

GRUPO 33
PRÁCTICA 1
APRENDIZAJE AUTOMÁTICO

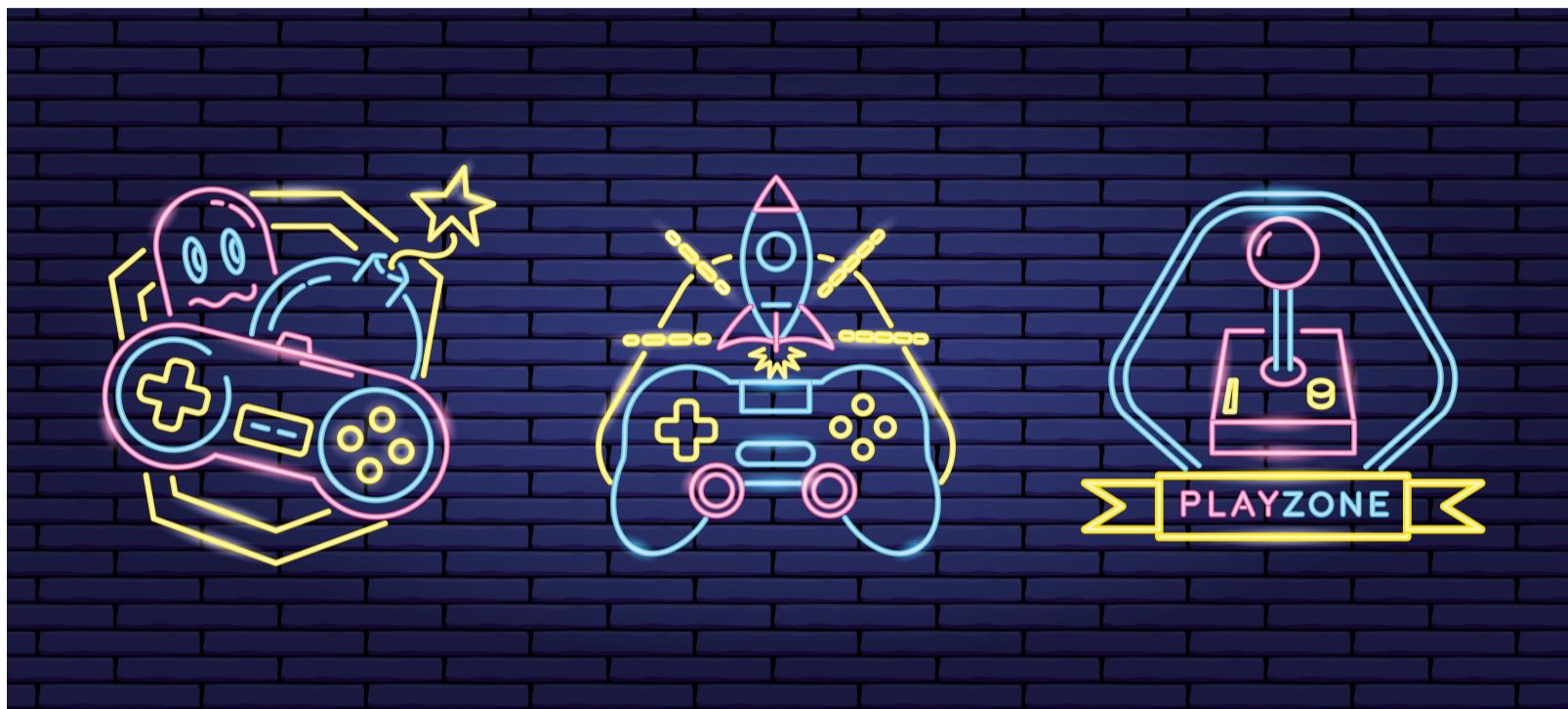
APRENDIZAJE AUTOMÁTICO

1. Introducción.....	3
2. Fase 1	4
3. Fase 2	7
4. Fase 3	8
5. Preguntas Fase 4	9
1. ¿Qué diferencias hay a la hora de aprender esos modelos con instancias provenientes de un agente controlado por un humano y uno automático?	9
2. Si quisieras transformar la tarea de regresión en clasificación ¿Qué tendrías que hacer? ¿Cuál crees que podría ser la aplicación práctica de predecir la puntuación?. 9	
3. ¿Qué ventajas puede aportar predecir la puntuación respecto a la clasificación de la acción? Justifica tu respuesta.....	10
4. ¿Crees que se podría conseguir alguna mejora en la clasificación incorporando un atributo que indicase si la puntuación en el instante actual ha descendido o ha bajado?	10
5. Conclusiones	11
6. Anexos	12
7.1. Conjuntos de datos	12
7. Bibliografía.....	13

1.

INTRODUCCIÓN

El presente documento desarrollará la creación de un proceso de aprendizaje automático para crear un agente del juego Pacman. En este caso dicho agente jugará como Pacman y no como los fantasmas, los cuales tienen un movimiento aleatorio.



Para el desarrollo de la práctica se ha utilizado un repositorio de tipo Git ubicado en la plataforma Github. No se indica enlace ya que aún permanece de tipo privado y si se desea acceso al mismo se puede solicitar utilizando los datos de contacto de los alumnos.

En Python se ha utilizado la versión 2.7, instalando los plugin *java-bridge* y *python-weka-wrapper*.

En la fase 1, se realizará una preparación de los datos en conjuntos de entrenamiento, validación y test. Dichos datos serán generados tanto de modo manual como automático y serán incluidos en los anexos de la presente memoria.

Las fases 2 y 3 buscarán la mejor manera de tratar dichos datos para que el funcionamiento del agente sea óptimo.

La fase 4 integrará los modelos generados en las fases 2 y 3 para visualizar nuestros modelos sobre Pacman en una pantalla en tiempo real.

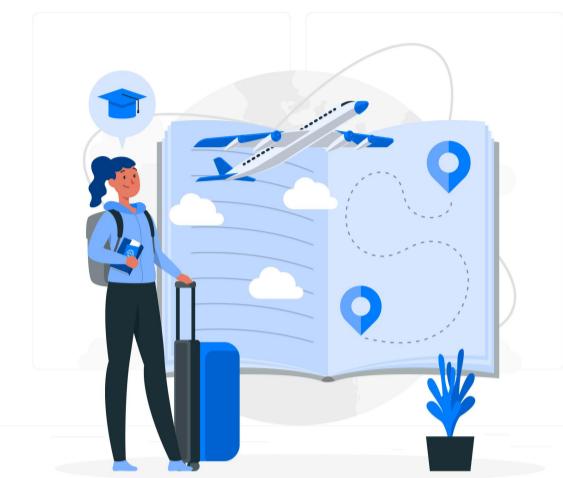
Posteriormente se tratarán las conclusiones y anexos que redundarán en el cierre de la presente práctica del modo “menos predictivo posible”.

2.

FASE 1

En el tutorial 3 ya escribimos un fichero .ARFF interpretable por WEKA, la cual contiene la información necesaria para la interpretación de cada clase así como la dirección tomada por Pacman.

Se muestra a continuación la matriz de ficheros generados, así como los mapas utilizados y el comando requerido para su generación. Cabe destacar que se ha generado el fichero cambiando el nombre manualmente sobre el código ya que la automatización de dicha acción requería tiempo no imprescindible para la realización de la práctica.

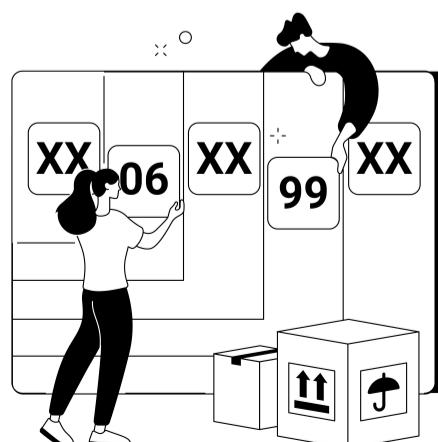


Fichero	Nº Inst	Mapas	Ghost	Control	Comando
training_keyboard.arff	4.000	20Hunt	RANDOM	Teclado	<code>python busters.py -g RandomGhost -l 20Hunt</code>
		bigHunt			<code>python busters.py -g RandomGhost -l big-Hunt</code>
		capsuleClassic			<code>python busters.py -g RandomGhost -l capsuleClassic</code>
		classic			<code>python busters.py -g RandomGhost -l classic</code>
		contestClassic			<code>python busters.py -g RandomGhost -l contestClassic</code>
test_samemaps_keyboard.arff	1.000	20Hunt	RANDOM	Teclado	<code>python busters.py -g RandomGhost -l 20Hunt</code>
		bigHunt			<code>python busters.py -g RandomGhost -l big-Hunt</code>
		capsuleClassic			<code>python busters.py -g RandomGhost -l capsuleClassic</code>
		classic			<code>python busters.py -g RandomGhost -l classic</code>
		contestClassic			<code>python busters.py -g RandomGhost -l contestClassic</code>
		bigHunt			<code>python busters.py -g RandomGhost -l 20Hunt</code>
test_othermaps_keyboard.arff	1.000	mediumClassic	RANDOM	Teclado	<code>python busters.py -g RandomGhost -l mediumClassic</code>
		smallHunt			<code>python busters.py -g RandomGhost -l smallHunt</code>
		minimaxClassic			<code>python busters.py -g RandomGhost -l minimaxClassic</code>
		newMap			<code>python busters.py -g RandomGhost -l newMap</code>
		oneHunt			<code>python busters.py -g RandomGhost -l one-Hunt</code>

APRENDIZAJE AUTOMÁTICO

Fichero	Nº Inst	Mapas	Ghost	Control	Comando
trainingTutorial1.arff	10.000	20Hunt	RANDOM	Automático	python busters.py -n 3 -l 20Hunt -p RandomPAgent -g RandomGhost -k 4 -t 0.01
		bigHunt			python busters.py -n 3 -l bigHunt -p RandomPAgent -g RandomGhost -k 4 -t 0.01
		capsuleClassic			python busters.py -n 3 -l capsuleClassic -p RandomPAgent -g RandomGhost -k 4 -t 0.01
		classic			python busters.py -n 3 -l classic -p RandomPAgent -g RandomGhost -k 4 -t 0.01
		contestClassic			python busters.py -n 3 -l contestClassic -p RandomPAgent -g RandomGhost -k 4 -t 0.01
testSameMapsTutorial1.arff	2.000	20Hunt	RANDOM	Automático	python busters.py -n 3 -l 20Hunt -p RandomPAgent -g RandomGhost -k 4 -t 0.01
		bigHunt			python busters.py -n 3 -l bigHunt -p RandomPAgent -g RandomGhost -k 4 -t 0.01
		capsuleClassic			python busters.py -n 3 -l capsuleClassic -p RandomPAgent -g RandomGhost -k 4 -t 0.01
		classic			python busters.py -n 3 -l classic -p RandomPAgent -g RandomGhost -k 4 -t 0.01
		contestClassic			python busters.py -n 3 -l contestClassic -p RandomPAgent -g RandomGhost -k 4 -t 0.01
testOtherMapsTutorial1.arff	200	mediumClassic	RANDOM	Automático	python busters.py -l mediumClassic -p RandomPAgent -g RandomGhost -t 0.01
		smallHunt			python busters.py -l smallHunt -p RandomPAgent -g RandomGhost -t 0.01
		minimaxClassic			python busters.py -l minimaxClassic -p RandomPAgent -g RandomGhost -t 0.01
		newMap			python busters.py -l newMap -p RandomPAgent -g RandomGhost -t 0.01
		oneHunt			python busters.py -l oneHunt -p RandomPAgent -g RandomGhost -t 0.2

Para los atributos necesarios sobre cada instancia, se han estimado los siguientes siguiendo la lógica “más es mejor”.



Estado actual:

1. `@ATTRIBUTE coords_x Numeric`
2. `@ATTRIBUTE coords_y Numeric`
3. `@ATTRIBUTE nearest_dot_distance Numeric`
4. `@ATTRIBUTE phantom_count Numeric`
5. `@ATTRIBUTE current_score Numeric`
6. `@ATTRIBUTE capsules_count Numeric`

Acción realizada:

1. `@ATTRIBUTE pacman_current_direction {East,North,South,West,Stop}`

Información sobre el siguiente turno:

1. `@ATTRIBUTE pacman_next_direction {East,North,South,West,Stop}`
2. `@ATTRIBUTE next_score {East,North,South,West,Stop}`



3.**FASE 2**

Para proceder a la comparativa de los algoritmos hemos utilizado la herramienta Experimenter de Weka, con los algoritmos listados a continuación. Dado que algunos de ellos no podían ofrecernos la generación directa de gráficas, hemos generado la siguiente tabla y el siguiente gráfico para la comparativa de los mismos. Se han generado 10 iteraciones por dataset y algoritmo y se ha clasificado con un cross-validation de 10 folds.

Como clase se ha utilizado pacman_current_direction. Se han discretizado los datos correspondientes a la distancia de los fantasmas (None = -1).

Ficheros de entrenamiento:

training_keyboard.arff
training_tutorial1.arff

Ficheros de test:

test_othermaps_keyboard.arff
test_othermaps_tutorial1.arff
test_samemaps_keyboard.arff
test_samemaps_tutorial1.arff

Algoritmo	Porcentaje acierto	Satisfactorio
ZeroR	29.96	V
PART	47.84	V
NaiveBayes	30.58	F
IBk	62.24	V
DecisionTable	42.24	V
J48	51.11	V

Denotamos como mejor algoritmo clasificador IBk.

4.**FASE 3**

Hemos elegido los siguientes algoritmos y en base al tratamiento de la fase 1 y 2 no se ha requerido tratar nada más.

El algoritmo de mejores resultados es random Forest. Se ha contrastado contra los ficheros de test.

Ficheros de entrenamiento:

training_keyboard.arff

training_tutorial1.arff

Ficheros de test:

test_othermaps_keyboard.arff

test_othermaps_tutorial1.arff

test_samemaps_keyboard.arff

test_samemaps_tutorial1.arff

Algoritmo	Porcentaje acierto	Satisfactorio
MultiLayerPerceptron	33.90	F
RandomForest	63.04	V

5. PREGUNTAS FASE 4

Se ha seleccionado el agente automático RandomForest (IBk). Se debe construir el agente automático para utilizar el aprendizaje continuo de los logs que genera y retroalimentan el propio algoritmo a través de la librería de Weka utilizable en Python (entre otros lenguajes).

1. ¿QUÉ DIFERENCIAS HAY A LA HORA DE APRENDER ESOS MODELOS CON INSTANCIAS PROVENIENTES DE UN AGENTE CONTROLADO POR UN HUMANO Y UNO AUTOMÁTICO?

Las instancias automáticas no saben diferenciar por sí mismas si el movimiento es “bueno” o “malo”, aunque sí pueden estimarlo en base a la puntuación actual y la del turno anterior: si ha aumentado significativamente o si desciende dado que no nos comemos ningún fantasma ni comida. Los automáticos (sin aprendizaje) utilizan modelos estadísticos y generan una cantidad ingente de datos de peor calidad, mientras que los datos manuales son de una calidad incomparable pero mucho más tediosos de generar (y por ende, hay menor cantidad). A la hora de aprender, son mejores los manuales, pero insuficientes en cantidad. Juntando con los automáticos, tenemos un buen dataset. Se recomendaría distribuirlo con los datos manuales como validación y de este modo tener datos contrastables.

Al aprender de datos provenientes del humano, son más difíciles de generar (y por ende, hay menos) pero son de mayor calidad siempre y cuando el usuario sepa jugar a Pacman. No obstante, los automáticos mejorarán la calidad progresivamente en base a la experiencia y acabarán teniendo igual o mayor calidad que los humanos con cantidades mucho mayores, pero para ello necesitan entrenamiento.

2. SI QUISIERAS TRANSFORMAR LA TAREA DE REGRESIÓN EN CLASIFICACIÓN ¿QUÉ TENDRÍAS QUE HACER? ¿CUÁL CREEES QUE PODRÍA SER LA APLICACIÓN PRÁCTICA DE PREDICIR LA PUNTUACIÓN?

Habría que transformar el modelo con Weka interoperando entre los algoritmos de los dos tipos. Sería una aplicación práctica para contrastar la clasificación.

3. ¿QUÉ VENTAJAS PUEDE APORTAR PREDECIR LA PUNTUACIÓN RESPECTO A LA CLASIFICACIÓN DE LA ACCIÓN? JUSTIFICA TU RESPUESTA.

Priorizaría el ganar puntuación y en base a esto predecir en segundo nivel la predicción del movimiento, ya que estaría haciendo énfasis en la puntuación.

4. ¿CREEES QUE SE PODRÍA CONSEGUIR ALGUNA MEJORA EN LA CLASIFICACIÓN INCORPORANDO UN ATRIBUTO QUE INDICASE SI LA PUNTUACIÓN EN EL INSTANTE ACTUAL HA DESCENDIDO O HA BAJADO?

Rotundamente sí. De esta manera podríamos clasificar el movimiento actual como bueno y que el algoritmo tienda a esta clase de movimientos siempre que sea posible, así como el camino hasta llegar al mismo para generar un aprendizaje automático eficiente.

Sí, dado que puede comparar si el paso que se ha dado anteriormente es factible, válido y óptimo.

5.**CONCLUSIONES**

En la presente práctica apreciamos lo necesario que es un buen sistema de información y el tratamiento de la misma. En caso de poseer un buen acceso y calidad de la información almacenada, podemos generar tanto modelos predictivos como automatizaciones sobre tareas ó labores que en el día a día nos atañen y mejorarán significativamente nuestra calidad de vida.

Enlazando con lo anterior y aplicando un contexto sobre el videojuego clásico Pacman, hemos podido apreciar un notable aumento progresivo de la calidad de juego sobre el modelo estadístico - *el cual cuando los fantasmas se mueven aleatoriamente y hay paredes cercanas se vuelve loco* - pudiendo predecir y mejorar un comportamiento basado en el comportamiento humano y estadístico sobre dicho juego.

La selección del algoritmo a aplicar sobre cada determinado caso requiere de experiencia de “campo” más que de un proceso de automatización por el que un algoritmo sugiere un algoritmo - *ojo con la dicibilidad, no todo es computable* - y es parte de la experiencia del programador y analista que deseen mejorar el comportamiento automatizado de un agente.

A título personal la práctica nos ha resultado amena y motivante de modo que se ha realizado en un tiempo récord y con la calidad que estimamos conveniente.

Como crítica constructiva nos gustaría plantear que a futuro se lleve a cabo un reto continuo online desde el inicio de la asignatura hasta su final, de este modo la competición no será únicamente al final sino que requerirá de una dedicación exhaustiva a lo largo del desarrollo de la asignatura más que de un empujón en un momento determinado. No obstante recalcar que estamos encantados con el desarrollo de la práctica.



6. ANEXOS

7.1. CONJUNTOS DE DATOS

Se presentarán junto a la entrega de la práctica en el mismo fichero comprimido.

7.

BIBLIOGRAFÍA

Imágenes: [freepik.com](https://www.freepik.com)

Weka: <https://www.cs.waikato.ac.nz/ml/weka/>

JetBrains: <https://www.jetbrains.com/help/pycharm/configuring-python-interpreter.html#add-existing-interpreter>

OCW UC3M: <http://ocw.uc3m.es/ingenieria-informatica/herramientas-de-la-inteligencia-artificial/contenidos/transparencias/TutorialWeka.pdf>

Regresión lineal: <https://programmerclick.com/article/2120201170/>

