Descripción global de las prácticas y criterios de corrección

Aprendizaje Automático

Las prácticas de esta asignatura tienen como objetivo el proveer de la capacidad de desarrollar un sistema de Aprendizaje Automático, para lo cual se resolverá un problema del mundo real escogido por cada equipo de prácticas durante los primeros días de clase. Para hacer esto, se divide el cuatrimestre en dos mitades. La primera mitad tiene como objetivo principal el desarrollo del código que se utilizará para resolver el problema, mientras que se estudia este desde un punto de vista fundamentalmente teórico, y durante la segunda mitad del cuatrimestre se aplicará este código fuente para resolver el problema en distintas aproximaciones, en las que habrá poco desarrollo de nuevo código, pero sí bastante trabajo experimental.

En la primera mitad del cuatrimestre, por lo tanto, se realizarán tres tareas de forma paralela:

- Selección del problema a desarrollar en la asignatura y estudio del mismo. Esto se realizará a partir de la presentación de la asignatura (primera clase de teoría), por lo que desde el punto de vista de las clases de prácticas, esto se realizará a partir de la segunda semana de clase. Esta parte también incluye la recolección de los datos que se utilizarán, bien sean señales o imágenes, y su procesado y extracción de las características correspondientes a la primera aproximación.

- Redacción de los primeros capítulos de la memoria relacionada con este problema a resolver.

- Ejercicios de desarrollo de un sistema de AA. En este aspecto, se plantearán 6 boletines de ejercicios distintos (uno de ellos dividido en dos partes), uno cada semana. Estos ejercicios están pensados para que, al realizarlos, se desarrolle de forma incremental un sistema de AA. El planteamiento de la asignatura es que este sistema se aplique sobre las características extraídas del problema seleccionado. Sin embargo, durante las primeras semanas no se tendrán características extraídas, por lo que, en su lugar, se podrá usar otra base de datos para realizar los ejercicios, como la de flores iris.

Antes de la finalización de la primera mitad del cuatrimestre tendrá lugar el primer TGR, para

el cual se pedirán las primeras secciones de la memoria, así como la parte que se tenga hecha de la primera aproximación (en esta primera entrega no se pedirá la primera aproximación completa).

La redacción de la memoria se realizará de forma incremental, para lo cual se realizarán varias entregas previas a las semanas en las que se realizarán los TGR. Estos TGR tendrán como objetivo el corregir y mejorar la memoria, como documento en el que se plasma el trabajo realizado por cada equipo. Por lo tanto, de forma previa al TGR cada equipo deberá entregar la memoria con los desarrollos solicitados, así como con los cambios que se pidieron en el TGR anterior, orientados a que el equipo aumente la calidad de su trabajo y aprenda a realizar la redacción de un documento técnico. En cada TGR, cada equipo recibirá una nota detallada de cada parte, así como una descripción de lo que les falta para alcanzar la nota máxima en esas partes, en función de los criterios que se especifican más abajo en este documento. De esta forma, en cada TGR no solamente se incluirán nuevas secciones, sino que se podrán modificar las anteriores para mejorar la memoria e intentar alcanzar la nota máxima.

Como se ha dicho, el primer TGR tendrá lugar antes de la finalización de la primera mitad del cuatrimestre, y tiene como objetivo evaluar el estudio teórico del problema, así como mejorarlo y aconsejar a los equipos sobre cómo desarrollar las aproximaciones (motivo por el cual se piden los primeros resultados de la primera aproximación). El resto de TGR tendrán lugar en la segunda mitad del cuatrimestre, puesto esta estará orientada a realizar la parte experimental para resolver el problema seleccionado, así como redactar la memoria.

Por lo tanto, en la segunda mitad del cuatrimestre, una vez realizadas las prácticas anteriores, ya se dispone del código necesario para ejecutar un sistema de Aprendizaje Automático. A partir de este momento, las prácticas de esta asignatura se centran en realizar distintas aproximaciones de cara a resolver el problema escogido, junto con la correspondiente redacción de la memoria. Esta memoria se irá entregando en distintos TGR en la que se corregirá, evaluará y se darán consejos de mejora de cara a alcanzar la máxima calificación.

A finales de cuatrimestre se solicitará una entrega final con todo el material desarrollado, lo cual incluye:

- Código fuente utilizado.

- Base de datos usada.
- Memoria redactada.
- Presentación realizada en clase (si se hace).

Con respecto al código fuente, es importante tener en cuenta que todo el código desarrollado en las prácticas anteriores es de aplicación en la mayoría de las aproximaciones (excepto en la de *Deep Learning*, en la cual se proveerá de un nuevo código fuente). A la hora de desarrollar el código, se debería seguir una estructura similar a esta:

- Crear una carpeta con un nombre similar a *fonts*, donde se sitúe el código de las prácticas anteriores.
- Crear una carpeta con un nombre similar a *datasets*, donde estarán los datos usados en las distintas aproximaciones. Dentro de esta carpeta se pueden crear carpetas nuevas, siendo la forma habitual el crear una carpeta para cada clase. Por ejemplo, si el trabajo es de detección de ojos, esta carpeta tendrá una que se llamará *ojo* y otra que se llamará *no-ojo*. Esta estructura puede cambiar como el equipo de trabajo considere necesario. Por ejemplo, es posible que a lo largo de las distintas aproximaciones se carguen datos distintos, por lo que los nombres de las carpetas podrían modificarse para ajustarse a esto, o incluso crear una jerarquía de carpetas para las distintas aproximaciones.
- En la carpeta raíz, crear un archivo para cada aproximación, de nombres *aprox1.jl*, *aprox2.jl*, etc. Estos archivos deberían de ser muy sencillos, puesto que sólo deberían realizar las siguientes operaciones:
    - Fijar la semilla aleatoria para garantizar la repetibilidad de los resultados.
    - Cargar los datos.
    - Extraer las características de esa aproximación.
    - Llamar a la función *modelCrossValidation* para realizar validación cruzada con distintos modelos y configuraciones de parámetros.
    - Entrenar de nuevo el modelo con la configuración seleccionada para extraer una matriz de confusión.

Una excepción a este esquema está en la aproximación en la que se implemente Deep Learning, como se verá posteriormente.

Una cuestión importante es que al ejecutar estos archivos el sistema debería entrenar los modelos y mostrar por pantalla los mismos resultados y gráficas (si las hay) que aparezcan en la memoria, en las secciones correspondientes a cada aproximación.

Con respecto a la memoria, en otro documento se detalla cómo debería ser su redacción. La memoria constituirá el centro de la evaluación del trabajo, puesto que cada parte se debería ver plasmada en ella. En este trabajo se pedirá un **mínimo de cuatro aproximaciones**. Además, **al menos dos de ellas deberán incluir características nuevas**.

A continuación se especifican los criterios que se utilizarán para corregir las memorias. La puntuación de cada parte de la memoria será la siguiente:

- Introducción: 0.25 puntos. Se valorará la claridad de la explicación.
- Descripción del problema: 0.25 puntos. Se valorará la claridad de la explicación y de los detalles aportados sobre los datos con los que se trabajará. Una explicación que es necesaria en esta parte es la justificación de qué métrica o métricas se utilizarán para valorar y comparar los clasificadores que se generen, y por qué son más adecuadas que el resto de métricas.
- Análisis bibliográfico: 0.25 puntos. Se valorará el número de trabajos descritos, además de las propias descripciones. En este aspecto, para obtener la nota máxima será necesario incluir al menos 8 trabajos y describirlos brevemente. En ocasiones no será posible encontrar referencias directamente relacionadas con la temática a desarrollar, en estos casos se podrán analizar trabajos con temáticas similares. Estos trabajos deberán estar correctamente referenciados siguiendo un estilo concreto, siendo estas referencias de alguna publicación en libros, revistas, actas de congreso o publicaciones similares, pero no de páginas web.
- Desarrollo: 0.8 puntos por aproximación. Para cada una, este punto se distribuirá de la siguiente manera:
    - Descripción: 0.2 puntos. Se valorará la claridad de la descripción, de esta parte, incluyendo conceptos como:

- Descripción razonada de las características, apoyándose en gráficas y/o imágenes explicativas. Si en esta aproximación no se utilizan nuevas características, esto debería estar justificado.

- Descripción de la base de datos utilizada en esta aproximación. A pesar de que en la sección “Descripción del problema” ya se hayan descrito los datos, es posible que en cada aproximación estos varíen ligeramente, por lo que es conveniente tener una descripción incluyendo cuántos patrones se han usado, cuántas entradas, clases, etc.

- Preprocesado de los datos, que generalmente suele ser relativo a la normalización de los mismos. Es necesario justificar el porqué del tipo de normalización utilizado, así como los parámetros de normalización (mínimo, máximo, media, etc.), o por qué no se realiza normalización.

- Otros datos relativos a los experimentos que se van a llevar a cabo, como metodología, número de *folds*, etc.

- Cualquier otro material como gráficas en las que se muestren los datos puede ser de interés para esta parte.

▪ Resultados: 0.4 puntos. Esta sección contiene la parte experimental. En ella, se valora la claridad de las explicaciones, así como el número de experimentos. Es necesario, en todas las aproximaciones excepto en la de *Deep Learning*, realizar experimentación con las 4 técnicas (Redes de Neuronas, SVM, árboles de decisión y kNN), y, para cada una, probar con distintos hiperparámetros.

  - Para el caso de RR.NN.AA., probar al menos 8 arquitecturas distintas, entre una y 2 capas ocultas.

  - Para SVM, probar con distintos kernels y valores de C. Como mínimo, 8 configuraciones de hiperparámetros de SVM.

  - Para árboles de decisión, probar al menos 6 valores de profundidad distintos.

- Para kNN, probar al menos 6 valores de k distintos.

En todos los experimentos realizados, es necesario usar las métricas descritas en la sección 2 de la memoria para evaluar y comparar los modelos obtenidos.

- Discusión: 0.2 puntos. En esta sección se valorará la claridad de la explicación y los razonamientos empleados, indicando el impacto que tienen en el sistema desarrollado, que deberán apoyarse en los resultados obtenidos, así como en otros que se quieran mostrar aquí, como pueden ser matrices de confusión o distintas gráficas explicativas.

- Aproximación final utilizando *Deep Learning*: 0.8 puntos. Para optar a la máxima nota, es necesario seguir una estructura similar a la de una aproximación anterior, que será evaluada siguiendo los mismos criterios (descripción: 0.2 puntos, resultados: 0.4 puntos, discusión: 0.2 puntos), con la diferencia de que la experimentación se realizará únicamente con redes convolucionales. En este aspecto, se deberá experimentar con un mínimo de 10 arquitecturas distintas para optar a la máxima calificación.

- Conclusiones y trabajo futuro: 0.25 puntos. Se valorará la claridad de la explicación, así como que las conclusiones que se extraigan estén apoyadas por los resultados obtenidos.

El desarrollo de nuevas aproximaciones supondrá la posibilidad de sumar hasta medio punto adicional (hasta el máximo de 5 puntos) siempre que su redacción se adecúe a la descrita en este documento.

Para realizar la aproximación basada en *Deep Learning* se proveerá de un código fuente que podréis modificar para adaptarlo a vuestro problema. Sobre esto, tened en cuenta las siguientes observaciones con respecto a realizar esta aproximación utilizando *Deep Learning*:

- En *Deep Learning* se suele trabajar con bases de datos enormes, con lo que hacer validación cruzada es muy lento. Por este motivo, aunque desde el punto de vista teórico habría que hacerla, no es algo habitual. Sin embargo, en este trabajo se utilizan bases de datos pequeñas, por lo que sí se deberá hacer validación cruzada. Además, al igual que en el entrenamiento de redes convencionales, en cada *fold*

habrá que entrenar la RNA varias veces debido al carácter no determinístico del proceso de entrenamiento de las redes. Para ello, se puede sencillamente modificar el código utilizado para que la función que realiza la validación cruzada admita un nuevo tipo de modelo (redes convolucionales), y, de forma similar a las redes convencionales, itere en un bucle en cuyo interior llame a una función que cree y entrene este tipo de redes.

- Igualmente, como las bases de datos en *Deep Learning* suelen ser muy grandes, los patrones de entrenamiento se suelen dividir en subconjuntos denominados *batches*. En vuestro caso, como el número de patrones no va a ser tan grande, esto no es necesario, así que utilizar todos los patrones de entrenamiento en un único *batch*, como se ha venido haciendo en las prácticas hasta el momento.

- Por usar bases de datos tan grandes, en *Deep Learning* no se suele usar conjunto de validación. En lugar de ello, se suelen usar otras técnicas de regularización. En esta aproximación, podéis probar a entrenar las redes sin validación, y si veis que se sobreentrenan, probad a usar validación.

- La RNA que os dejamos en Moodle tiene varias capas convolucionales y *maxpool*. Cada capa convolucional implementa filtros de un tamaño para pasar a un número distinto de canales. Al igual que en aproximaciones anteriores, es interesante hacer pruebas con distintas arquitecturas (un entrenamiento con cada una), para hacer una tabla comparativa que poner en la memoria. En este aspecto, para tener la máxima nota en esta parte se exige realizar pruebas con al menos 8 arquitecturas distintas.

- En redes convolucionales las imágenes de entrada son de tamaño fijo, porque se tiene una neurona de entrada por pixel de la imagen y canal (RGB o escala de grises). Como hemos estado trabajando con ventanas de tamaño variable, lo que se suele hacer es cambiar el tamaño de las ventanas para que todas tengan el mismo tamaño. Esto se puede hacer fácilmente en el código después de cargar la base de datos con la función *imresize*, tenéis más documentación en https://juliaimages.org/latest/pkgs/transformations/

- Si en lugar de usar redes convolucionales para procesado de imagen se usan para procesado de señales, es necesario realizar modificaciones adicionales al código. Por ejemplo, el tamaño de los filtros de convolución empleado en el código es (3, 3), en lugar de ser bidimensional (para imagenes) debería ser de una dimensión (para

señales).

- La base de datos del ejemplo (MNIST) ya tiene un conjunto de parones que debe ser usado para test. En vuestro caso no será así, sino que el conjunto de test será creado mediante validación cruzada.

En alguna ocasión, debido a las propiedades del problema en cuestión (si no se trabaja con señales o imágenes), no será posible realizar la aproximación de *Deep Learning*, En este caso, se puede sustituir por una aproximación adicional en la que se utilicen características nuevas que puntuará de igual manera que el resto (0.8 puntos). En todo caso, consultad con el profesor de prácticas para que verifique esta posibilidad.

En base a estos criterios, la memoria será puntuada en cada TGR, dando la posibilidad de que se aumenten los puntos obtenidos en el TGR anterior. La memoria constituye el documento a ser puntuado puesto que debe ser un reflejo del trabajo realizado. Sin embargo, el código fuente también será analizado para verificar su corrección. Como consecuencia de ello, se podrá tener alguna penalización en la nota final del trabajo si se encuentran errores de concepto graves, como pueden ser:

- Código que no se ejecuta (2 puntos).
- Los resultados no coinciden con los mostrados en la memoria (1 punto).
- Utilizar muestras de test para entrenar los modelos (1 punto).
- etc.