Título del trabajo (elegir uno original)

Domínguez-Adame Ruiz, Alberto
Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
albdomrui@alum.us.es

Vilaplana de Trias, Francisco David
Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
fravilde1@alum.us.es

***Resumen*—Escribir aquí dos párrafos indicando el objetivo principal del trabajo, y un resumen de las conclusiones obtenidas. Cabe mencionar que este documento se ha confeccionado siguiendo el formato de conferencias de IEEE (ver la guía para autores para más información, existen plantillas para Word y LaTeX). Este documento se debe emplear como guía, se pueden añadir nuevas secciones según sea necesario. Es importante dotarlo de un número razonable de referencias bibliográficas.**

***Palabras clave*—Inteligencia Artificial, otras palabras clave...**

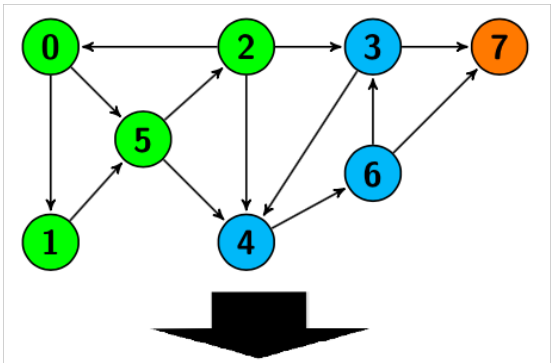
I. Introducción

Para empezar, el aprendizaje se define como la adquisición del conocimiento de algo por medio del estudio, el ejercicio o la experiencia, en especial de los conocimientos necesarios para aprender algún arte u oficio. Al hablar sobre el aprendizaje automático estaríamos entrando en el campo de la Inteligencia Artificial (IA). Podemos definir esta como un programa de computación diseñado para realizar determinadas operaciones que se consideran propias de la inteligencia humana, como el Aprendizaje Automático (Machine Learning), una rama de la inteligencia artificial, cuyo objetivo es el desarrollo de un sistema (modelo matemático que realiza una determinada tarea) el cual tiene un mejor desempeño con la experiencia, dados una serie de datos.

Nuestro estudio hace uso de datos relacionales, los cuales están unidos entre sí (tienen una relación) que queda representado como aristas en un grafo, entendiendo un grafo como un conjunto de nodos unido (o no) mediante aristas. En la Fig. 1 podemos ver un ejemplo de un grafo a partir de unos datos relacionales.

Para la realización de este trabajo se ha hecho uso del entorno de desarrollo Jupyter y de la herramienta notebook. El código está escrito mediante el lenguaje de programación Python, usando principalmente las librerías pandas, NetworkX y scikit-learn, y otras como keras, tensorflow y matplot.

Los modelos de clasificación usados son: KNN, Naive Bayes, Árboles de Decisión y Redes Neuronales.



NodeID	PR
1	0.1
2	0.3
...	...

NID_From	NID_To
1	5
2	3
...	...

Fig. 1. Ejemplo de Grafo

II. Conjunto de datos y Tarea de predicción a realizar

Hemos elegido un dataset (fíjese en la Fig. 2) de la página de streamings en directo twitch.tv, en la que se ve: las visitas, días (ambos de tipo entero) que transmitido un canal en directo, si tiene contrato o no con twitch (partner de tipo Boolean) y si el contenido del canal (id de tipo Integer) es o no para adultos (mature de tipo Boolean). Al ser la propia Twitch la que ofrece este contrato, hemos decidido que la predicción para nuestros modelos sea el atributo partner (true/false). Las aristas (edges) representan si dos usuarios tienen una relación de amistad.

Este dataset tenía dos tipos de id asociados a cada usuario, uno generado aleatoriamente y otro donde al usuario se le asignaba un valor entre 0 y el numero total de usuarios que recoge el conjunto de datos, por lo que decidimos que eliminar el id generado de forma aleatoria para mayor claridad y sólo hace falta un id por usuario para identificarlos.

Para este estudio hemos decidido que el atributo a predecir será partner. Para ello, haremos uso de los atributos: days, views y mature (el id no es relevante en este caso). Lo hemos planteado como una tarea de clasificación binaria, ya que partner solo tiene dos valores posibles (True o False).

Por último hemos codificado (codificación one-hot) los valores de mature de Booleano a Integer, pasando True-False a 1-0 respectivamente, para que sea más eficiente estimar las probabilidades del atributo partner.

	id	partner	days	views	mature
0	2299	False	1459	9528	0
1	153	False	1629	3615	1
2	397	False	411	46546	1
3	5623	False	953	5863	1
4	5875	False	741	5594	1
...	...	...	...	...	...
7121	3794	False	2624	3174	0
7122	6534	False	2035	3158	1
7123	2041	False	1418	3839	1
7124	6870	False	2046	6208	1
7125	3919	False	1797	3545	0

Fig. 2. Tabla de Datos

III. Métricas Relacionales

Como este es un trabajo de desarrollo de modelos orientados al aprendizaje automático grafos, hemos elegido una serie de métricas relacionales que aplicaremos como atributos relacionales. En concreto hemos elgido tres métricas:

1) **Betweenness centrality** (Centralidad de intermediación): La centralidad en un grafo puede ser entendida como una medida que determina la relevancia de un nodo dentro del grafo y permite comparar o contrastar dicho vértice con otros. La Betweenness Centrality es una medida de centralidad que cuantifica la frecuencia en la que un nodo se encuentra en el camino más corto entre dos nodos determinados. Cuando en un grafo existen nodos de alta intermediación, estos suelen jugar un rol importante en la estructura a la que pertenecen. Estos nodos también poseen capacidades de ser controladores o reguladores de los flujos de información dentro de la estructura total del grafo.
2) **Clustering** (agrupamiento): es una tarea que tiene como finalidad principal lograr el agrupamiento de nodos, para lograr construir subconjuntos de datos conocidos como Clusters.
3) **Degree** (grado): Número de aristas (relaciones) que inciden sobre el nodo.

IV. Algoritmos de Aprendizaje Automático

A continuación exponemos los algoritmos que hemos utilizado con nuestro dataset:

A. KNN

El algoritmo kNN (del inglés *k Nearest Neighbors*, k vecinos más cercanos), es un clasificador de aprendizaje supervisado no paramétrico, por lo que la cantidad de parámetros del modelo coincide exactamente con la cantidad de ejemplos de entrenamiento. Se basa en la proximidad de los k nodos más cercanos calculando la cercanía a partir de los atributos, si estos fueran discretos sería necesaria una codificación de los mismo.

Analizando nuestro dataset, se puede observar como el atributo "*views*" toma valores de mayor magnitud con respecto al resto lo que provoca que tenga más en cuenta para la predicción del modelo. Con el fin de evitar esta situación se hace uso de la normalización a los atributos del conjunto de entrenamiento. Existe una gran variedad de maneras de normalizar un atributo numérico, nosotros hemos optado por el método min-max [1] que consiste en que a cada atributo restarle el valor mínimo de este y dividir esta diferencia entre la resta del valor máximo y mínimo:

$$v_i' = \frac{v_i - m}{M - m} \tag{1}$$

Para realizar kNN es necesario dar un valor al hiperparámetro k, referido al número de vecinos que se va a tener en cuenta a la hora de clasificar un nuevo dato, nosotros le hemos dado valores del uno al diez para más tarde evaluar el rendimiento de cada uno y elegir el mejor. También existen métodos para calcular la distancia, en nuestro caso hemos usado:

- Distancia de *Hamming* para el dataset sin atributos relacionales:

$$d(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{n} \mathbb{1}(x_i \neq x_i') \tag{2}$$

  Hemos usado esta métrica ya que según la docmentación de la librería [2] es la recomendada para valores enteros.

- Distancia *Manhattan* para el dataset con atributos relacionales:

$$d(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{n} |x_i - x_i'| \tag{3}$$

  En este caso es la adecuada al tratarse de valores numéricos reales.

Para el cálculo del modelo ha sido necesaria la clase neighbors de la librería sklearn. Por último, una vez se ha construido el modelo lo evaluamos mediante validación cruzada (cross validation) con 10 pliegues. Se divide el dataset según el numero de pliegues, siendo uno de estos el que se usará para probar el modelo y el resto para entrenarlo, de tal manera que cada pliegue acabe siendo usado como subconjunto de prueba. Cada una de estas iteraciones dará como resultado una tasa de acierto, la media de estas será el valor asociado a cada k. Finalmente se comparan y se escoge el valor de k que obtuviera la media la más alta.

B. Naive Bayes

Naive Bayes se trata de un modelo de clasificación paramétrico, ya que no depende de la cantidad de ejemplos que contega el conjunto de entrenamiento. Los valores deben ser discretos, y en caso contrario deberán someterse a un proceso de discretización, donde se divide el rango de los valores de los atributos en subintervalos, los cuales constituiran los posibles valores de la variable discretizada. En nuestro era necesaria la discretización al tratarse de enteros con una gran diferencia

entre los valores. Para realizar la tarea de clasificación se aprende primero los siguientes valores de los parámetros:

- La probabilidad de cada clase c:

$$\mathbb{P}(c) = \frac{N_c}{N} \tag{4}$$

  Donde N_c es la cantidad total de ejemplos que pertenecen a la clase c, y N la cantidad total de los ejemplos que hay en el conjunto de entrenamiento.

- La probabilidad de para cada atributo X, cada posible valor de x el atributo y cada clase c:

$$\mathbb{P}(X = x|c) = \frac{N_{X=x,c}}{Nc} \tag{5}$$

  Donde $N_{X=x,c}$ es la cantidad total de ejemplos del conjunto de entrenamiento que pertenecen a la clase c en los que el atributo X toma el valor x.

A la hora de discretizar los atributos hemos usado KBinsDiscretizer [3], de la libretria sklearn. Para los parámetros usamos una codificación one-hot, ya que algunos atributos tiene valores numéricos muy elevados comparados con otros. Por esta razón hemos elgido esta codificación, pues proporciona mejor resultado que otras (como por ejemplo la de tipo ordinal).

A la hora de construir el modelo se ha usado MultinomialNB(alpha=k) [4], pues es el más apropiado para clasificación de atributos numéricos discretizados. El hiperparámetro usado, k, se usa para realizar el suavizado de Laplace. Dicho suavizado se realiza cuando alguna de las probabilidades calculadas para una clase c toma el valor 0, haciendo que un ejemplo nunca se clasifique en dicha clase c. La fórmula que nos indica el suavizado de Laplace es la siguiente:

$$\mathbb{P}(X = x|c) = \frac{N_{X=x,c} + 1}{N_c + k|X|} \tag{6}$$

Para la evaluación de este modelo también hemos usado el método de validación cruzada con 10 pliegues.

C. Árboles de Decisión

Los Árboles de Clasificación y regresión (del inglés CART, clasification and regresion tree) son modelos para adecuados para tareas de tanto de clasificación como de regresión dados atributos necesariamente numéricos, ya sean continuos o discretos.En un CART cada nodo interno está etiquetado con un atributo y un valor umbral para dicho atributo, mientras que cada hoja está etiquetada con una clase si es una tarea de clasificación, o un número en caso de que se trate de una tarea de regresión. Dado nuestro conjunto de datos, la tarea a abordar es una de clasificación, por lo que elegimos el tipo de árbol correspondiente.

El algoritmo CART para el aprendizaje del modelo es tal que dado un conjunto de ejemplos de entrenamiento $\mathscr{D}$, denotamos $\mathscr{D}_n$ al subconjunto de ejemplos que satisfacen todas las condiciones de los nodos desde la raíz hasta hasta n. Dado un nodo n un atributo X, siendo x_1, x_2, ... , x_k la secuencia de distintos valores que toma X en $\mathscr{D}_n$, estando estos ordenados

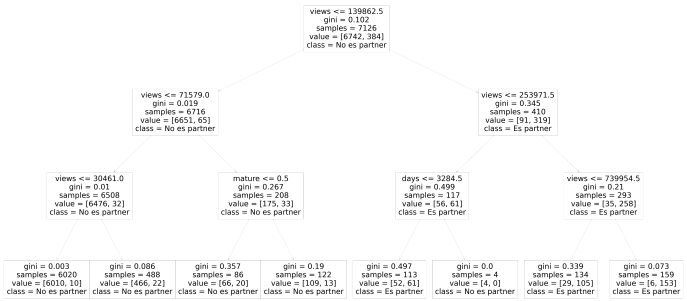


Fig. 3. Árbol de Decisión

de menor a mayor. Siendo $u_i = \frac{(x_i + x_{i+1})}{2}$ el umbral candidato para X y con i en el rango $(1, 2, ..., k-1)$. A continuación se divide $\mathscr{D}_n$ según el umbral. Si los ejemplos de X toman un valor menor o igual que u estos se agrupan en $\mathscr{D}_n^{Izq}$, mientras que si son mayores que u se agruparan en en el subconjunto $\mathscr{D}_n^{Der}$. Por último, se hace uso de una función I que permite medir la impureza de un conjunto de ejemplos, para poder elegir el par (X, u) que nos de una mejor partición de $\mathscr{D}_n$, siendo dicha función:

$$\textbf{arg mín } \frac{|\mathscr{D}_n^{Izq}|}{|\mathscr{D}|} I(\mathscr{D}_n^{Izq}) + \frac{|\mathscr{D}_n^{Der}|}{|\mathscr{D}|} I(\mathscr{D}_n^{Der}) \tag{7}$$

Una representación del algoritmo sería:

CART($\mathscr{D}$)
1 **Si** *NODIVISIBLE($\mathscr{D}$)* **entonces**
2 **Devolver** un nodo etiquetado con *ETIQUETA*($\mathscr{D}$)
3 **Si no entonces**
4 Elegir el par (X, u) que proporcione la mejor
5 partición $(\mathscr{D}^{Izq}, \mathscr{D}^{Der})$ de $\mathscr{D}$
6 $T_1 \leftarrow$ CART($\mathscr{D}^{Izq}$)
7 $T_2 \leftarrow$ CART($\mathscr{D}^{Der}$)
8 **Devolver** un nodo etiquetado con (X, u) y cuyos
9 hijos sean T_1 y T_2

La función de impureza usada para una tarea de clasificación suele serel índice de Gini, pues estima la probabilidad de clasificar incorrectamente un ejemplo. Tal que:

$$G(\mathscr{D}) = \sum_{c \in C} \hat{\pi}_c (1 - \hat{\pi}_c) = 1 - \sum_{c \in C} \hat{\pi}_c^2 \tag{8}$$

Para abordar esta tarea hemos hecho uso de las librerías sklearn y matplotlib para mostrar el árbol resultado por pantalla. Dada la naturaleza hemos decidido usar la función DecisionTreeClassifier de la clase tree, puesto que nos proporciona un árbol de decisión el cual ha sido limitado a una profundidad máxima de 5. Para terminar, realizamos una evaluación del modelo mediante el método score, que calcula el rendimiento del modelo, dados por separado el conjunto de ejemplos de prueba y la clase de cada uno de estos ejemplos.

D. Redes Neuronales

Las Redes Neuronales son un modelo de predicción inspirado en el funcionamiento del cerebro, permitiendo que

programas informáticos reconozcan patrones y resuelvan problemas comunes en el campo de la Inteligencia Artificial, Aprendizaje Automático y Aprendizaje Profundo. Son un conjunto de neuronas artificiales, cada una ellas con un sesgo y un peso asociado e interconectadas entre sí. Se agrupan en forma de capas, una de entrada donde se reciben los datos del problema, una o varias capas ocultas y una de salida para dar la predicción según los datos introducidos. Una neurona mandará información a otra neurona de la siguiente capa si la salida de esta es superior al valor del sesgo, si no, no transmitirá informacion alguna.

A la hora de entrenar el modelo se le asginarán unos valores aleatorios a los pesos de cada neurona, el peso se irá actualizando en función del siguiente algoritmo:

1 Inicializar los pesos aleatoriamente
2 **Repetir** hasta que se cumpla la condición de terminación:
3 **Para cada** ejemplo de entrenamiento
4 **Si** la salida del perceptrón es incorrecta entonces
5 Actualizar los pesos
6 **Devolver** los pesos

Cuando en el bucle del algoritmo se han considerado todos los ejemplos se dice que ha transcurrido una época.

En nuestro caso (fíjese en la Fig. 2), tenemos una capa de entrada de 3 neuronas, correspondientes a cada uno de los atributos usados para calcular el objetivo. Para la capa oculta hemos optado por un total de 10 neuronas con función de activación rectificador (ReLU). Al tratarse de un problema de clasificación binaria, la última capa de nuestra red neuronal estará compuesta por una neurona con funcion de activación sigmoide, ya que los valores de salida oscilarán entre 0 y 1. Finalmente el algoritmo nos dará una tasa de aciertos, cada usuario será clasificado como Partner (1) si esta es mayor que 0.5.

Para el entrenamiento de la Red Neuronal usamos la entropía cruzada binaria como funcion de coste y el método de descenso por el gradiente (SGD, Stochastic Gradient Descent) para actualizar los pesos en la direccion $-\nabla C$.

· *Binary Crossentropy*:

$$-ylog_e(a^L) - (1-y)log_e(1-a^L) \tag{9}$$

Para la evaluación del modelo hemos dividido los datos en dos subconjuntos de prueba y entrenamiento en un 25% y un 75% respectivamente, aportándonos unos resultados con un rendimiento excelente.

V. Resultados

En esta sección se detallarán tanto los experimentos realizados como los resultados conseguidos:

- Los experimentos realizados, indicando razonadamente la configuración empleada, qué se quiere determinar, y como se ha medido.
- Los resultados obtenidos en cada experimento, explicando en cada caso lo que se ha conseguido.
- Análisis de los resultados, haciendo comparativas y obteniendo conclusiones.

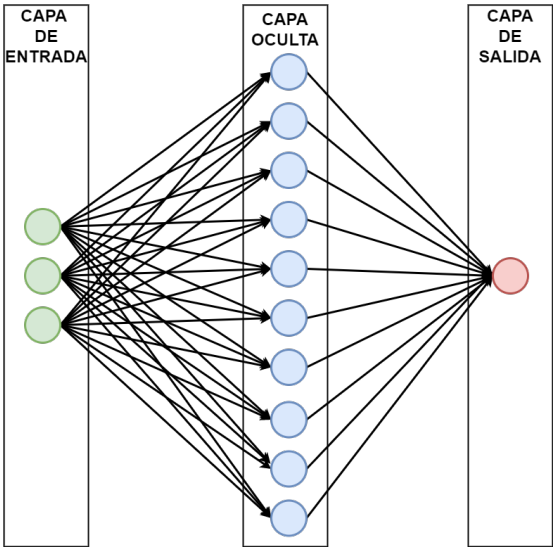


Fig. 4. Nuestra Red Neuronal

VI. Conclusiones

Finalmente, se dedica la última sección para indicar las conclusiones obtenidas del trabajo. Se puede dedicar un párrafo para realizar un resumen sucinto del trabajo, con los experimentos y resultados. Seguidamente, uno o dos párrafos con conclusiones. Se suele dedicar un párrafo final con ideas de mejora y trabajo futuro.

VII. Bibliografía

Referencias

[1] https://scikit-learn.org/stable/modules/preprocessing.html#normalization
[2] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.DistanceMetric.html
[3] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.KBinsDiscretizer.html
[4] https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
[5] https://scikit-learn.org/stable/modules/naive_bayes.html
[6] https://www.ibm.com/cloud/learn/neural-networks
[7] https://www.draw.io
[8] https://scikit-learn.org/stable/modules/tree.html