Examen Aprendizaje Automático y Minería de Datos

Grado en Desarrollo de videojuegos, enero 2024

Instrucciones generales

El examen consta de dos partes, una teórica que vale 2 puntos y una práctica que vale 8 puntos.

La parte teórica se entregará en el campus con el nombre del alumno sin espacios y en uno de los siguientes formatos: txt, docx, pdf. No se corregirá si está en otro formato diferente y se calificará como no presentada.

La parte práctica se debe entregar en un Jupiter Notebook dentro de un zip con todos los recursos que necesite para ejecutar. Todo, tanto la implementación como la posible explicación debe estar en el notebook. Para ello podéis hacer uso de las celdas de markdown en caso de necesitar escribir alguna explicación o contestar alguna pregunta. El fichero de Jupiter notebook debe ser ejecutable por si mismo sin necesidad de intervención del profesor, es decir, aseguraros que lo que entreguéis ejecuta y tiene todos los recursos necesarios para que ejecute. La única excepción es el posible uso de librerías, aunque inicialmente no deberíais necesitar ninguna más allá de las vistas en clase. Si la entrega no es correcta y no ejecuta (por ejemplo faltan los datos, las rutas son absolutas y no encuentra los datos del dataset, etc) se penalizará la nota de la parte práctica del examen con hasta -1 puntos.

En el zip también debéis incluir vuestra librería con la implementación del perceptrón multicapa generalizado para múltiples capas y neuronas por capa. Esta librería se debe basar en el código entregado en la práctica 5 y 6. Puede tener modificaciones, pero si el código es radicalmente diferente se suspenderá el examen y se tratará como una copia del mismo.

Se permite para la realización de la parte práctica y teórica el uso de los recursos del campus, apuntes y cualquier material de apoyo que traigais. No está permitido el uso de ChatGPT o herramientas de IA similares.

Parte teórica

Tiempo disponible 30 minutos.

Contestad en un documento de texto y entregadlo en el campus virtual con vuestro nombre en el nombre del fichero y dentro del mismo como primera línea del archivo. Poned el número de la pregunta que estáis contestando y a continuación la respuesta.

Pregunta 1 (0.5 puntos)

Si tengo un dataset de clasificación con imágenes de tamaño 128x128 y usando un perceptrón multicapa no he conseguido buenos resultados a la hora de predecir el tipo de imagen que se trata. ¿Qué transformaciones harías a los datos y/o al modelo para intentar mejorar los resultados de dicho perceptrón? Nota, al final la clasificación la debe hacer el perceptrón, lo que se pide es qué podemos hacer antes de aplicar el perceptrón. Justifica la respuesta.

Pregunta 2 (0.5 puntos)

Cuál de los siguientes modelos tiene capacidad de modelar problemas no lineales. Justifica tu respuesta.

- Árbol de decisión basado en ID3.
- Una Red de neuronas profunda con 3 capas convolucionales y una última capa Linear() the Pythorch.
- Un Perceptrón multicapa con función de activación y=m*z+b siendo z el valor de la neurona.
- Ninguno de los anteriores.

Pregunta 3 (0.5 puntos)

Tenemos un modelo de red neuronal que nos da un accuracy de entrenamiento del 70% y un accuracy de validación del 50%. Sabiendo que nuestro baseline es de una precisión de clasificación del 65%. ¿Qué estrategia seguirías para intentar mejorar los resultados?

Pregunta 4 (0.5 puntos)

Disponemos de 200 datos de entrenamiento de una partida guardada por un jugador y con estos datos, queremos construir un modelo de machine learning que intente imitar al jugador para desarrollar una IA. Los datos que se han guardado son: La posición del jugador (en que casilla está), el input que se ha realizado (moverse a la izquierda, a la derecha, disparar o saltar), la posición de los enemigos en el escenario (casilla en la que están) y el contenido de las 8 casillas contiguas al jugador (valores que puede tomar las casillas: vacía, suelo, enemigo, item). El juego se mueve de forma continua, pero está dividido a nivel lógico en forma de cuadricula de tamaño 1x1 unidades de juego.

Si queremos usar un perceptrón multicapa. ¿Qué número de neuronas de entrada y de salida tendrá el perceptrón y explica brevemente como lo entrenarías? Justifica la respuesta.

Parte práctica

Tiempo máximo para la realización de la parte práctica 2h y 30 minutos.

Disponemos de un dataset sobre demencia con los siguientes campos:

- Subject ID: identificación del paciente.
- MRI ID: identificación de la visita al médico.

- Group: Grupo al que pertenece, que puede ser:
  - Demented: Ha sido diagnosticado con la enfermedad desde el principio.
  - Nondemented: No ha sido diagnosticada la enfermedad en ningún momento.
  - Converted: A 14 pacientes se le diagnosticó mal inicialmente y acabaron teniendo demencia.
- Visit: número de visita.
- MR Delay: Deley desde la última consulta.
- M/F: Género.
- Hand: Si son diestros o no.
- Age: Edad

Y los siguientes valores que son pruebas médicas realizados a los pacientes: SES,MMSE,CDR,eTIV,nWBV,ASF

En el dataset hay varios pacientes, cada uno de ellos han visitado el médico al menos 2 veces. Queremos construir un modelo que prediga el estado del paciente entre Demented, Nondemented y Converted. Con esto podremos identificar a tiempo sobre todo aquello pacientes a los que las pruebas diagnósticas iniciales indicaba que no eran dementes, pero que acabaron siéndolo.

Ejercicio 1 (1 punto): Limpia el dataset de errores y elementos que no sean relevantes. Justifica en una celda de markdown las decisiones que has tomado.

Ejericcio 2 (1 punto): Representa gráficamente los datos una vez hayan sido limpiados.

Ejercicio 3 (2 puntos): Usa vuestro Perceptrón Multicapa de la práctica 5 y 6 para construir el modelo de predicción. Calcula accuracy y la matriz de confusión. El resultado mínimo que debéis conseguir de precisión es de un 65%. El modelo puede tener cualquier capa, pero **debe existir una prueba realizada con más de una capa oculta**, aunque no sea el modelo definitivo que establezcais. **Dejad bien claro cual es el modelo con el que finalmente os quedais.**

Ejercicio 4 (1.5 puntos): Usa el algoritmo Decision Tree de Sklearn para construir un segundo modelo. Calcula accuracy y la matriz de confusión. El resultado mínimo que debéis conseguir de precisión es de un 85% (random_state=42)

Ejercicio 5 (1 punto): Compara escribiendo en una celda en markdown ambos modelos y elige justificadamente cual crees que es el que mejor se adapta al problema que queremos modelar y por qué.

Ejercicio 6 (1.5 puntos): Transforma la clase Converted en Demented y crea un modelo de clasificación binaria con tu perceptrón multicapa con una única neurona de salida.

Forma de entrega:

En el campus virtual con un archivo zip donde esté todo el contenido necesario para ejecutar el jupiter notebook. El zip debe tener vuestro nombre y en la primera línea del notebook también debe aparecer vuestro nombre.

Anexo: API con algunas funciones de Python útiles.

En Pandas: isna() : selecciona campos nulos, combinada con .sum() podeis comprobar el número de campos nulos de un dataframe.

.drop("columna",axis=1) elimina una columna (igual pero con un indice y axis 0 elimina una fila)

dropna(how='any'): borra campos nulos.

to_numpy() convierte un dataframe en un array de numpy.

De Numpy:

np.hstack((A,B)) fusiona dos matrices A y B añadiendo B al final de A. Deben tener un tamaño compatible.

np.zeros_like(np_array) genera un array del mismo tamaño que np_array de ceros.

Teneis además la documentación descargada de MLPClassifier y del DEcission-TreeClassifier que podeis consultar.