso un método para distinguir entre un hombre consciente y una máquina inconsciente: "*El único argumento que tengo para asegurar que, lo que parece ser consciente, no es un ser consciente sino un muñeco o una máquina, es el hecho de que falle en las pruebas empíricas por medio de las cuales se determina la presencia o ausencia de la conciencia*". O, lo que es lo mismo, "*algo no está consciente si reprueba una prueba de consciencia*".

  Esta es una propuesta muy similar a la prueba de Turing, aunque esta se enfoca en la conciencia en vez de en la inteligencia. Además se desconoce si Turing estaba familiarizado con el clásico filosófico de Ayer.

¿En qué consistía el Test de Turing? El Test de Turing se basaba en el Juego de la Imitación, una prueba en la que se ubicaban en una habitación un **hombre** y a una **mujer** frente a terminales que tuviesen algún sistema de comunicación (teletipos en los años 50).

En otra habitación, estando aislado, se encontraba un sujeto bajo estudio que actuaba como **interrogador**. Este último debía averiguar quién era el hombre y quién era la mujer, simplemente, con preguntas en lenguaje natural (escribiendo en un lenguaje que todos entendían).

El objetivo del interrogador era descubrir quién era la mujer y quién era el hombre, mientras que el de los otros dos era convencer al interrogador de que son la mujer.

Turing proponía realizar un cambio en este juego: coger a uno de los dos sujetos y sustituirlo por una **máquina**. Así, Turing cambiaba el objetivo de reconocer el sexo por el de reconocer la máquina.

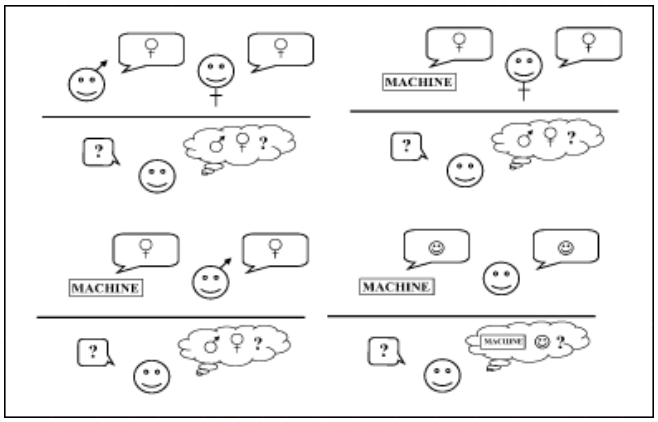

Figura 3: Test de Turing

La finalidad de estos cambios era hacer el juego lo más justo posible.

1. Lo primero, era que no tenía que consistir en un concurso de engaños, por lo que uno de los implicados no tendría por qué aparentar ser otra cosa.
2. Otro detalle es que a Turing poco le importaba si el ordenador empleaba trucos preestablecidos para eludir o manipular las respuestas (por ejemplo, equivocándose en preguntas aritméticas o tardando más tiempo del necesario en responderlas). Suponía que el interrogador también las empleaba para reconocerle, así que lo importante era lo que resultaba del juego, no los métodos que se empleaban para jugar ni los mecanismos internos de razonamiento, que, entre otras cosas, también eran desconocidos en el ser humano.

Dado que el interrogador era humano, la máquina debería ser capaz de responder, inteligentemente, a cualquiera de las cuestiones que se le planteasen. Si la máquina era lo suficientemente hábil, el interrogador no podría distinguir quién era la máquina y quién el ser humano.

Pero, "¿puede una máquina responder con un lenguaje natural a cualquier pregunta planteada por el interrogador?¿Puede pensar una máquina?"

Esas son las grandes preguntas que Turing plantea en este experimento. Lógicamente, salvo que una máquina emita sonidos que imiten la voz humana, la mejor manera de realizar la prueba es mediante lenguaje escrito y evaluar, únicamente, la inteligencia del oponente y la "humanidad"de las respuestas obtenidas a las preguntas planteadas.

Para que una máquina pudiese pasar el Test de Turing, ésta tendría que ser capaz de realizar las tareas para las que fue programada y, además, aprender por sí misma a realizar otras funciones.

Se dice que una máquina podría pasar el Test de Turing cuando el interrogador no lograra reconocerlo en un número significativo de ocasiones.

Hoy en día, estamos acostumbrados a ver algunos mecanismos que se apoyan en el Test de Turing, por ejemplo, los *Captchas* (*Completely Automated Public Turing Test to tell Computers and Humans Apart*) que aunque no son un test de Turing puro, se utilizan en muchas páginas web para evitar que procesos automatizados puedan registrar usuarios falsos o escribir mensajes de spam.

5. La red neuronal en el tiempo

Las **redes neuronales** son un conjunto de neuronas artificiales interconectadas, son un paradigma de aprendizaje y procesamiento automático inspirado en el sistema nervioso, que es objeto de estudio en el campo de la inteligencia artificial, pero **¿Cuándo surgió esta idea?** y **¿Qué ideas han dado lugar a las redes neuronales tal y como son en la actualidad?**.

En el año **1943**, el neurofisiólogo **Warren McCulloch** y el matemático **Walter Pitts** escribieron un documento en el que se describe como podrían funcionar las neuronas construyendo un modelo simple de red neuronal usando circuitos electrónicos. Para esto se basan en dos puntos de vista distintos, los procesos biológicos del cerebro y la aplicación de estas redes en la inteligencia artificial.

En el año **1949 Donald Hebb** en su obra *''The Organization of Behavior"* señala el hecho de que cada vez que se usa un camino de una red neuronal estas se fortalecen explicando que este concepto es fundamental para el aprendizaje de un ser humano y esto es conocido como regla

de Hebb o teoría de la asamblea celular. Este es un mecanismo de plasticidad sináptica (que se encarga de la modulación de percepción se estímulos de una neurona) en el que el valor de una conexión sináptica se incrementa si las neuronas de ambos lados de dicha sinapsis se activan repetidas veces de forma simultanea, la teoría suele resumirla la frase: "Las células que disparan juntas, permanecerán conectadas", aunque esta no debe tomarse literalmente. Esta forma de aprendizaje se denomina **aprendizaje de Hebb** o **Hebb Learning**.

Hasta ahora los ordenadores no tenían la capacidad de simular una red neuronal pero a partir de los años 50, estos obtienen mayor capacidad y es Nathanial Rochester de los laboratorios de investigación de IBM el que intento por primera vez simular pero fue un primer intento fallido.

Marvin Lee Minsky y **Dean Edmonds** construyeron la **primera máquina que ejecutaría una red neurona**l haciendo uso del aprendizaje de Hebb anterior en el año 1951, aunque en entre los años 1954-1956 también hay otros autores que simulan una red neuronal sobre máquinas llamadas por entonces calculadoras.

Por el año **1958 Frank Rosenblatt** desarrollar el **perceptrón** un algoritmo de reconocimiento de patrones que será explicado con mayor profundidad en una sección posterior y describió que no era posible procesar un circuito xor por una red neuronal hasta la creación del algoritmo de backpropagation.

Durante los próximos años se desarrolla el **primer modelo de red neuronal aplicado a un problema real** apodado **Madaline** o Multiple adaptative lineal elements, el problema fue eliminar el eco producido en las lineas telefónicas y esto se trataba de hacer a través de un filtro adaptativo. En los años 60 después de que la arquitectura tradicional de Von Neumann entrase en escena, las redes neuronales se dejaron de lado. En el año 1969 Marvin Minsky y Seymour Papert publicaron un documento en el que se se dice que un computador no tenia la suficiente capacidad de procesamiento como para ejecutar una red neuronal extensa por mucho tiempo, a parte de que el perceptrón no era capaz de procesar los circuitos x-or.

A partir de los 80 el interés en el área se restablece y aparecen nuevos trabajos y documentos que pretendían hacer más funcionales las máquinas actuales haciendo **bidireccionales** las **conexiones** entre las neuronas que previamente fueron unidireccionales, además surgen nuevas ideas y conceptos que se apoyan en la base de las redes neuronales artificiales como las redes neuronales multicapa que extiende el problema introducido por Widrow y Hoff en el año 1952.

Actualmente el avance de este área se esta realizando a nivel de **hardware** más que del software, esto es debido a que se necesita mucha capacidad para procesar y la eficiencia de una red neuronal depende mucho del hardware usado, normalmente se construyen máquinas dedicadas a una tarea con un hardware específico para poder ejecutar la red neuronal necesaria como algunas que veremos en secciones posteriores.

5.1. BackPropagation

La **propagación hacia atrás de errores** o **"backward propagation of errors"**, es un conocido método de entrenamiento de redes neuronales artificiales, se trata de un algoritmo de aprendizaje que usa un método de optimización que normalmente suele ser un descenso local o descenso de gradiente. El algoritmo fue descrito en el año **1986** por **David Rumelhart**, **Geoff Hinton** y **Ronald J. Williams** aunque **Seppo Linnainmaa** publicó en el año **1970** un método denominado **Diferenciación automática** que corresponde al actual concepto de propagación hacia atrás, el algoritmo se basaba en la repetición de dos procesos la propagación y la actualización de pesos de una neurona, cuando una instancia se usa como entrada de una neurona, se propaga hacia delante de capa en capa hasta que llega a la salida o última capa, y es aquí donde se hace uso de

una función denominada función de pérdida para calcular el error producido por la entrada que se propaga hacia la primera capa de la neurona para recalcular la entrada haciendo uso de ese error para optimizar los resultados con el método de descenso de gradiente.

6. El problema de la Clasificación

En el aprendizaje automático y la estadística en **problema de las clasificación** consiste en identificar un conjunto de subconjuntos de miembros de una población basándose en un conjunto de entrenamiento que contiene instancias que ya son conocidas. La clasificación es un ejemplo de reconocimiento de patrones en una población.

En primer lugar es preciso explicar sobre que se parte a la hora de aplicar estos algoritmos que aprenden para clasificar. Los **datos** son representación simbólica de atributos o un conjunto de atributos, estos son la principal materia prima para el aprendizaje, sin ellos no sería posible realizar una predicción con mayor o menor seguridad de que la predicción es acertada. Los datos normalmente son divididos en dos conjuntos llamados **conjunto de test o prueba** y **conjunto de entrenamiento**, y lo que se busca con la aplicación de alguno de estos algoritmos es la creación de un modelo de clasificación que sea más general a partir de los datos obtenidos, para ello se hace uso de el algoritmo sobre el conjunto de entrenamiento para generar el modelo y luego se prueba sobre el conjunto de prueba. De aquí surge otra de los grandes problemas de los algoritmos de clasificación el **sobreajuste** o sobreentrenamiento, se produce cuando un modelo se ajusta muy bien al conjunto de datos que usamos para entrenar y que de alguna manera elimina la generalidad que debería de tener el modelo para poder tratar con él.

Figura 4: Ejemplo de sobreajuste

La importancia de estos conjuntos de entrenamiento y prueba es notoria, normalmente el conjunto de entrenamiento es usado para descubir relaciones en los datos que posteriormente nos puedan servir para predecir y el conjunto de prueba es se usa para evaluar el ajuste que dichas relaciones tienen sobre otros datos. Se buscan muchas formas para evitar el sobreajuste, para ello se introducen las técnicas de validación. La validación es un conjunto de técnicas que se aplican normalmente antes de evaluar el modelo obtenido con el aprendizaje sobre e conjunto de entrenamiento y sirve para obtener resultados que ilustren una hipotética relación entre los atributos. Una de las técnicas que más usan esto se llama validación cruzada, que consiste realizar k particiones del mismo tamaño que se usaran para aprender y evaluar de forma independiente en cada caso para finalmente realizar aplicar una media aritmética sobre las aproximaciones obtenidas.

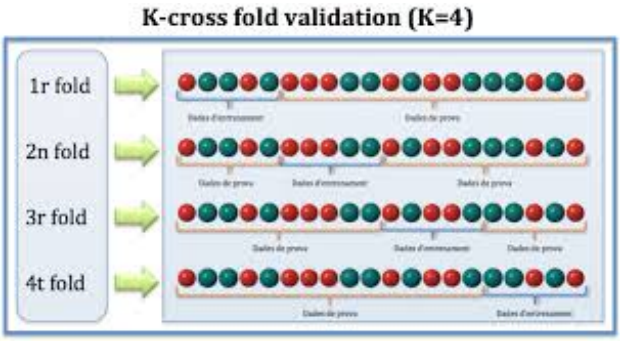

Figura 5: Ejemplo validación cruzada

Con esta breve introducción vamos a repasar algunos de sus algoritmos más importantes han surgido a lo largo del tiempo y han sido un referente para algoritmos posteriores.

6.1. perceptrón

Una de las características más significativas de las redes neuronales es su capacidad para aprender a partir de alguna fuente de información interactuando con su entorno.

La primera red neuronal conocida, fue desarrollada en 1943 por Warren McCulloch y Walter Pitts; esta consistía en una suma de las señales de entrada, multiplicadas por unos valores de pesos escogidos aleatoriamente. La entrada es comparada con un patrón preestablecido para determinar la salida de la red. Si en la comparación, la suma de las entradas multiplicadas por los pesos es mayor o igual que el patrón preestablecido la salida de la red es uno (1), en caso contrario la salida es cero (0).

Al inicio del desarrollo de los sistemas de inteligencia artificial, se encontró gran similitud entre su comportamiento y el de los sistemas biológicos y en principio se creyó que este modelo podía computar cualquier función aritmética o lógica.

La red tipo Perceptrón fue inventada por el psicólogo **Frank Rosenblatt** en el año **1957**. Su intención era ilustrar algunas propiedades fundamentales de los sistemas inteligentes en general.

Rosenblatt creía que la conectividad existente en las redes biológicas tiene un elevado porcentaje de aleatoriedad, por lo que se oponía al análisis de McCulloch y Pitts en el cual se empleaba lógica simbólica.

Rosenblatt opinaba que la herramienta de análisis más apropiada era la teoría de probabilidades, y esto lo llevó a una teoría de separabilidad estadística que utilizaba para caracterizar las propiedades más visibles de estas redes de interconexión ligeramente aleatorias.

El primer modelo de Perceptrón fue desarrollado en un ambiente biológico imitando el funcionamiento del ojo humano. El **fotoperceptrón**, como se le llamó, era un dispositivo que respondía a señales ópticas.

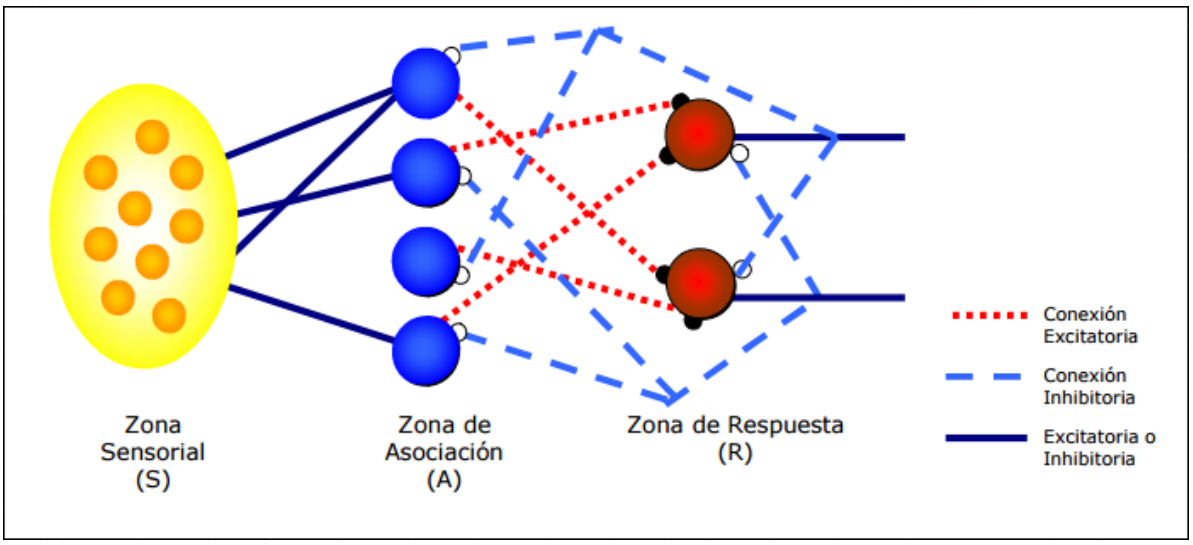

Figura 6: Fotoperceptron

- La luz incide en los puntos sensibles (S) de la estructura de la retina y cada uno de esos puntos responde en forma todo-nada a la luz entrante.
- Los impulsos generados por (S) se transmiten a las unidades de asociación (A), donde cada unidad de (A) está conectada a un conjunto aleatorio de puntos de (S). Estas conexiones tienen los valores posibles $+1$, -1 y 0.
- Cuando aparece un conjunto de estímulos en la retina (S), una unidad de (A) se activa si la suma de sus entradas sobrepasa algún valor umbral. Si la unidad esta activada, produce una salida que se envía a la siguiente capa de unidades.
- De forma similar, las unidades de (A) están conectadas a unidades de respuesta (R) y la conectividad vuelve a ser aleatoria entre ellas.

Se pudo demostrar que el Perceptrón era capaz de clasificar patrones correctamente, en lo que Rosenblatt denominaba un entorno diferenciado, en el cual cada clase estaba formada por patrones similares. El Perceptrón también era capaz de responder de manera congruente frente a patrones aleatorios, pero su precisión iba disminuyendo a medida que aumentaba el número de patrones que intentaba aprender.

En 1969 Marvin Minsky y Seymour Papert publicaron su libro: "Perceptrons: An introduction to Computational Geometry", en el que se presentaba un análisis detallado del Perceptrón, en términos de sus capacidades y limitaciones. La mayor desventaja de este tipo de redes es su incapacidad para solucionar problemas que no sean linealmente separables.

Minsky y Papert se apartaban de la aproximación probabilística de Rosenblatt y volvían a las ideas de cálculo de predicados en el análisis del Perceptrón.

La estructura de un Perceptrón sencillo es similar a la siguiente:

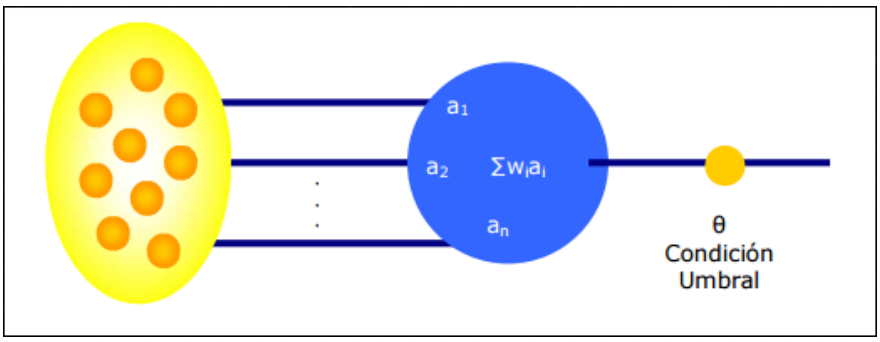

Figura 7: Estructura perceptrón

en la cual se observa la adición de una condición umbral en la salida. Si la entrada completa a esta condición es mayor que el valor umbral, la salida de la red es 1 y en caso contrario es -1.

De manera más específica, el Perceptrón se puede definir de la manera siguiente:

*Un **perceptrón simple** es un dispositivo de computación con umbral θ y N entradas reales $x_1, ..., x_N$ a través de arcos con pesos $w_1, ..., w_N$ y que tiene salida 1 cuando $\sum_i w_i x_i \geq \theta$ y -1 en caso contrario.*

Es decir, supongamos la función de clasificación f en $\mathbb{R}^n$ tal que:

$$f(x_1, x_2, ..., x_N) = \begin{cases} 1 & si \quad w_1x_1 + w_2x_2 + ... + w_Nx_N \geq \theta \\ \\ -1 & si \quad w_1x_1 + w_2x_2 + ... + w_Nx_N < \theta \end{cases}$$

Dicha función realiza una partición en el espacio $\mathbb{R}^n$ de patrones de entrada: por una parte estarían los patrones con salida $+1$ y por otra parte los patrones con salida -1. Por lo tanto, diremos que la función f clasifica a los patrones de entrada en dos clases.

En este caso, se expresa la función f mediante la función signo, es decir:

$$f(x_1, x_2, ..., x_N) = sign(x_1, x_2, ..., x_N)$$

donde la función signo es:

$$sign(x_1, x_2, ..., x_N) = \begin{cases} 1 & si \quad x \geq 0 \\ \\ -1 & si \quad x < 0 \end{cases}$$

El algoritmo de aprendizaje del perceptrón consiste en lo siguiente:

1. Suponemos que el conjunto de datos es linealmente separable y que se puede encontrar una solución correcta del problema.
2. El objetivo será buscar el vector W (tal que su hiperplano asociado separe los puntos) que alcanza la solución correcta para todos los puntos.
3. Suponemos unas salidas esperadas prefijadas Y_i para cada punto a clasificar (con valores $+1$ y -1).
4. **Paso 1:** Elegir un punto X_i y seleccionar el vector W_i de los pesos en ese momento (al principio dicho vector se podrá inicializar a 0). Es decir, tenemos las parejas siguientes: $(X_1, Y_1), ..., (X_N, Y_N)$ y el algoritmo cogerá un punto que está mal clasificado en ese momento.

5. **Paso 2 y sucesivos:** Para el punto X_i seleccionado actualmente, calcular $sign(W^T \times X_i)$.
   - Si $sign(W^T \times X_i) \neq Y_i \Rightarrow$ aplicar la regla de actualización $W_{i+1} = W_i + Y_i X_i$.
   - En caso contrario, volver al paso 2 y continuar el algoritmo con el siguiente punto.
6. La regla de adaptación de pesos del Perceptrón realiza un movimiento en la dirección correcta para clasificar bien el punto actual X_i.
7. El algoritmo finaliza cuando no haya más puntos mal clasificados.

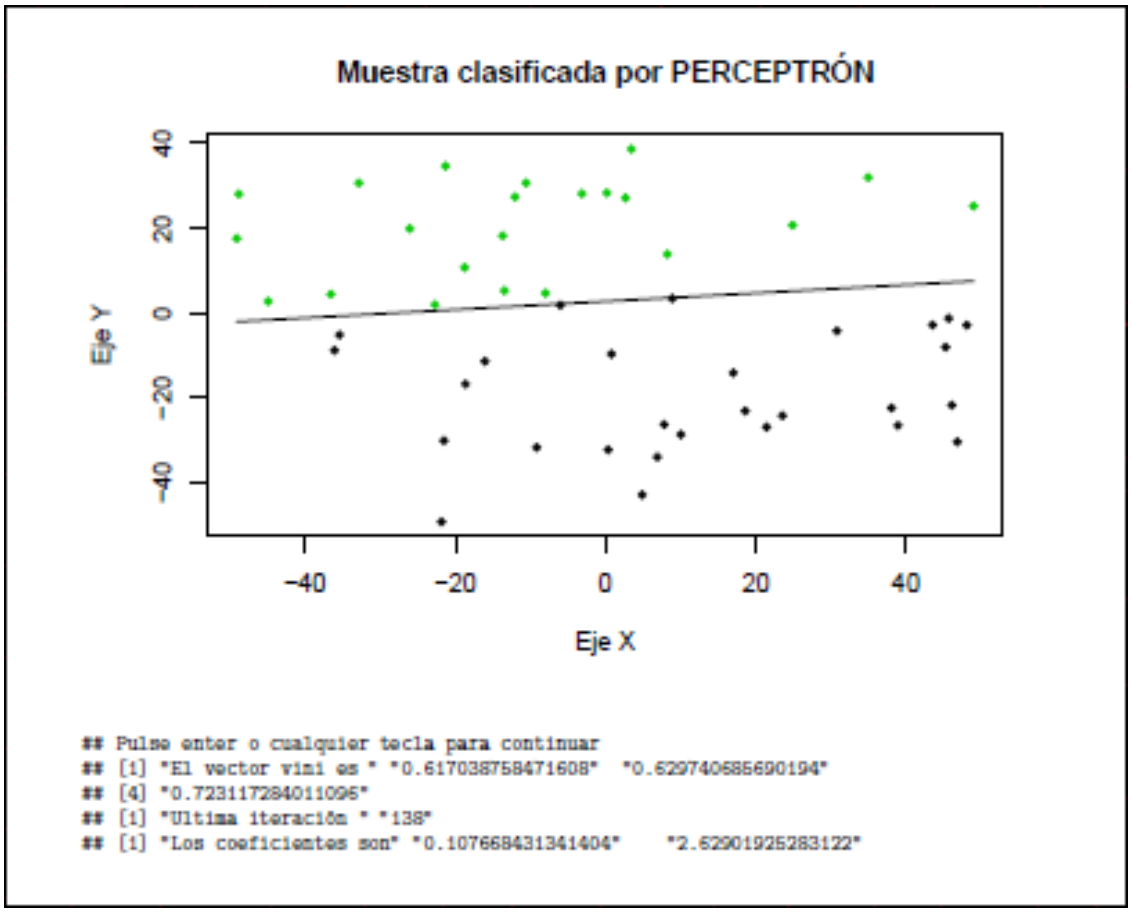

Figura 8: Ejemplo Perceptrón

6.2. Nearest Neighbor

El algoritmo de clasificación **K-Nearest Neighbour (k-NN)** o **K-Vecino más Cercano** es uno de los métodos de clasificación más fundamentales y sencillos, y debería ser una de las primeras opciones para un estudio de clasificación cuando hay poco o ningún conocimiento previo sobre la distribución de los datos.

Este algoritmo de clasificación se desarrolló debido a la necesidad de realizar análisis discriminantes (que es una técnica estadística capaz de decirnos qué variables permiten diferenciar distintos grupos y cuántas de estas variables son necesarias para alcanzar la mejor clasificación posible) cuando las estimaciones paramétricas fiables de las densidades de probabilidad son desconocidas o difíciles de determinar.

En 1951, **Fix y Hodge** introdujeron este método no paramétrico para la clasificación de patrones, conocido desde aquel entonces como la regla del k-vecino más cercano.

Pero no fue hasta el año **1967** cuando se tuvo constancia de este método y se resolvieron algunas de las propiedades formales de dicho algoritmo. Por ejemplo, se demostró que para $k = 1$ y $n \to \infty$ el error de clasificación de este algoritmo está delimitado por dos veces la tasa del error de Bayes *(Cover & Hart, 1967)*.

Por ello, y de manera resumida, se dice que en 1967, Fix & Hodge desarrollan el algoritmo **Nearest Neighbour** o **Vecino más Cercano**.

Este hito está considerado como el nacimiento del *campo del reconocimiento de patrones* o *pattern recognition* en computadores.

Una vez que se establecieron en 1967 tales propiedades formales, se siguió una larga línea de investigación incluyendo nuevos enfoques de rechazo *(Hellman, 1970)*, refinamientos con respecto a la tasa de error de Bayes *(Fukunaga y Hostetler, 1975)*, enfoques basados en la distancia *(Dudani, 1976; Bailey y Jain, 1978)*, soft computing *(Bermejo y Cabestany, 2000)* y métodos difusos *(Jozwik, 1983; Keller et al., 1985)*.

Los primeros programas capaces de reconocer patrones fueron diseñados en base a este algoritmo.

Dicho algoritmo se basa en:

- El aprendizaje basado en casos: extracción de información de un conjunto de datos conocidos para clasificar nuevos o agrupar existentes.
- El razonamiento basado en casos: imitación del sentido común humano a través de experiencias pasadas y solución de problemas presentes con analogías de problemas pasados.
- Métodos basados en vecindad y de medición de distancia.

Las aplicaciones más importantes de este algoritmo son:

1. Escalamiento de gráficos evitando la menor pérdida de píxeles.
2. Sistemas de GPS actuales.
3. Reconocimiento de rostros.

¿Cómo funciona k-NN? Parte de la idea de que una nueva muestra será clasificada en la clase a la cual pertenezcan la mayor cantidad de vecinos más cercanos (del conjunto de entrenamiento más cercano) a esta.

Es decir, cuando se le da un nuevo objeto al programa, éste lo compara con los datos del conjunto de entrenamiento y clasifica el objeto con el vecino más cercano o con el objeto más similar en la memoria.

Veamos un ejemplo del funcionamiento de esta regla de clasificación:

Representamos 12 muestras pertenecientes a dos clases distintas: la Clase 1 está formada por 6 cuadrados de color azul y la Clase 2 está formada por 6 círculos de color rojo.

Seleccionamos tres vecinos, es decir, $k = 3$.

De los 3 vecinos más cercanos a la muestra x (representada en la figura por una cruz) uno de ellos pertenece a la Clase 1 y los otros dos a la Clase 2.

Por tanto, la regla $3 - NN$ asignará la muestra x a la Clase 2.

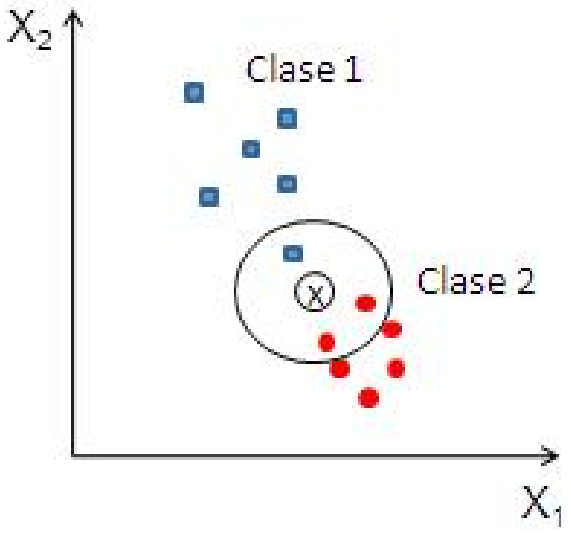

Figura 9: Ejemplo kNN

Es importante señalar que si se hubiese utilizado como regla de clasificación la $1-NN$, la muestra x sería asignada a la Clase 1, pues el vecino más cercano de la muestra x pertenece a la Clase 1.

6.3. Otros algoritmos que clasifican

6.3.1. Árboles de decisión

Un **árbol de decisión** es un grafo dirigido donde los nodos son reglas sobre los atributos del conjunto de instancias y las hojas del árbol son el atributo clase o el atributo sobre el que se va a intentar clasificar. A lo largo de la historia se han ido surgiendo algoritmos que siguen esta estructura con el objetivo de dar una solución al problema de la clasificación, algunos de los algoritmos más relevantes que aprenden basándose en árboles de decisión son:

- **Algoritmo ID3:** Es un algoritmo que indice árboles de decisión a partir de un conjunto dado de instancias con atributos determinados, fue inventado por **John Ross Quinlan** en los años 60 siguiendo el sencillo principio de la **navaja de Occam** para intentar construir un árbol de decisión lo más pequeño posible.

  El algoritmo trata de calcular la entropía o valor promedio que cada uno de los atributos de un conjunto de datos y haciendo uso de esta información dividir en subconjuntos más pequeños basándose en las instancias con la entropía mínima o máxima dependiendo del criterio, para después construir un árbol de decisión que contenga el atributo con dicha entropía, a partir de aquí utiliza la recursividad para repetir el proceso con los demás atributos.
- **Algoritmo C4.5:** Es el precursor del algoritmo ID3 desarrollado también por Quinlan que empezó a tomar relevancia después de que obtuviera el primer puesto en el documento publicado por Springer LNCS llamado *"Top 10 Algorithms in Data Mining"*. El algoritmo construye árboles de decisión de la misma manera que lo hace el algoritmo ID3 usando el concepto de entropía solo que ahora el criterio de selección de que atributo usar para dividir el conjunto en subconjuntos de forma efectiva cambia usándose una ganancia de información o entropía relativa, esta medida es la diferencia entre dos distribuciones de probabilidad P y Q donde P representa la distribución de valores correctamente clasificados y Q es una aproximación de P.

- **Algoritmo Random Forest:** Recientemente en el año **1995**, **Tinkam Ho** un analista de datos publica un documento en el que se describía este algoritmo, este algoritmo trata de construir un conjunto de árboles de decisión. Haciendo uso de un conjunto de entrenamiento hacemos que se construya un conjunto de árboles de decisión en el que dada una entrada, que puede ser una nueva instancia, esta se clasifica por todos los árboles del conjunto y se predice a que clase esta instancia pertenece en función de la mayoría de los resultados de la clasificación anterior, es decir si hay 3 árboles de decisión y 2 dicen que pertenece a el conjunto de instancias mal clasificadas entonces ese será su valor de clase.

6.3.2. Support Vector Machine

El algoritmo **support vector machine** o SVM son un conjunto de **técnicas de aprendizaje supervisado** desarrollados por publicado en un documento en el año **1995** en el que trabajaron **Corinna Cortes** y **Vladimir Vapnik**, es un método que dado un conjunto de entrenamiento construye un modelo que servirá para realizar predicciones sobre un conjunto de instancias o una muestra del mismo tipo. Es un modelo que representa las instancias en el espacio separando las clases lo máximo haciendo uso de un hiperplano o un conjunto de hiperplanos dependiendo de la dimensionalidad de los atributos del conjunto de datos la idea es que estos separen de forma óptima las clases. Este algoritmo puede ser usado tanto en problemas de regresión como en problemas de clasificación.

Este algoritmo pertenece a la familia de clasificadores lineales, es decir, algoritmos que tratan de clasificar una instancia basándose en valores de una combinación lineal de sus atributos.

7. Máquinas y juegos

En esta sección mencionaremos los años y nombraremos las personalidades que han sido capaces de crear un programa o programas basándose en algoritmos de aprendizaje para que las máquinas sean capaces de aplicar las reglas del juego de una forma lógica y siempre con la intención de ganar.

7.1. Máquinas jugando a las Damas

En el año **1952 Arthur Samuel** que acababa de unirse a un laboratorio de investigación de IBM escribe uno de los primeros programa de aprendizaje automático capaz de jugar a las damas. Se le denomina como un pionero en el campo de juegos de ordenador e inteligencia artificial. El programa de Samuel parece que fue el **"primer programa"** en el mundo capaz de aprender.

Figura 10: Arthur Samuel jugando a las damas

El programa hacía uso de un árbol de juegos basándose en la estrategia minimax principalmente y usando la poda $\alpha\beta$ algoritmos que tratan de anticiparse a lo que hará el contrincante generando un árbol de acciones y podando quedándose con los caminos más probables y eligiendo aquel que obtenga un mayor porcentaje de victoria, pero luego diseñó varios mecanismos que le permitirían mejorar su programa, a estos mecanismos los denomino rote learning o aprendizaje por rutina que es básicamente una técnica que recuerda cada una de las posiciones que ya han sido visitadas basándose en la repetición y una revaloración del estado de juego.

7.2. Máquinas jugando al Backgammon

En el año **1992**, **Gerry Tesauro** desarrolla un programa de aprendizaje capaz de jugar al backgammon, basándose en conocimiento obtenido del juego. Esta aplicación del aprendizaje automático fue capaz de jugar casi tan bien como el campeón mundial en el juego. El aprendizaje estaba basado en un algoritmo abreviado TD o temporal difference learning, es un algoritmo que trata de realizar predicciones basado en una combinación del método de Monte Carlo y la programación dinámica.

7.3. Máquinas jugando al Ajedrez

Figura 11: Máquina Deep Blue de IBM

La supercomputadora **Deep Blue** instalada el año **1996** y desarrollada por IBM era capaz de jugar al ajedrez, lo novedoso de esto es que fue la primera máquina que consiguió vencer una partida al vigente campeón del mundo **Gary Kasparov** en un encuentro de 6 partidas realizado el mismo año. Sin embargo deep blue no fue capaz de ganar el encuentro debido a que Kasparov gano 4 de los 6 y empató los otros 2. En el año **1997** esta máquina tuvo una considerable mejora pasándose

a llamar Deeper Blue que volvió a jugar contra aún vigente campeón del mundo Kasparov de nuevo en un encuentro a 6 partidas donde esta vez ganó **Deeper blue** con 3 victorias 2 derrotas y 1 empate.

7.4. Máquinas jugando al go

En el año **2015 Google DeepMind** de Londres desarrolla un programa de ordenador llamado AlphaGo capaz de jugar al juego Go. Que resulta ser el primer programa que juega a ese juego en derrotar una persona profesional en un tablero de 19x19. El juego Go es considerado mucho más difícil de tratar computacionalmente que otros juegos como el ajedrez debido a que se tiene una gran cantidad de opciones y por tanto un gran árbol de decisión generado. Esto hace que los típicos algoritmos como la poda $\alpha\beta$ y otros métodos de búsqueda sean difíciles de usar, la capacidad de aprendizaje del programa se basa en una red neuronal que usa deep learning. El deep learning es una rama del machine learning que toma un conjunto de algoritmos que intentan obtener un modelo de abstracciones sobre los datos.

8. Stanford cart y otros experimentos

El **Stanford Cart** nació de una investigación para controlar un Moon Rover desde la Tierra. Fue construido en **1960** por un estudiante, **James L. Adams**, quien había estado trabajando en un proyecto de la NASA que pretendía que alguien desde la Tierra pudiese conducir alrededor de la Luna usando una cámara de vídeo en el vehículo. Adams se centró en estudiar la controlabilidad del vehículo con varias configuraciones de la velocidad de comunicación. Demostró que con un retardo de comunicación correspondiente al viaje de ida y vuelta a la Luna (aproximadamente 2.5 segundos), el vehículo no podía ser controlado de manera fiable si viajaba a más de 0.3 km/h.

En **1962** otro estudiante, **Paul W. Braisted**, ideó un esquema para mejorar la controlabilidad del vehículo agregando una computadora que funcionaba como un predictor que tomaba en cuenta comandos de dirección anteriores y dibujaba un punto en la pantalla de televisión que indicaba la localización cuando una orden se llevara a cabo. Con esta mejora el vehículo podría ser controlado a 8 km/h.

En **1969**, **Lester Earnest** intentó construir un vehículo de carretera auto-dirigido basado en el aporte visual. Earnest tenía en mente tratar de conducir por la carretera circular que rodea la instalación de SAIL usando la línea central del camino y otras referencias visuales. Sin embargo sabía que el carro no podría llevar una computadora bastante grande para hacer eso, así que planeó utilizar un transmisor en el carro para enviar imágenes de la cámara de la TV a la computadora y otro enlace de radio para enviar órdenes de guía de la computadora al carro. Las operaciones experimentales comenzaron con un operador humano que controlaba el carro a través de la computadora usando imágenes de televisión. Utilizando el procesador KL10 entonces disponible, que funcionaba a unos 2,5 MIPS, **Hans Moravec** fue capaz de usar la visión multi-ocular para navegar lentamente alrededor de los obstáculos en un ambiente interior. En **1979**, el carro cruzó con éxito una sala llena de sillas sin intervención humana en unas cinco horas.

Figura 12: Stanford Cart

Hoy en un día hay varios programas activos de investigación para fabricar coches completamente autónomos.En agosto de 2016 la empresa estadounidense nuTonomy, filial del MIT, lanzó el primer taxi autónomo del mundo en Singapur.

9. Neocognitrón

El neocognitrón es una red neuronal artificial jerárquica y multicapa propuesta por **Kunihiko Fukushima** en los años ochenta.Se ha usado para el reconocimiento de caracteres escritos a mano y otras tareas de reconocimiento de patrones, y sirvió de inspiración para redes neuronales convencionales.Su diseño se basó en el trabajo anterior realizado por Hubel y Weisel que aclaraba parte de la arquitectura funcional de la corteza visual.

El neocognitrón consiste en múltiple tipos de células, las más importantes son las llamadas células S y células C. Las células S funcionan como células que extraen características. Sus conexiones de entrada son variables y se modifican a través del aprendizaje. En términos generales, las características locales, tales como bordes o líneas en orientaciones particulares, se extraen en etapas inferiores. Más características globales, como partes de los patrones de aprendizaje, se extraen en etapas superiores. Las células C, se insertan en la red para permitir errores de posición en las características del estímulo. Cada célula C recibe conexiones de entrada excitatoria de un grupo de células S que extraen la misma característica, pero desde posiciones ligeramente diferentes. La célula C responde si al menos una de estas células S produce una salida.

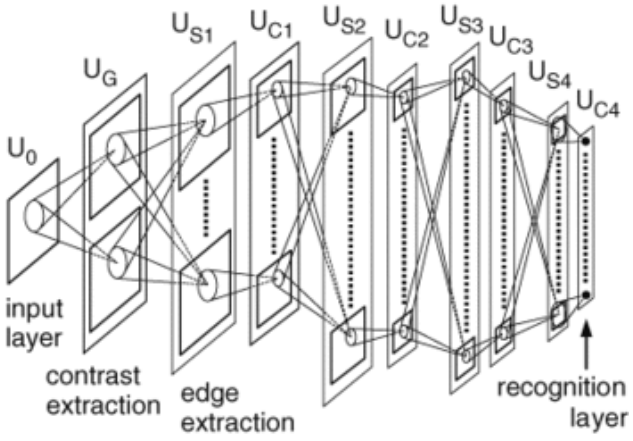

Figura 13: Estructura del neocognitrón.

10. Otras técnicas de aprendizaje

10.1. Colonias de hormigas

El algoritmo de la colonia de hormigas es una técnica probabilística para solucionar problemas computacionales que pueden reducirse a buscar los mejores caminos o rutas en grafos.

Inicialmente fue propuesto por **Marco Dorigo** en **1992**, en su tesis de doctorado.

Dicho algoritmo surgió como método para buscar el camino óptimo en un grafo, basado en el comportamiento de las hormigas cuando estas están buscando un camino entre la colonia y una fuente de alimentos.

¿Cómo funciona este algoritmo? En nuestro mundo natural, las hormigas (inicialmente) vagan de manera aleatoria, al azar, y una vez han encontrado la comida regresan a su colonia dejando un rastro de feromonas. Si otras hormigas encuentran dicho rastro, es probable que estas no sigan caminando aleatóriamente y empiecen a seguir el rastro de feromonas dejado por las anteriores hormigas y reforzándolo si estas encuentran comida finalmente. Sin embargo, con el paso del tiempo, el rastro de feromonas comienza a evaporarse, reduciéndose así su fuerza de atracción.

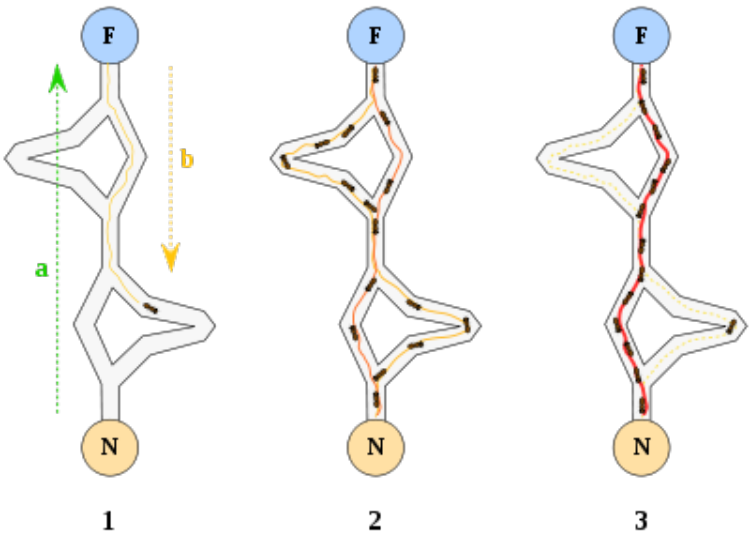

Figura 14: Colonias de hormigas

La idea original de este algoritmo proviene de la observación de la explotación de los recursos alimentarios entre hormigas, en el que las habilidades cognitivas de las hormigas son individualmente limitadas (son ciegas) pero en conjunto son capaces de buscar el menor camino existente entre la fuente de comida y su hormiguero o colonia:

1. La primera hormiga encuentra la fuente de alimentos (F) a través de cualquier camino (a), entonces retorna a la colonia (N), dejando tras ella un rastro de feromonas.
2. Las demás hormigas aleatoriamente siguen cuatro caminos posibles, pero el fortalecimiento de la pista hace más atractiva la ruta más corta. Por lo tanto, las hormigas van tomando la ruta más corta y largas porciones de otras rutas empiezan a perder su rastro de feromonas.

10.2. Algoritmos genéticos

Los algoritmos genéticos son una técnica de búsqueda basada en la teoría de la evolución de Darwin. Esta técnica se basa en los mecanismos de selección que utiliza la naturaleza, de acuerdo a los cuales los individuos más aptos de una población son los que sobreviven, al adaptarse más fácilmente a los cambios que se producen en su entorno.

Hoy en día se sabe que estos cambios se efectúan en los genes de un individuo, y que sus atributos más deseables (los que le permiten adaptarse mejor a su entorno) se transmiten a sus descendientes cuando éste se reproduce sexualmente.

Este algoritmo fue desarrollado por un investigador de la Universidad de Michigan, llamado John Holland. Holland era consciente de la importancia de la selección natural, y a fines de los 60s desarrolló una técnica que se pudo incorporar a un programa informático. Su objetivo era lograr que las computadoras aprendieran por sí mismas. A la técnica que inventó Holland se le llamó originalmente **"planes reproductivos"**, pero se hizo popular bajo el nombre de **algoritmo genético** tras la publicación de su libro en 1975.

¿En qué consiste un algoritmo genético? Un algoritmo genético consiste en una función matemática o una rutina de software que toma como entradas a los individuos y retorna como salidas cuáles de ellos deben generar descendencia para la nueva generación.

Se supone que los individuos (posibles soluciones del problema), pueden representarse como un conjunto de parámetros (que denominaremos *genes*), los cuales agrupados forman una cadena de valores (a menudo referida como *cromosoma*). El alfabeto utilizado para representar los individuos puede ser cualquiera, aunque el más usual es el constituido por el $\{0, 1\}$.

La *función de adaptación* debe ser diseñada para cada problema de manera específica. Dado un cromosoma particular, la función de adaptación le asigna un número real, que refleja el nivel de adaptación al problema del individuo representado por el cromosoma.

Durante la *fase reproductiva* se seleccionan los individuos de la población para cruzarse y producir descendientes, que constituirán, una vez mutados, la siguiente generación de individuos.

La *selección de padres* se efectúa al azar usando un procedimiento que favorezca a los individuos mejor adaptados, ya que a cada individuo se le asigna una probabilidad de ser seleccionado que es proporcional a su función de adaptación. Según dicho esquema, los individuos bien adaptados se escogerán probablemente varias veces por generación, mientras que los peor adaptados al problema, no se escogerán más que de vez en cuando.

Una vez seleccionados dos padres, sus cromosomas se combinan, utilizando habitualmente los *operadores de cruce* y *mutación*:

- **Operador de cruce:** coge dos padres seleccionados y corta sus cadenas de cromosomas en una posición escogida al azar, para producir dos subcadenas iniciales y dos subcadenas finales. Después se intercambian las subcadenas finales, produciéndose dos nuevos cromosomas completos. Ambos descendientes heredan genes de cada uno de los padres.

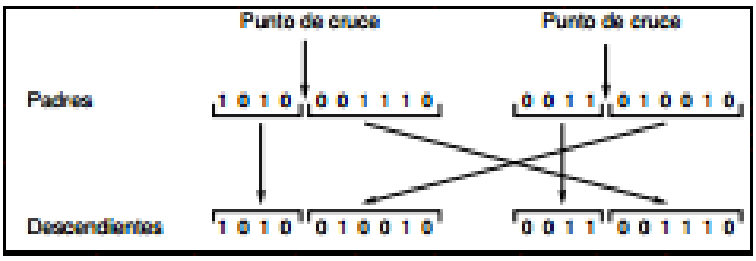

Figura 15: Operador de Cruce

- **Operador de mutación:** se aplica a cada hijo de manera individual, y consiste en la alteración aleatoria (normalmente con probabilidad pequeña) de cada gen componente del cromosoma.

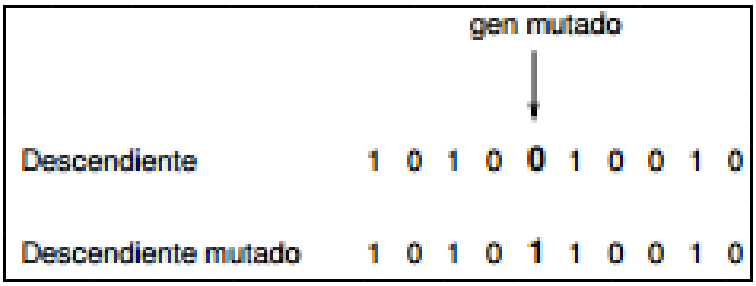

Figura 16: Operador de mutación (5º gen del cromosoma)

Si el Algoritmo Genético ha sido correctamente implementado, la población evolucionará a lo largo de las sucesivas generaciones de tal manera que la adaptación media de todos los individuos de la población, así como la adaptación del mejor individuo se irán incrementando hacia el óptimo global.

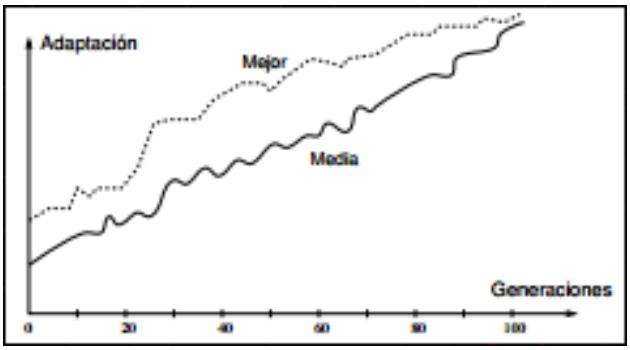

Figura 17: Adaptación Media y Mejor Adaptación

A medida que el número de generaciones aumenta, es más probable que la adaptación media se aproxime a la del mejor individuo.

10.3. Clustering

Clustering es una técnica de minería de datos (data mining) dentro de la disciplina de Inteligencia Artificial que identifica de forma automática agrupaciones o clústeres de elementos de acuerdo a

una medida de similitud entre ellos.

El objetivo fundamental de las técnicas de clustering consiste en identificar grupos o clústeres de elementos tal que la similitud media entre elementos del mismo clúster sea alta o la similitud media entre elementos de distintos clústeres sea baja.

La identificación de clústeres o grupos de elementos se basa en una medida de similitud. Diferentes medidas de similitud dan lugar a diferentes clústeres.

Por ejemplo, dado el siguiente grupo individuos:

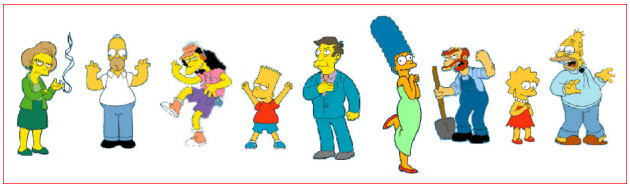

Figura 18: Grupo de Individuos

Podemos hacer, por ejemplo, los siguientes dos tipos de agrupaciones:

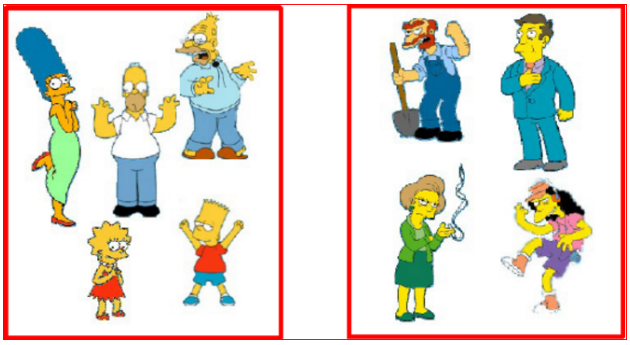

Figura 19: Agrupación 1: Similitud Familia/Trabajo

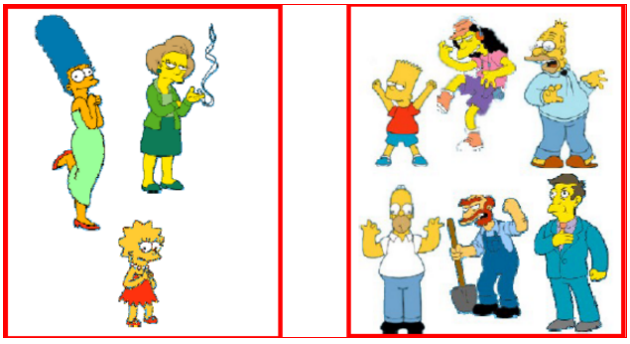

Figura 20: Agrupación 2: Similitud Sexo

Existen principalmente dos tipos diferentes de técnicas de clustering:

1. **Clústering Jerárquico:** construye un árbol que representa las relaciones de similitud entre los distintos elementos. Como la exploración de todos los posibles árboles es computacionalmente intratable, suelen seguirse algoritmos aproximados guiados por determinadas heurísticas.

   Existen dos aproximaciones diferentes al clustering jerárquico:

- **Clústering jerárquico aglomerativo:** se comienza con tantos clústeres como individuos haya y consiste en ir formando (aglomerando) grupos según su similitud.
- **Clústering jerárquico de división:** se comienza con un único clúster y consiste en ir dividiendo clústeres según la similitud entre sus componentes.

En el clústering jerárquico no es necesario especificar el número de clústeres a priori, es posible seleccionarlo a posteriori según un umbral de corte. La estructura jerárquica es cercana a la intuición humana.

La principal desventaja consiste en la acumulación de errores. Es decir, los errores que se comenten en un paso de agrupamiento se propagan durante el resto de la construcción del árbol sin ser posible su reajuste.

2. **Clústering de Partición:** esta técnica de clústering realiza una distribución de los elementos entre un número prefijado de clústeres o grupos.

   Como ejemplo y algoritmo más conocido de esta técnica, hay que destacar el algoritmo K-means.

   El algoritmo K-means es un método de agrupamiento creado por MacQueen en el año 1967, que tiene como objetivo la partición de un conjunto N en k grupos en el que cada observación pertenece al grupo más cercano a la media.

   Dados k clústeres conocidos, cada uno de ellos tiene asociado un centroide (centro geométrico del cluster).

   Los diferentes N puntos se asignan al cluster cuyo centroide esté más cerca (utilizando cualquier métrica de distancia). Iterativamente, se van actualizando los centroides en función de las asignaciones de puntos a clusters, hasta que los centroides dejen de cambiar.

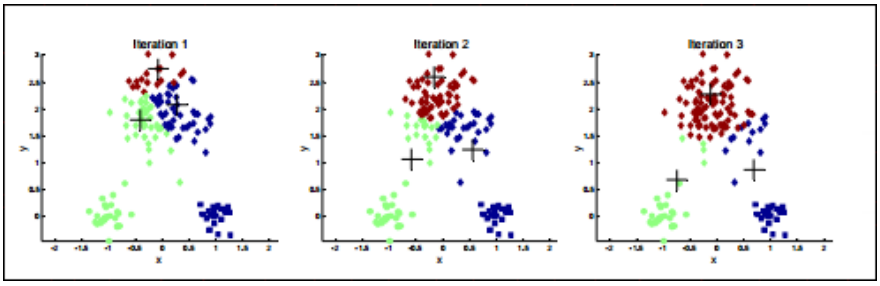

Figura 21: Iteraciones K-Means 1, 2 y 3

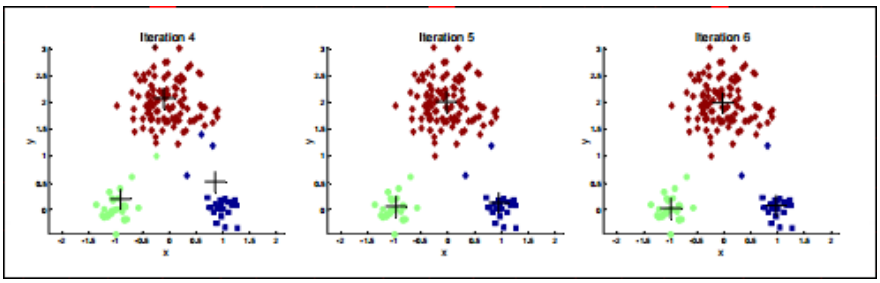

Figura 22: Iteraciones K-Means 4, 5 y 6

El algoritmo, más específicamente, es:

- **Inicialización:**
  - Escoger k centroides aleatoriamente.

- Formar k grupos, asignando cada punto al centroide más cercano.

- **Proceso iterativo:** Mientras que los centroides cambien:
  - Calcular las distancias de todos los puntos a los k centroides.
  - Formar k grupos, asignando cada punto al centroide más cercano.
  - Recalcular los nuevos centroides.

Referencias

1. https://en.wikipedia.org/wiki/Timeline_of_machine_learning
2. **Introducción**, http://cleverdata.io/que-es-machine-learning-big-data/
3. **Introducción**, ftp://decsai.ugr.es/pub/usuarios/castro/Actividades/Redes-Neuronales/Apuntes/Apuntes%20Javier%20Rodriguez%20Blazquez/Conceptos%20basicos.pdf
4. **Tipos de aprendizaje** http://www.cs.us.es/~fsancho/?e=75
5. http://www.centrodeinnovacionbbva.com/noticias/las-cinco-tribus-del-machine-learning
6. **Neural Networks**, http://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/History/history2.html
7. **Neural Networks**, http://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/History/history1.html
8. **Neural Networks**, https://en.wikipedia.org/wiki/Artificial_neural_network
9. **Neural Networks**, https://en.wikipedia.org/wiki/Marvin_Minsky
10. **Perceptrón**, http://sge.wonderville.ca/machinelearning/history/history.html
11. **Perceptrón**, http://www.lcc.uma.es/~jmortiz/archivos/Tema4.pdf
12. **Perceptrón**, ftp://decsai.ugr.es/pub/usuarios/castro/Actividades/Redes-Neuronales/Apuntes/Apuntes%20Javier%20Rodriguez%20Blazquez/Redes%20de%20una%20capa.pdf
13. **Nearest Neighbour**, http://sge.wonderville.ca/machinelearning/history/history.html
14. **Nearest Neighbour**, http://www.synergicpartners.com/una-breve-historia-del-machine-learning/
15. **Nearest Neighbour**, http://www.scholarpedia.org/article/K-nearest_neighbor
16. **Nearest Neighbour**, http://todobi.blogspot.com.es/2016/02/una-breve-historia-del-machine-learning.html
17. **Nearest Neighbour**, https://prezi.com/0oyoj6v-eycs/algoritmo-knn/
18. **Nearest Neighbour**, https://www.ecured.cu/Regla_de_los_K_vecinos_m%C3%A1s_cercanos
19. **Nearest Neighbour**, https://prezi.com/lzilzh3g8xvf/algoritmo-de-los-k-vecinos-mas-proximos/
20. **Máquina de aprendizaje de Turing**, https://hipertextual.com/2011/10/inteligencia-artificial-test-de-turing
21. **Máquina de aprendizaje de Turing**, http://matap.dmae.upm.es/cienciaficcion/DIVULGACION/3/TestTuring.htm
22. **Máquina de aprendizaje de Turing**, https://es.wikipedia.org/wiki/Test_de_Turing
23. **Máquina de aprendizaje de Turing**, http://xamanek.izt.uam.mx/map/cursos/Turing-Pensar.pdf

24. **Máquina de aprendizaje de Turing**, http://enciclopedia.us.es/index.php/Prueba_de_Turing
25. **Stanford Cart**, http://web.stanford.edu/~learnest/sail/cart.htm
26. **Neocognitron**, https://en.wikipedia.org/wiki/Neocognitron
27. **Neocognitron**, http://www.scholarpedia.org/article/Neocognitron
28. **Colonias de Hormigas**, https://es.wikipedia.org/wiki/Algoritmo_de_la_colonia_de_hormigas
29. **Colonias de Hormigas**, http://www.cs.us.es/~fsancho/?e=71
30. **Algoritmos Genéticos**, http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/temageneticos.pdf
31. **Algoritmos Genéticos**, http://www.uv.es/asepuma/X/J24C.pdf
32. **K-means**, https://www.cs.us.es/~fran/curso_unia/clustering.html
33. **K-means**, http://www.sc.ehu.es/jiwdocoj/remis/docs/GarreAdis05.pdf
34. **K-means**, http://elvex.ugr.es/idbis/dm/slides/41%20Clustering%20-%20Partitional.pdf
35. **K-means**, https://prezi.com/1b3laud8egcx/clustering-k-means-k-medoids/