Guía Didáctica de la asignatura: "Aprendizaje Automático I"

Máster en Ingeniería y Ciencia de Datos

Código de asignatura: 31110037

Curso: 2020-2021

José Manuel Cuadra Troncoso
Antonio Rodríguez Anaya
Manuel Luque Gallego
Dpto. de Inteligencia Artificial - E.T.S.I. Informática - UNED

Versión del 9 de octubre de 2020

UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

Guía Didáctica de la asignatura: "Aprendizaje Automático I"

Nota: *En este documento, todas las palabras de género masculino que se apliquen a personas, se deben entender referidas en genérico a ambos sexos indistintamente.*

Índice general

Información general sobre la asignatura

El aprendizaje automático es clave para el desarrollo de sistemas inteligentes y el análisis de datos en ciencia e ingeniería. Tiene un papel decisivo en tecnologías de uso cada vez más común como los reconocedores de voz de los teléfonos móviles o los vehículos autónomos, por ejemplo, así como también sirve para permitir el acceso a la información contenida en nuestro ADN o para dotar de sentido a la avalancha de datos que pueden recogerse en la web. Así, dado que el aprendizaje automático es una de las bases de la ciencia de datos, en este curso se ofrece una introducción a los métodos más comunes que son de uso frecuente en este ámbito. Se tratarán los fundamentos teóricos, así como algoritmos esenciales en el aprendizaje supervisado, no supervisado y una introducción al semi-supervisado.

Información adicional sobre la asignatura está disponible en la guía de la asignatura[1].

Bibliografía

El primer paso a dar para acceder a los libros de esta y otras asignaturas es daros de alta en la colección de libros electrónicos de nuestra biblioteca O'Reilly for Higher Education (New Safari). Para ello hay que autenticarse en la Web de la UNED, hacer click en el enlace anterior y darse de alta con el usuario UNED que se tenga (dirección de correo terminada en @alumno.uned.es). Para futuros accesos, tras el registro, es necesario la autenticación previa en la UNED. En las siguientes listas de libros damos los enlaces para su lectura en O'Reilly o su descarga.

Bibliografía básica

- El libro fundamental de la asignatura es: Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition.
  Al final de cada tema del libro se hallan actividades autoevaluables cuya solución se encuentra en el apéndice A del libro o en archivos del código de ejemplo.
- El método Naive Bayes se verá con Python Data Science Handbook.
- El método K-Nearest-Neighbor se verá con Python Machine Learning.

[1]Esta guía tiene múltiples enlaces a páginas web por ello debe ser leída con una lector de pdfs que permita el seguimiento de enlaces. Los navegadores web permiten el seguimiento pero pueden no mostrar el enlace de forma visible, por lo que se recomiendo usar un programa visualizador.

Bibliografía complementaria

- Aspectos matemáticos se pueden consultar en The Elements of Statistical Learning (Second Edition).
- Otros libros:
  Python Machine Learning Blueprints - Second Edition
  Machine Learning Algorithms - Second Edition
  Hands-On Neural Networks with Keras.

Código de ejemplos

Todos los libros, salvo The Elements of Statistical Learning (Second Edition), tienen disponibles el código fuente de los ejemplos. Algunos tienen un Github para su descarga o son descargables desde la web de Packt Publishing; hay que crearse una cuenta y luego en Support->Code Downloads se busca el título del libro para descargar el código.

El código de Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition también está disponible directamente en los Documentos de Alf en la carpeta code en el archivo handson_ml2_master.zip.

Temario

- Módulo 1 Métodos básicos de aprendizaje automático
  - Tema 1 Introducción al Aprendizaje Automático
    - Introducción
    - Tipos de sistemas de Aprendizaje Automático
    - Principales desafíos del Aprendizaje Automático
    - Testeo y validación
  - Tema 2 Introducción a las Herramientas a utilizar
    - Instalación de las principales herramientas.
    - Introducción a Numpy
    - Introducción a Pandas
    - Introducción a Matplotlib
    - Introducción a Scikit-learn
    - Introducción a la metodología de los proyectos de Aprendizaje Automático
  - Tema 3 Métodos simples de Aprendizaje Automático Supervisado
    - Naive Bayes
    - k-Nearest-Neighbors

Plan de trabajo

- Módulo 1 Métodos básicos de aprendizaje automático
  - Duración: Semanas 1 a 5, con el siguiente desglose:
    - Temas 1 y 2: Semanas 1 y 2 (del 5 de octubre de 2020 al 18 de octubre de 2020).
    - Tema 3: Semanas 3 y parte de la 4 (del 19 de octubre de 2020 al 28 de octubre de 2020).
    - Tema 4: Semanas parte de la 4 y la 5 completa (del 29 de octubre de 2020 al 8 de noviembre de 2020).
  - Actividades a realizar (a través del curso virtual):
    - Estudio de los materiales proporcionados por el Equipo Docente a través del curso virtual. 30 horas.
    - Realización de cuestionarios de autoevaluación publicados en el curso virtual. 4 horas.
    - Consulta de foros. 4 horas.
    - Realización del trabajo práctico de evaluación de este módulo * **. 20 horas. Semanas 3 a 6.
    - Realización de la PEC de este módulo * **. 2 horas. Semana 6.
- Módulo 2 Redes neuronales
  - Duración: Semanas 7 a 9, con el siguiente desglose:
    - Tema 1: Semanas 6 y parte de la 7 (del 9 de noviembre de 2020 al 18 de noviembre de 2020).
    - Tema 2: Semanas parte de la 7 y la 8 completa (del 19 de noviembre de 2020 al 29 de noviembre de 2020).
  - Actividades a realizar (a través del curso virtual):
    - Estudio de los materiales proporcionados por el Equipo Docente a través del curso virtual. 17 horas.
    - Realización de cuestionarios de autoevaluación publicados en el curso virtual. 2 horas.
    - Consulta de foros. 2 horas.
    - Realización del trabajo práctico de evaluación de este módulo * **. 10 horas. Semanas 8 y 9.
    - Realización de la PEC de este módulo * **. 2 horas. Semana 9.
- Módulo 3 Máquinas de vectores soporte
  - Duración: Semanas 9 a 11, con el siguiente desglose:

- Tema 7: Semanas 9 a 11 (del 20 de noviembre de 2020 al 19 de diciembre de 2020).

- Actividades a realizar (a través del curso virtual):
  - Estudio de los materiales proporcionados por el Equipo Docente a través del curso virtual. 17 horas.
  - Realización de cuestionarios de autoevaluación publicados en el curso virtual. 2 horas.
  - Consulta de foros. 2 horas.
  - Realización del trabajo práctico de evaluación de este módulo * **. 10 horas. Semanas 11 y 12.
  - Realización de la PEC de este módulo * **. 2 horas. Semana 12.

- Módulo 4 Selección de variables y reducción de la dimensionalidad
  - Duración: Semanas 12 y 13, con el siguiente desglose:
    - Tema 8: Semanas 12 y 13 (del 20 de diciembre de 2020 al 17 de enero de 2021).
  - Actividades a realizar (a través del curso virtual):
    - Estudio de los materiales proporcionados por el Equipo Docente a través del curso virtual. 10 horas.
    - Realización de cuestionarios de autoevaluación publicados en el curso virtual. 2 horas.
    - Consulta de foros. 2 horas.
    - Realización del trabajo práctico de evaluación de este módulo * **. 7 horas. Semanas 12 y 13.
    - Realización de la PEC de este módulo * **. 1 horas. Semana 13.

* Las fechas de realización de la PEC y el trabajo práctico se publicarán en el curso virtual con suficiente antelación. Es necesario alcanzar una puntuación mínima en estas pruebas de evaluación para poder aprobar la asignatura (los detalles se encuentran en el apartado de evaluación de esta guía).

** La calificación de la PEC y del trabajo práctico se guarda para la convocatoria extraordinaria. Además, para el trabajo práctico, se abrirá otro periodo de entrega para dicha convocatoria.

Tema 1

Introducción al Aprendizaje Automático

1.1. Material de estudio

Los materiales de estudio de este tema son: capítulo 1 de Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition completo. En él se da una visión de lo que es el Aprendizaje Automático, se comentan su tipos y sus principales desafíos y técnicas. No es un mero tema introductorio, muchas de las las cuestiones que se comentan se aplicarán en muchas de las actividades con código del curso.

1.2. Índice

- Introducción
- Tipos de sistemas de Aprendizaje Automático
- Principales desafíos del Aprendizaje Automático
- Testeo y validación

1.3. Actividades autoevaluables

Realizar los ejercicios que se encuentran en la sección Exercices al final del capítulo 1 del libro, en el apéndice A del libro están las soluciones.

1.4. Actividades evaluables

Este tema no tiene actividades evaluables.

Tema 2

Introducción a las Herramientas a utilizar

2.1. Material de estudio

Los materiales de estudio de este tema son: capítulo 2 y el capítulo 3 de Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition completos y adquirir un conocimiento inicial de la mayoría de las herramientas a usar durante el curso. Iremos detallando el uso de estos materiales en las secciones siguientes.

Sobre las ecuaciones del libro

Para la correcta visualización de las ecuaciones del libro en en Chrome es necesario instalar la extensión Math Anywhere que le proporciona el soporte para MathML. En otros navegadores como FireFox, Safari y Opera este soporte ya está incluido. Para Internet Explorer, dependiendo de la versión, existe un plugin llamado MathPlayer.

2.2. Índice

- Instalación de las principales herramientas.
- Introducción a Numpy
- Introducción a pandas
- Introducción a Matplotlib
- Introducción a scikit-learn
- Introducción a la metodología de los proyectos de Aprendizaje Automático

2.3. Instalación de las principales herramientas

Lo primero va a ser la instalación de casi todas las herramientas que se van a usar en la asignatura, en el tema 6 instalaremos las específicas de redes neuronales (Tensorflow y Keras).

En el libro se describe la instalación usando entornos virtuales en los sistemas operativo (SO) Linux y Mac, en Linux incluso se puede instalar usando paquetes de sistema y la herramienta de instalación de Python pip. Sin embargo vamos a recomendar la instalación en cualquier SO usando Anaconda Distribution, la versión open source de Anaconda, con Python 3.7. Desde ese enlace se elige el SO la versión de Python (3.7) y nos descargamos el instalador gráfico. Tras lanzarlo se aceptan los términos de licencia y se establece el directorio de instalación, se recomienda usar el que viene por defecto, Anaconda instala por defecto la versión elegida de Python y todos los paquetes que usaremos en la asignatura, salvo los de tema 6. Para más información consultad la documentación de Anaconda Distribution, en especial la guía de usuario de Anaconda Navigator.

Una interesante introducción a Python, o para consultar su programación, se puede ver en Introducing Python, 2nd Edition, con código de ejemplos descargable.

Anaconda también se puede usar para instalar paquetes de R e instala RStudio por defecto.

2.4. Introducción a las herramientas

A lo largo del capítulo 2 del libro encontraremos ejemplos del uso de las herramientas usando como entorno de programación Jupyter Notebook, que Anaconda instala por defecto y que será el que utilicemos durante el curso. En el primer tema se comentó que los dos usos principales del aprendizaje supervisado son la clasificación y la regresión. En este capítulo se da un ejemplo del uso de las herramientas para un estudio de la predicción de precios de las viviendas en California usando regresión. También el capítulo 3 del libro da ejemplos del uso de las herramientas en este caso dedicados a la clasificación de imágenes de dígitos escritos a mano.

Dos de las herramientas, Numpy y Pandas, son objeto de estudio de la asignatura Programación en Entornos de Datos por lo que aquí solo se pretende dar una pequeña introducción suficiente hasta que en dicha asignatura se estudien con profundidad.

2.4.1. Numpy

Numpy es un paquete de computación científica para Python, en esta asignatura lo usaremos básicamente para el manejo de arrays. Como introducción se recomienda leer el Quickstart tutorial.

2.4.2. pandas

pandas es una biblioteca para análisis de datos en Python. En esta asignatura lo usaremos básicamente para la lectura de archivos CSV a través de los DataFrame. Como introducción se recomienda leer las secciones Package overview y 10 minutes to pandas del Getting started.

2.4.3. Matplotlib

Matplotlib es una biblioteca para la generación de gráficos en Python. Como introducción se recomienda leer los tutoriales Usage Guide, Intro to Pyplot y enSample plots in Matplotlib se tienen ejemplos de uso, haciendo click en los gráficos se accede al código que los genera.

2.4.4. scikit-learn

scikit-learn, o abreviadmente sklearn, es la herramienta fundamental de esta asignatura. Es una de las principales bibliotecas para Aprendizaje Automático en Python. La iremos estudiando en casi todos los temas de esta asignatura y una introducción interesante se da en los capítulos 2 y 3 del libro. La descripción de todos los algoritmos implementados en scikit-learn se puede ver en su User Guide, cada descripción contiene un amplio comentario sobre las características del algoritmo, su formulación matemática, enlaces a la API de scikit-learn y está acompañada de ejemplos, de uso cuyo código se puede descargar de la página correspondiente como scripts de Python o como notebooks de Jupyter.

2.5. Introducción a la metodología de los proyectos de Aprendizaje Automático

En esta sección nos referimos a la lectura de los capítulo 2 y 3 del libro completos. Como comenta en el libro para la lectura de estos capítulos no hace falta conocer los algoritmos que se emplean solo familiarizarse con el uso de las técnicas contenidas en scikit-learn y el resto de herramientas y a la metodología a emplear en los proyectos de Aprendizaje Automático. De todas formas recomendamos una lectura rápida en la User Guide de las primeras secciones correspondientes a cada algoritmo usado en los dos capítulos. Las APIs de scikit-learn de los distintos algoritmos tiene muchas características de uso en común, lo que facilita su aprendizaje y uso, como puede verse en los ejemplos del libro. Esta ha sido una de las razones, junto con su amplia disponibilidad de algoritmos y excelente documentación, para usar esta herramienta.

En estos dos capítulos se comentan técnicas para:

- La obtención de datos.
- La visualización de los datos para obtener información.
- La preparación de los datos para aplicar algoritmos y obtener mejores resultados.
- La selección y entrenamiento de modelos para tareas de regresión y clasificación y su afinación.
- El uso de medidas de rendimiento.
- La utilización práctica de los modelos obtenidos.

El espacio de aplicación de estas técnicas va más allá de su uso en esta asignatura, se utilizan en proyectos relacionados con el resto de asignaturas del máster del ámbito de la Ciencia de los Datos: Modelado Estadístico de Datos, Aprendizaje Automático II, Deep Learning, Minería de Textos, Modelos Bayesianos Jerárquicos y Minería de Datos de los Medios Sociales, así como en el TFM si se realiza en este ámbito.

2.6. Actividades autoevaluables

Realizar los ejercicios que se encuentran en la sección Exercices al final de los capítulos 2 y 3 del libro, en el código del libro disponible en Alf se hallan las soluciones.

2.7. Actividades evaluables

Este tema no tiene actividades evaluables.

Tema 3

Métodos simples de Aprendizaje Automático Supervisado

3.1. Material de estudio

Los materiales de estudio de este tema son: la sección In Depth: Naive Bayes Classification del capítulo 5 de Python Data Science Handbook, el capítulo 9 de Python Machine Learning y el capítulo 6 de Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition completos y adquirir un conocimiento de las herramientas que en ellos se comenta principalmente a partir de la documentación de scikit-learn. Iremos detallando el uso de estos materiales en las secciones siguientes.

Para repasar conceptos de probabilidad, no hay que conocerlos en profundidad solo tener una idea de ellos, recomendamos: Introduction to Probability o consultar las páginas correspondientes de Wikipedia especialmente en inglés, la versión en español suele ser un resumen.

Los códigos de ejemplo nuevos en este tema son PythonDataScienceHandbook_master.zip y Python Machine Learning Source Code.zip.

3.2. Índice

- Naive Bayes.
- K-Nearest-Neighbors
- Árboles de decisión

3.3. Naive Bayes

Naive Bayes (Bayes ingenuo) son un conjunto de técnicas de aprendizaje supervisado basadas en la aplicación del teorema de Bayes asumiendo que las características son independientes entre sí dada una clase, o sea, no se tienen en cuenta correlaciones entre variables. En la sección In Depth: Naive Bayes Classification del capítulo 5 de Python Data Science Handbook, se describe

esta idea, su base matemática y se comentan y dan ejemplos de dos clasificadores de este tipo: Gaussian Naive Bayes, Multinomial Naive Bayes.

En la sección de Gaussian Naive Bayes el notebook 05.05-Naive-Bayes.ipynb de PythonDataScienceHandbook_master contiene el ejemplo y hace uso del generador de muestras aleatorias make_blobs de scikit-learn, estos generadores permiten construir datos artificiales para pruebas controladas.

En la sección de Multinomial Naive Bayes se comenta un ejemplo de clasificación de textos, está en el notebook comentado anteriormente, los textos deben ser convertidos en números para que este y cualquier otro clasificador pueda trabajar con ellos. En el texto hay un enlace a la sección Feature Engineering donde se da una introducción a la vectorización de textos, estos temas se tratarán con mayor profundidad en la asignatura Minería de textos, aquí solo se trata de saber que existen esos procedimientos y su uso simple.

Completaremos el estudio de este tema con la lectura y estudio de la documentación scikit-learn sobre Naive Bayes, en ella se comentan otros dos procedimientos: Complement Naive Bayes y Bernoulli Naive Bayes. La sección Out-of-core naive Bayes model fitting no constituye material de estudio. En el enlace a la documentación de Naive Bayes se incluyen enlaces a las APIs de los 4 procedimientos motivo de estudio que es muy conveniente conocer para su aplicación práctica. Los ejemplos incluidos al final de las secciones de las APIs son de lectura opcional, el ejemplo corto situado antes de estas secciones da una visión rápida de cómo emplear las distintas herramientas.

3.4. K-Nearest-Neighbors (KNN)

Vamos a ver el uso de KNN para tareas de clasificación y regresión. El uso de KNN para aprendizaje no supervisado no es materia de este tema.

3.4.1. Clasificación

Como su nombre indica este algoritmo clasifica una instancia según la clase predominante entre sus K vecinos más cercanos. En el capítulo 9 de Python Machine Learning se comenta este algoritmo. Básicamente el capítulo tiene dos secciones: Implementing Knn in Python y Using scikit-learn'S Kneighborsclassifier Class for Knn. De la primera no es necesario conocer cómo se implementa el código, sino tener la idea de qué está haciendo es código, ya que es la descripción del algoritmo. De la segunda sí hay que adquirir un conocimiento detallado ya que explica en profundidad cómo aplicar KNN y encontrar el valor óptimo de K. Para ello usa un dataset de juguete de scikit-learn dataset de juguete de scikit-learn, load_iris, si antes de la carga del dataset `iris = datasets.load_iris()` se añade esta línea `from IPython.display import Markdown` y tras la carga esta otra `display(Markdown(iris.DESCR))`, se mostrará la descripción del dataset, si se usara directamente `print(iris.DESCR)` la salida sería un tanto desordenada. Al ver la descripción nos daremos cuenta, entre otras cosas, de porqué habla de sépalos en el notebook y cuáles son los tipos de iris.

El código del ejemplo se halla en el notebook Chapter 9 - Supervised Learning—Classification Using K-Nearest Neighbors (KNN).ipynb de Python Machine Learning Source Code. En el se usa el paquete seaborn de visualización estadística de datos que está basado en Matplotlib, no será materia de estudio de la asignatura pero se puede usarlo si se desea, viene instalado por defecto en Anaconda. Una nota sobre el cálculo del step de numpy.meshgrid, en el notebook se calcula así `h = (x_max / x_min)/100`, si x_min fuera 0 habría un error, creemos que quisieron poner `h = (x_max - x_min)/100`.

Para completar el estudio de KNN para clasiificación consultar de la documentación de scikit-learn Nearest Neighbors la introducción y la sección 1.6.2 y los enlaces a las APIs contenidos en 1.6.2: KNeighborsClassifier y RadiusNeighborsClassifier. El ejmplo al final de la sección 1.6.2 no es necesario verlo ya que clasifica iris de la misma manera que el ejemplo del libro pero con menos profundidad.

3.4.2. Regresión

En el libro omite el estudio de la regresión usando KNN. Para estudiarlo haremos uso de la sección 1.6.3 de la documentación de scikit-learn Nearest Neighbors y los enlaces a las APIs: KNeighborsRegressor y RadiusNeighborsRegressor. Como ejemplo basta con estudiar el Nearest Neighbors regression de la sección Examples de 1.6.3.

3.4.3. Notas

Aunque el libro no lo comenta dependiendo del problema puede ser necesario escalar las variables para una correcta aplicación del KNN, para los otros procedimientos comentados en este tema no suele ser necesario. Ver en el tema 2 el ejemplo sobre clasificación en que se empleaba el escalamiento.

3.5. Árboles de decisión

Los árboles de decisión (decision trees) también permiten la clasificación y la regresión, se basan en grafos con estructura de árboles. Los árboles de decisión son modelos del tipo denominado de "caja blanca", son intuitivos y fáciles de interpretar y entran dentro de la denominada Inteligencia Artificial explicable, por contra los resultados obtenidos de un modelo de "caja negra" hay que admitirlos per se, no son explicables. Estas técnicas tanto para clasificación como para regresión están explicadas en el capítulo 6 de Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition y su correspondiente notebook 06_decision_trees.ipynb de handson-ml2-master.

3.5.1. Clasificación

La clasificación usando árboles de decisión está explicada en las siguientes secciones del libro.

En Training and Visualizing a Decision Tree se comenta cómo visualizar los árboles usando los datos de flores iris ya comentados en KNN pero usando las medidas de los pétalos en lugar de la de los sépalos. En Making Predictions se comenta cómo el procedimiento realiza predicciones. El algoritmo es capaz de proporcionar las probabilidades de pertenencia a clases lo que se explica en la sección Estimating Class Probabilities. El algoritmo en el que se basa el procedimiento se comenta en The CART Training Algorithm, Computational Complexity y Gini Impurity or Entropy?. Para terminar cuestiones de selección de parámetros para mejora del entrenamiento se ven en Regularization Hyperparameters. En esta última sección se usa otro generador de muestras aleatorias make_moons. En la sección Instability se comenta limitaciones del procedimiento.

El código del ejemplo se halla en el notebook 06_decision_trees.ipynb de handson-ml2-master. Para el correcto funcionamiento de notebook es necesario instalar el paquete python-graphviz desde Anaconda Navigator. Si se quiere hace uso del comando dot para exportar los árboles es necesario instalar pydot.

La documentación de scikit-learn Decision Trees da una información muy similar a la del libro con la diferencia de que el árbols se muestra con más niveles, solo sería necesario consultar la información de la API de DecisionTreeClassifier y la sección 1.10.5 por sus aspectos prácticos. La sección 1.10.3 Multi-output problems queda fuera de los objetivos de la asignatura, se puede leer como ampliación.

3.5.2. Regresión

La regresión usando árboles de decisión está explicada en las siguientes sección del libro titulada Regression. La documentación de scikit-learn Decision Trees da una información muy similar a la del libro, solo sería necesario consultar la información de la API de DecisionTreeRegressor y la sección 1.10.5 por sus aspectos prácticos.

3.6. Actividades autoevaluables

3.6.1. Naive Bayes

1. ¿Qué parámetros son aprendidos durante el entrenamiento de Naive Bayes?
2. ¿Qué diferencia a cada uno de los procedimientos estudiados?
3. ¿Cuál es uno de los usos más frecuentes de Multinomial Naive Bayes y porqué?
4. Ventajas y desventajas

3.6.2. KNN

1. ¿Qué parámetros son aprendidos durante el entrenamiento de KNN?
2. Describir el procedimiento KNN como se implementa en Implementing Knn in Python.

3. ¿Cómo se deshacen empates? Realizar una búsqueda por Internet.

4. ¿Es la distancia euclídea la única posible en scikit-learn para ordenar los vecinos?

3.6.3. Árboles de decisión

Realizar los ejercicios que se encuentran en la sección Exercices al final del capítulo 6 de Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition. El último ejercicio constituye una implementación del procedimiento Random Forest que se ve con más detalle en la asignatura Aprendizaje Automático 2.

3.7. Actividades evaluables

La actividad evaluable asociada a este tema y al módulo 1 de la asignatura va a constituir en un ejercicio de clasificación empleando los tres métodos descritos en el tema y la aplicación de varias de las técnicas estudiadas en el tema 2. Para ello habrá que crear un notebook con celdas de tipo code y de tipo markdwon, en las de este último tipo se incluirán comentarios de lo que se esté haciendo en las de tipo code o sobre los resultados.

La fecha de entrega de la actividad aparece en la plataforma.

3.7.1. Datos

Los datos provienen de la web InsideAirBnB, dedicada al estudio de los alquileres vacacionales ofrecidos en la plataforma AirBnB. Es necesario descargar el fichero airbnb.csv de la carpeta datos de Alf. Este fichero es una versión editada, a fin de facilitar la tarea, del listado original de información sobre las ofertas existentes, para la ciudad de Madrid, en abril de 2017. Contiene 13321 registros con 11 campos cada uno, correspondientes a diferentes características de cada oferta de alojamiento.

El fichero se puede descargar a mano y colocarlo en un directorio desde donde lo cargue el notebook.

La tarea de clasificación consistirá en clasificar los datos según el tipo de alojamiento, definido en el campo room_type, a partir del resto de características. Es decir, en room_type estarán codificadas las clases y en el resto de campos los atributos.

3.7.2. Estudio estadístico y limpieza de datos

Se realizará un breve estudio estadístico de los datos numéricos y de la variable room_type se contarán los valores de cada clase. Si las clases no estuvieran balanceadas habrá que usar los mecanismos que puedan tener los algoritmos de clasificación para tratar con este caso. Para Naive Bayes ya se habla de esto en la actividades autoevaluables, para los otros métodos consultar en la documentación de sus APIs los parámetros de los constructores para ver si hay mecanismos para balancear.

En cuanto a la limpieza de datos estudiar si hay datos faltantes, transformar datos categóricos y escalar datos numéricos haciendo uso de pipelines cuando sea posible.

3.7.3. Entrenamiento y validación

1. Dividir los datos en conjunto de entrenamiento y test de manera que el conjunto de test sea un 20 % del total.

2. Evaluar los modelos midiendo la exactitud (accuracy) usando validación cruzada para los tres métodos estudiados utilizando los parámetros por defecto de los tres métodos, salvo lo comentado para el balance, para cross_val_score usar cv=10. Comparar los resultados de los tres modelos.

3. Realizar la afinación de hiperparámetros para KNN y Árboles de decisión empleando GridSearchCV, Naive Bayes no tiene hiperparámetros que afinar.

   a) Para KNN buscando el valor óptimo de K.

   b) Para Árboles de decisión: variando max_leaf_nodes entre 2 y 50 ambos inclusive[1], min_samples_split entre 2 y 6 ambos inclusive y max_depth entre 1 y 20 ambos inclusive. Ver la documentación de la API.

3.7.4. Valoración

Se valorará la creación del código que realice los requisitos enumerados, la presencia de comentarios de lo que hace el código y sus resultados y las comparaciones entre los resultados obtenidos por los distintos algoritmos, así como algún gráfico que muestre los resultados. Para terminar incluir un apartado de conclusiones. Tanto los comentarios como las conclusiones no es necesario que sea extensos, sino que describan de forma concisa.

3.8. Soluciones actividades autoevaluables

3.8.1. Naive Bayes

1. Usando la notación de $P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$, los parámetros aprender son: las probabilidades de cada clase/etiqueta $P(y)$ y las condicionadas de cada característica por cada clase $P(x_i \mid y)$.

2. La principal diferencia entre los 4 procedimientos estudiados está en la distribución que se supone para las probabilidades condicionadas de cada característica por cada clase $P(x_i \mid y)$. En Gaussian Naive Bayes la distribución se supone gaussiana. En Multinomial Naive Bayes es multinomial. En Complement Naive Bayes la distribución es también multinomial

[1] range(a, b) incluye a pero llega hasta b-1

pero las muestras no están balanceadas, no tienen aproximadamente el mismo número de elementos. En Bernoulli Naive Bayes la distribución es la de Bernoulli.

3. Multinomial Naive Bayes se usa frecuentemente en la clasificación de textos. La distribución multinomial es apropiado para características que representan recuentos o tasas de recuento y dado que la vectorización de textos está relacionada con el recuento de palabras o su frecuencia de aparición, Multinomial Naive Bayes se adecua a la clasificación de textos.

4. Ventajas y desventajas:

   a) Ventajas

      1) Es extremadamente rápido tanto para el entrenamiento como para la predicción.
      2) Proporciona predicciones probabilísticas sencillas.
      3) A menudo es muy fácil de interpretar.
      4) Tiene muy pocos (si alguno) parámetros ajustables.

   b) Desventajas

      1) La suposición sobre los datos es muy estricta y esto hace que pueda ser peor que otros clasificadores más complejos.

3.8.2. KNN

1. En KNN no hay parámetros que se aprendan durante el entrenamiento, el método simplemente clasifica una determinada instancia en función de qué datos etiquetados están más cerca de la instancia.

2. La clasificación de los elementos del conjunto de test se realiza así:

   a) Se calcula para cada dato del conjunto de test la distancia a cada dato del conjunto de entrenamiento.

   b) Se ordenan los datos del conjunto de entrenamiento en función de las distancias calculadas de menor a mayor.

   c) Se eligen los K primeros, o sea, los K vecinos más cercanos.

   d) Se asocia al elemento del conjunto de test la clase predominante en los K vecinos más cercanos.

3. Si solo hay dos clases se elige K impar. Si no siempre podría darse el caso de empates, por ejemplo si el número de clases es 3 y K es 5 podría haber 2 vecinos de una clase, 2 de otra y 1 de la restante. En este caso conviene tener una regla heurística: seleccionar la clase que contenga al vecino más próximo, seleccionar la clase con distancia media menor, etc.

4. El parámetro metric de KNeighborsClassifier permite pasar un objeto de clase DistanceMetric que permite es uso de diferentes métricas como la euclídea, la Manhattan, la Chebyshev, etc. Estas métricas también pueden depender del tipo de datos: reales, enteros, booleanos, etc.

3.8.3. Árboles de decisión

En el apéndice A de Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition y en su correspondiente notebook, 06_decision_trees.ipynb de handson-ml2-master están las soluciones.

Tema 4

Métodos simples de Aprendizaje Automático No-Supervisado y Semi-supervisado

4.1. Material de estudio

Los materiales de estudio de este tema son: el capítulo 9 de Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition, la sección In-Depth: Kernel Density Estimation del capítulo 5 de Python Data Science Handbook y adquirir un conocimiento de las herramientas que en ellos se comenta principalmente a partir de la documentación de scikit-learn. Iremos detallando el uso de estos materiales en las secciones siguientes.

Los códigos de ejemplo para este tema ya han sido descargados en temas anteriores.

4.2. Índice

En este tema vamos a dar un introducción al aprendizaje no-supervisado este tema se complementa con el tema 8 que tratará de la reducción de la dimensionalidad. Además en la asignatura Aprendizaje Automático 2 se verán otras cuestiones relacionadas con el aprendizaje no-supervisado. El aprendizaje no-supervisado se ocupa de crear modelos a partir de datos no etiquetados, no hay conocimiento a priori.

Además daremos una breve introducción al aprendizaje semi-supervisado que se ocupa de crear modelos cuando se dispone de (unos pocos) datos etiquetados y (bastantes más) datos no etiquetados.

- Técnicas de agrupamiento: K-means y DBSCAN.
- Modelos de mezclas gaussianas (GMM)
- Estimación de densidad mediante núcleos (kernel) (KDE)
- Propagación y extensión de etiquetas

4.3. Técnicas de agrupamiento

Las técnicas agrupamiento (clustering en inglés) se utilizan para crear grupos de objetos, no etiquetados, similares. La similitud puede venir dada por la distancia entre los objetos, densidad de áreas, distribuciones estadísticas, etc. En este tema veremos dos de las técnicas no probabilísticas más usadas K-Means y DBSCAN. Estas técnicas se tratan en la sección Clustering del capítulo 9 de Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, la sección comenta una serie de aplicaciones de estas técnicas. Como complemento en la documentación de scikit-learn Overview of clustering methods se da un listado de métodos, algunos de ellos se describen al final de la sección Clustering del libro Other Clustering Algorithms. Los aspectos más prácticos de las dos técnicas se pueden ver en el notebook 09_unsupervised_learning.ipynb de handson-ml2-master.

4.3.1. K-Means

La sección Clustering del libro tiene varias subsecciones dedicadas a K-Means en las que se da una introducción, se comenta el funcionamiento algoritmo, los métodos de inicialización, métodos para acelerar la ejecución, cómo encontrar el número óptimo de grupos y los límites de la técnica. En el aspecto práctico ofrece un ejemplo del uso para segmentación de imágenes. El resto de la sección: Using Clustering for Preprocessing y Using Clustering for Semi-Supervised Learning es lectura complementaria, no motivo de examen.

Para complementar el estudio de esta técnica consultar la documentación de scikit-learn 2.3.2. K-means. En el enlace se incluyen enlaces a las APIs de los 2 procedimientos (KMeans y MiniBatchKMeans) motivo de estudio que es muy conveniente conocer para su aplicación práctica. Los ejemplos incluidos al final de las secciones de las APIs son de lectura opcional, salvo Demonstration of k-means assumptions que da una idea rápida del empleo del método y alterar los datos proporcionados por make_blobs.

4.3.2. DBSCAN

La sección Clustering del libro tiene una subsección dedicadas a DBSCAN en la que se da una introducción al algoritmo. En el aspecto práctico ofrece un ejemplo del uso para agrupación de lunas creadas con make_moons.

Para complementar el estudio de esta técnica consultar la documentación de scikit-learn 2.3.7. DBSCAN. En el enlace se incluyen enlaces a la API de DBSCAN que es muy conveniente conocer para su aplicación práctica.

4.4. Modelos de mezclas gaussianas (GMM)

Los GMM son modelos probabilísticos en los que se supone que las instancias son generadas a partir de mezclas de distribuciones normales. Veremos su uso para agrupamiento, generación

de datos y detección de anomalías y novedad. La primera parte de la secciónGaussian Mixtures del capítulo 9 de Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow comenta la teoría, su uso para agrupamiento y la generación de nuevos datos. Este método difiere de los anteriores en que es un método probabilístico y por ello puede dar las probabilidades de pertenencia a los grupos. A partir de la subsección Anomaly Detection Using Gaussian Mixtures se comenta su uso para la detección de outliers. La detección de outliers se suele usar en las etapas de análisis exploratorio de los datos para eliminar datos anómalos que puedan distorsionar los resultados de las técnicas de aprendizaje automático. Las subsecciones siguientes: Selecting the Number of Clusters y Bayesian Gaussian Mixture Models comentan técnicas para hallar el número óptimo de grupos usando los criterios de información BIC y AIC. La sección termina comentando otros algoritmos para la detección de anomalías y novedad, su estudio no es necesario, algunos de ellos serán estudiados en temas posteriores.

Para complementar el estudio de esta técnica consultar la documentación de scikit-learn 2.1. Gaussian mixture models. De esta sección es importante conocer los pros y contras de los dos métodos comentados: GaussianMixture y BayesianGaussianMixture, los aspectos teóricos de la inferencia variacional y el proceso de Dirichlet son lectura complementaria y no serán motivo de estudio. En el enlace se incluyen enlaces a las APIs de los dos métodos comentados que es muy conveniente conocer para su aplicación práctica.

Los aspectos más prácticos de GMM se pueden ver en el notebook 09_unsupervised_learning.ipynb de handson-ml2-master.

4.5. Estimación de densidad mediante núcleos (KDE)

La estimación de densidad mediante núcleos, en inglés kernel density estimation, es una técnica de estimación de funciones de densidad de probabilidad, GMM es otra. Veremos el KDE en las primeras partes de la sección In-Depth: Kernel Density Estimation del capítulo 5 de Python Data Science Handbook. En ellas se comenta la motivación del KDE, los usos prácticos y la forma de selección del parámetro bandwith, las subsecciones Example: KDE on a Sphere y Example: Not-So-Naive Bayes son lectura complementaria y no serán motivo de estudio.

Para complementar el estudio de esta técnica consultar la documentación de scikit-learn 2.8. Density Estimation el contenido es prácticamente el mismo que el del libro, aunque da una visión un poco más matemática de los que son los kernels que es conveniente conocer. De los ejemplos dos son los mismos que en el libro y en el titulado Kernel Density Estimation se ve cómo usar KDE para generar nuevas muestras, este ejemplo es también lectura complementaria.

Los aspectos más prácticos de KDE se pueden ver en el notebook 05.13-Kernel-Density-Estimation.ipynb de PythonDataScienceHandbook_master.

4.6. Propagación y extensión de etiquetas

La propagación y extensión de etiquetas será la única toma de contacto que tendremos con el aprendizaje semi-supervisado en esta asignatura, y quizás en el máster, debido a que ha sido mucho menos usado que las versiones supervisada y no-supervisada. El aprendizaje semi-supervisado puede ser usado para clasificación y regresión.

Para su estudio usaremos la documentación de scikit-learn 1.14. Semi-Supervised en ella se comentan las técnicas LabelPropagation y LabelSpreading, consultar los enlaces a sus APIs. De los ejemplos que incluye esta documentación bastará con estudiar el primero para ver una aplicación sencilla del aprendizaje semi-supervisado, el resto de ejemplo es lectura complementaria. Al principio de la documentación hay un enlace a Wikipedia en el que se puede encontrar un artículo sobre el aprendizaje semi-supervisado.

4.7. Actividades autoevaluables

4.7.1. Técnicas de agrupamiento y GMM

Realizar los ejercicios que se encuentran en la sección Exercices al final del capítulo 9 de Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow hasta el ejercicio 11 inclusive, salvo el 6.

4.7.2. KDE

1. ¿Qué elecciones pueden llevar a histogramas con distinta apariencia usando los mismos datos y cómo se puede evitar el problema?

2. ¿Qué es un kernel, qué parámetros tiene y cuál es su efecto?

3. ¿Cuáles son los tipos de kernel disponibles en scikit-learn?

4. Describir un procedimiento para buscar un valor apropiado para el parámetro bandwith.

4.7.3. Propagación y extensión de etiquetas

1. ¿En qué consiste el aprendizaje semi-supervisado?

2. ¿Cuáles son los modelos disponibles en scikit-learn para el aprendizaje semi-supervisado y cómo funcionan?

4.8. Actividades evaluables

Este tema no tiene actividades evaluables.

4.9. Soluciones actividades autoevaluables

4.9.1. Técnicas de agrupamiento y GMM

En el apéndice A de Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow y en su correspondiente notebook, 06_decision_trees.ipynb de handson-ml2-master están las soluciones.

4.9.2. KDE

1. El tamaño y posición de los intervalos (bins). Un histograma se puede ver como pilas de bloques, uno por cada punto. Si en lugar de apilar los bloques alineados con los intervalos se alinean con los puntos que representan y se suman las alturas en de cada posición se consigue una mejor representación de los datos.

2. Un kernel es una función positiva controlada por el parámetro bandwith (ancho de banda) que especifica la forma de la distribución que se usa para suavizar la función de densidad de probabilidad que se obtiene a partir de los datos. El parámetro controla la compensación de sesgo-varianza en la estimación de la función de densidad: un ancho de banda demasiado estrecho conduce a una estimación de alta varianza (es decir, overfitting), donde la presencia o ausencia de un solo punto crea una gran diferencia. Un ancho de banda demasiado amplio conduce a una estimación de alto sesgo (es decir, underfitting) donde el núcleo ancho elimina la estructura de los datos.

3. gaussian, tophat, epanechnikov, exponential, linear, cosine.

4. Utilizando el procedimiento GridSearchCV al que se le manda el tipo de kernel, un rango de valores para el parámetro y utilizando un procedimiento de validación cruzada de manera que se obtenga el valor del parámetro que de la mejor puntuación (parámetro scoring) para la métrica elegida.

4.9.3. Propagación y extensión de etiquetas

1. El aprendizaje semi-supervisado es un tipo de aprendizaje en el que el entrenamiento se efectúa con datos etiquetados y no-etiquetados. Se hace uso de los datos adicionales sin etiquetar para capturar mejor la forma de la distribución subyacente de los datos y generalizar mejor a nuevas muestras.

2. Los modelos disponibles son LabelPropagation y LabelSpreading. Ambos construyen un grafo de similitud sobre todos los elementos del conjunto de datos de entrada, si la estructura de los datos no-etiquetados es consistente con la estructura de clases de los datos etiquetados, estas etiquetas se propagan por el árbol a los datos no-etiquetados.

Tema 5

Introducción a las redes neuronales

5.1. Material de estudio

Los materiales de estudio de este tema son la primera parte del capítulo 10 de Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition hasta Implementing MLPs with Keras exclusive y adquirir un conocimiento de las herramientas que en ellos se comenta principalmente a partir de la documentación de scikit-learn. Iremos detallando el uso de estos materiales en las secciones siguientes.

Los códigos de ejemplo para este tema ya han sido descargados en temas anteriores.

5.2. Comentarios

En este tema vamos a dar una breve introducción a las redes neuronales desde un punto de vista más bien teórico. En el libro comenta la historia de los inicios de las redes neuronales como un intento de implementar el comportamiento de las neuronas biológica de manera muy simplificada. Empezando por el modelo de McCulloch-Pitts siguiendo con el perceptrón hasta el perceptrón multicapa y su entrenamiento por retropropagación (backpropagation), la explicación de la retropropagación se completa con la sección Gradient Descent del capítulo 4 del libro. El tema termina comentando los usos de las redes neuronales para regresión y clasificación. En esta última sección se habla de la función softmax que se introduce en el capítulo 4 del libro, basta con conocer su definición y para qué sirve.

El notebook asociado a este tema es 10_neural_nets_with_keras.ipynb de handson-ml2-master hasta la celda 8 inclusive. En él se usan principalmente las herramientas Tensorflow y Keras que instalaremos en el tema 7. Sin embargo en las 8 primeras celdas del notebook se usan las herramientas que proporciona scikit-learn 1.17. Neural network models (supervised) para trabajar con perceptrones, la lectura y estudio de los contenidos del enlace complementan el estudio de este tema.

Dado que no se va a instalar TensorFlow en este tema, para ejecutar esas 8 primeras celdas sin problemas hay que comentar en la celda 1 del notebook las siguientes líneas:

```
import tensorflow as tf
```

```
assert tf.__version__ >= "2.0"
```

5.3. Actividades autoevaluables

Realizar los ejercicios del 1 al 5 inclusive que se encuentran en la sección Exercices del capítulo 10 Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow. En el apéndice A de Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition están las soluciones. En el ejercicio 1 se propone "jugar" con el simulador TensorFlow Playground, una interesante actividad antes de pasar a programar en el tema siguiente. En el ejercicio 3 se habla de la regresión logística, en el tema 8 de la asignatura Modelado Estadístico de datos se estudia este modelo, para la realización del ejercicio bastará con leer el apartado Estimating Probabilities del principio de la sección Logistic Regression del capítulo 4 de Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow. Realizar también los ejercicios 4 a 7 que tratan del descenso del gradiente, inclusive del capítulo 4.

Aparte de los ejercicios de libro contestar estas cuestiones:

1. Ventajas y desventajas del perceptrón multicapa.
2. ¿Cuál es la complejidad computacional del perceptrón multicapa?
3. ¿Qué transformación de los datos es conveniente para el perceptrón multicapa?

Las soluciones a estas cuestiones se hallan en la documentación de scikit-learn, más concretamente:

1. Al final del 1.17.1. Multi-layer Perceptron.
2. En 1.17.6. Complexity.
3. El primer punto de 1.17.8. Tips on Practical Use.

5.4. Actividades evaluables

Este tema no tiene actividades evaluables.

Tema 6

Aspectos prácticos del uso de perceptrones multicapa.

6.1. Material de estudio

Los materiales de estudio de este tema son la segunda parte del capítulo 10 de Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition desde Implementing MLPs with Keras inclusive y adquirir un conocimiento de las herramientas que en ellos se comenta principalmente a partir de la documentación de Keras y TensorFlow. Iremos detallando el uso de estos materiales en las secciones siguientes.

Los códigos de ejemplo para este tema ya han sido descargados en temas anteriores.

6.2. Comentarios

6.2.1. Instalación de las herramientas necesarias

Instalación local

Para instalar las herramientas TensorFlow y Keras lo mejor es no seguir las indicaciones del libro. Sino iniciar Anaconda Navigator, con permisos de administrador (en Windows), hacer click en Enviroments, listar paquetes "no instalados" o "todos" y buscar "keras". En la lista aparecerán varios paquetes, los importantes son keras y keras-gpu. Si tenéis un ordenador con una GPU de NVIDIA con CUDA Compute Capacity $>= 3.5$ marcad keras-gpu, si no, marcad keras. A continuación haced click en el botón Apply que aparecerá abajo a la derecha. No es necesario marcar más paquetes, keras arrastrará todos los demás que son necesarios entre ellos tensorflow y tensorboard. Si la versión de tensorflow-eigen (o tensorflow-mlk) es $< 2.0.0$ tendréis que actualizar anaconda primero, esto solo debería pasar si la instalasteis antes de empezar este curso. Para muchas de las tareas usuales es preferible, por la comodidad que aporta, usar Keras que usar TensorFlow directamente.

Si ya se tiene instalado TensorFlow versión 1 lo mejor es instalar TensorFlow versión 2 en un nuevo entorno como se comenta en la sección siguiente salvo que el punto 4 no sería necesario,

ir directamente del punto 3 a 5.

Ejecución en la nube

Cuando escribía el texto de este tema accedí al github de los notebooks y ví que hace un mes han sido modificados de manera que incluyen un botón para ejecutarlos en Google Colab. Google Colab nos evita tener que instalar las herramientas en local, usa TensorFlow versión 1 por defecto aunque se le puede decir que emplee la 2, ver enlace, los notebooks del libro ya usan la 2. En este blob se da información de cómo usar aceleración por hardware mediante GPUs, también se pueden usar TPUs. Para los casos que nos ocupan el uso de GPUs es recomendable frente al de TPUs, ver comparativa. El uso de aceleración por hardware es muy recomendable frente a usar solo la CPU, se pueden conseguir reducciones de tiempo de 10 veces o incluso de hasta 50. Esto valores los he observado con ejecuciones en local, teniendo el hardware apropiado en el PC. Sin embargo para que se note la mejora en la velocidad con GPUs en Google Colab hay que usarlo por la noche que es cuando menos usuarios están usándolo en Europa, de día puede ser incluso más lento que usando CPU.

Si queréis utilizar Google Colab volved a descargar los notebooks del github.

Google Colab también permite la ejecución en local instalándonos las herramientas necesarias.

Código de testeo de GPUs

Este código testea si TensorFlow detecta detecta nuestra(s) GPU(s) o las de Google Colab:

```
from tensorflow.python.client import device_lib
def get_available_gpus():
        local_device_protos = device_lib.list_local_devices()
        return [x.name for x in local_device_protos if x.
           device_type == 'GPU']
print("Available GPUs ")
get_available_gpus()
```

si el resultado es algo como ['/device:GPU:0'] es que la ha detectado, si hay más de una aparecerán :1, :2, etc.

Código para liberar GPUs

Puede ser conveniente incluir este código al final de los notebooks que usen tensorflow-gpu, si no puede ser que la memoria de la GPU no se libere hasta que se cierre Jupyter.

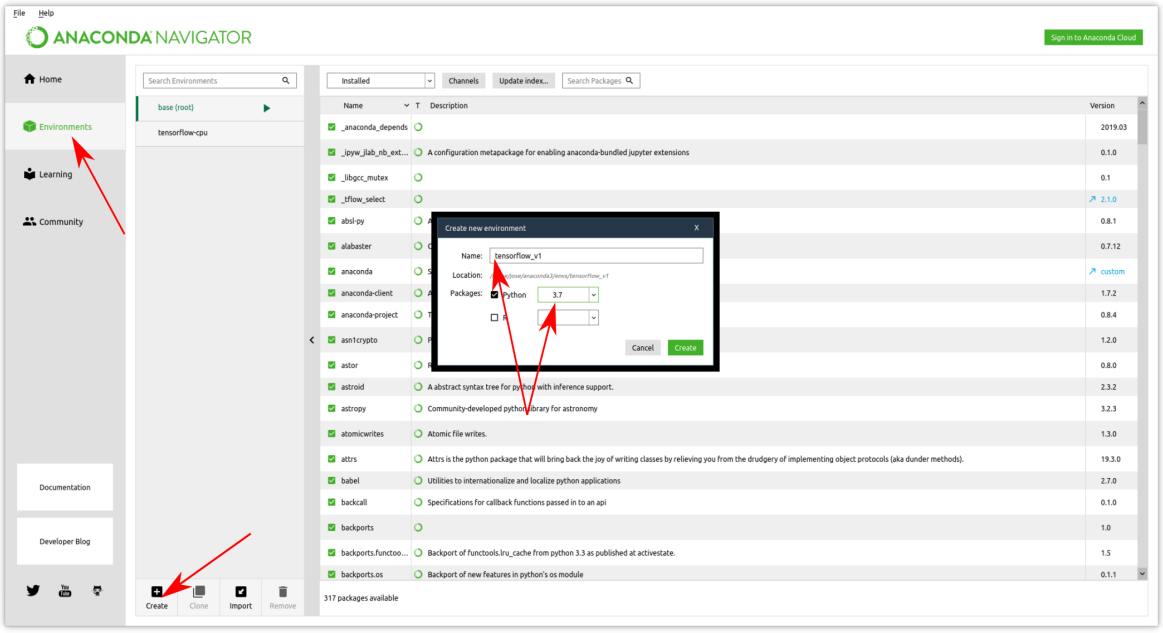
```
from numba import cuda
cuda.select_device(0)
cuda.close()
```

si hay más devices añadir cuda.select_device(1), etc.

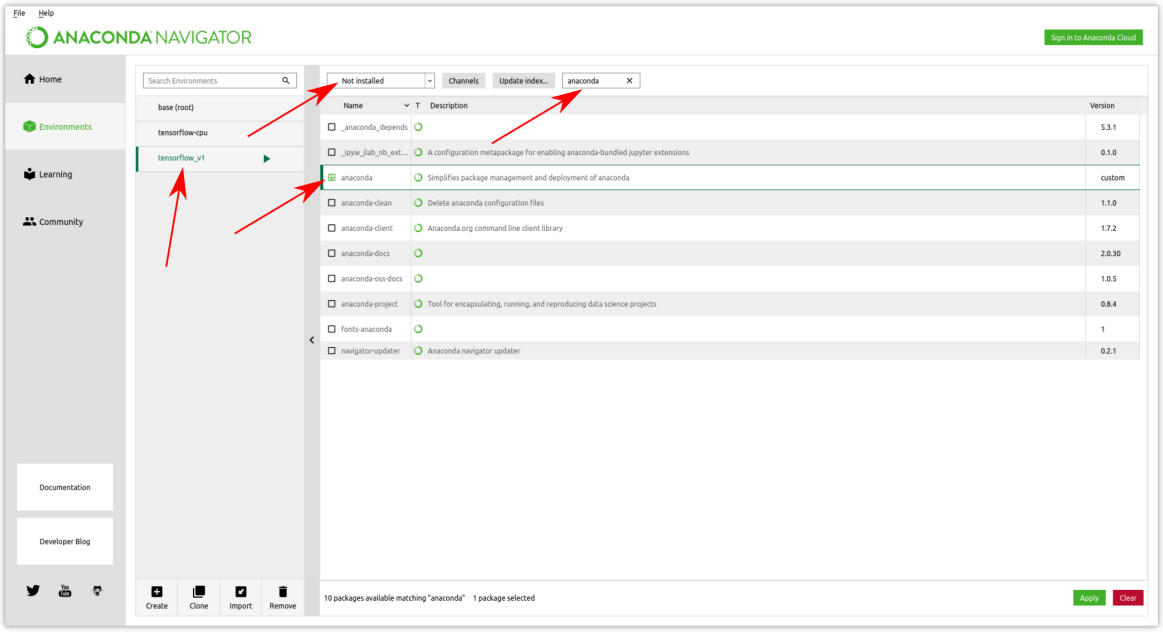
6.2.2. Instalación opcional de TensorFlow versión 1

Con las instrucciones anteriores se instala TensorFlow en su nueva versión 2 que es la que necesitamos para los notebooks del libro del curso y las actividades. Sin embargo existen bastantes diferencias entre la versión 2 y la 1 y gran parte de los ejemplos que se pueden encontrar en Internet hacen uso de la versión 1 hasta que se actualicen. Google digamos que está forzando el uso de la nueva versión y que esta tiene una integración más fluida con Keras, pero no hay una traducción directa del código de para adaptarlo a la nueva versión, consultar la guía de migración. Si se quiere ejecutar ejemplos que usen la versión 1 lo mejor es crear un nuevo entorno en Anaconda. Damos las instrucciones a continuación, hay que ejecutar Anaconda Navigator con permisos de administrador (en Windows):

1. Crear el nuevo entorno, al que denominaremos por ejemplo tensorflow_v1, instalando los paquetes básicos.



2. Instalar anaconda. Elegir el entorno, paquetes Not installed, buscar anaconda y click izquierdo sobre la marca a la izquierda del paquete. La instalación de anaconda arrastra muchos paquetes entre ellos matplotlib, jupyter, numpy, pandas, scipy, seaborn, scikit-learn.



3. De la misma forma instalar conda.

4. A continuación se instala tensorflow para cpu (paquete tensorflow a secas) o gpu (paquete tensorflow-gpu), las versiones pueden ser la 1.15.0 o la 1.14.0. Para ello procedemos como con los paquetes anteriores pero hacemos click derecho sobre la marca a la izquierda del paquete y elegimos en el desplegable Mark for specific version installation y en el siguiente desplegable elegimos la versión.

5. A continuación instalamos directamente, igual que los demás que hemos instalado en 2. y 3., keras o keras-gpu, dependiendo de si hemos instalado tensorflow o tensorflow-gpu.

6. Opcionalmente si queremos que se muestren los gráficos de habrá que instalar pydot y pygraphviz, así como el propio Graphviz que no es un paquete Python. Esto también permite exportar los Árboles de Decisión.

6.2.3. Comentarios sobre el texto del libro

En la sección Building an Image Classifier Using the Sequential API se va a crear una red que clasifique fotos de prendas de vestir de MNIST usando el Sequential model de Keras, también hay que conocer su API. En el libro se crea el modelo, se compila, se entrena y evalúa y se usa para hacer predicciones. A continuación en la sección Building a Regression MLP Using the Sequential API se aplica el modelo secuencial para hacer predicciones mediante regresión.

En las siguientes secciones se comentan usos de la Functional API de Keras, cómo crear subclases de las APIs comentadas, cómo guardar y recuperar un modelo previamente entrenado, el uso de callbacks para un mejor control del proceso de entrenamiento. A continuación se comenta el uso de Tensorboard para visulización. Termina el texto con explicaciones de cómo ajustar los hiperparámetros de una red, los más importantes son el número de capas ocultas y el número de neuronas en cada capa oculta.

6.2.4. Comentarios sobre el notebook

En notebook asociado a este tema es 10_neural_nets_with_keras.ipynb. Conviene que ejecutar el notebook para comprobar que todo va bien. La celda que consumen más tiempo de ejecución es la 100, donde se usa RandomizedSearchCV. Si no usamos GPU, poniendo el parámetro n_jobs > 1 o con un valor negativo se aumenta la velocidad, ver enlace anterior.

6.3. Actividades autoevaluables

Realizar los ejercicios del 6 al 9 inclusive que se encuentran en la sección Exercices del capítulo 10 Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow. En el apéndice A de Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition están las soluciones.

6.4. Actividades evaluables

La actividad evaluable va a consistir en realizar el ejercicio 10 del capítulo 10 del libro. En el github de los notebooks del libro este ejercicio aparece como TODO. En el libro se comenta que se entrene un MLP profundo, vamos a limitar este requisito ya que si no lo tiempos de ejecución se puede disparar especialmente si no se usa GPU. En el ejemplo del libro en la sección CREATING THE MODEL USING THE SEQUENTIAL API se construye una red con una capa oculta. En esta actividad se va a pedir que se construyan otras redes con al menos 2 y 3 capas ocultas opcionalmente otra con cuatro. El número de neuronas de las capas ocultas queda a vuestra discreción. Recordad que no se va a evaluar estrictamente el alcanzar el 98 % de precisión, basta con no quedarse muy lejos. Lo que sí hay que hacer es diseñar un procedimiento para buscar el learning rate apropiado, dibujar la función de pérdida, usar early stopping y las demás cuestiones que se piden en el ejercicio del libro.

Habrá que crear un notebook con celdas de tipo code y de tipo markdwon, en las de este último tipo se incluirán comentarios de lo que se esté haciendo en las de tipo code o sobre los resultados.

6.4.1. Valoración

Se valorará la creación del código que realice los requisitos enumerados, la presencia de comentarios de lo que hace el código y sus resultados y las comparaciones entre los resultados obtenidos por las distintas redes, así como algún gráfico que muestre los resultados. Para terminar incluir un apartado de conclusiones. Tanto los comentarios como las conclusiones no es necesario que sea extensos, sino que describan de forma concisa.

6.5. Otras actividades opcionales

Leer los artículos que se citan en el libro y en las notas al final del capítulo. Leer el apéndice E del libro sobre otros modelos de redes neuronales. Para profundizar más en los aspectos matemáticos del tema leer el capítulo 11 de The Elements of Statistical Learning.

Tema 7

Máquinas de vectores soporte (SVM).

7.1. Material de estudio

Los materiales de estudio de este tema son el capítulo 5 de Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition y adquirir un conocimiento de las herramientas que en ellos se comenta principalmente a partir de la documentación de scikit-learn. Iremos detallando el uso de estos materiales en las secciones siguientes.

Los códigos de ejemplo para este tema ya han sido descargados en temas anteriores.

7.2. Comentarios

SVM es uno de los algoritmos de Aprendizaje Automático más populares, fue introducido en 1963. Permite la clasificación, la regresión y la detección de anomalías. Originalmente es un clasificador binario aunque puede extenderse a clasificación multiclase. Está basado en encontrar el margen ("calle") más amplio que separa a las dos clases.

En el libro comienza por describir el clasificador lineal, primero el caso completamente separable (hard margin) para pasar a continuación al caso no completamente separablr (soft margin). Continúa con la clasificación no lineal, ahora la "calle" no es recta. La clasificación puede realizarse añadiendo características que son funciones no lineas de las características originales o mediante una función se similitud. Estas técnicas se implementan mediante la aplicación kernels.

A continuación se explica la regresión mediante SVM, ahora el concepto de "calle" que separa las clases se convierte en "calle" que contiene tantas muestras como sea posible. Se puede hacer regresión lineal y no lineal.

En la sección Under the Hood se describen las técnicas matemáticas en las que está fundamentado SVM, que al fin y al cabo es un problema de optimización cuadrática. Se describen las optimizaciones para hard y soft margin, las funciones de decisión y predicción, el problema dual, las ecuaciones de los kernels más comunes y la forma de entrenar SVM incrementalmente (on-line).

Para completar los contenidos del libro tenemos la documentación de scikit-learn 1.4. Support Vector Machines. En esta documentación se comentan ventas y desventajas del uso de SVM.

Se describen las clases para clasificación SVC, NuSVC y LinearSVC, esto enlaces llevan a las APIs de las clases que hay que conocer para su uso. También se comenta como usar SVM para la clasificación multiclase, un ejemplo lo tenemos en el ejercicio 9 del libro. SVM es un método de clasificación discriminativo, no aporta probabilidades de pertenencia a las clases, pero SVC y NuSVC disponen de una opción para proporcionarlas usando validación cruzada. También se comenta cómo usar las clases proporcionadas por scikit-learn en problemas no balanceados.

Las clases para regresión son SVR, NuSVR y LinearSVR.

Para detección de anomalías (outliers) scikit-learn dispone de OneClassSVM.

La documentación también comenta la complejidad computacional de los distintos procedimientos y da consejos para el uso práctico. Como lectura opcional se puede ver cómo crear kernel personalizados. El resto de secciones de la documentación, la formulación matemática, ya se han visto en el libro.

7.3. Actividades autoevaluables

Realizar los 10 ejercicios que se encuentran en la sección Exercices del capítulo 5 Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow. En el apéndice A de Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition y en el notebook 05_support_vector_machines.ipynb están las soluciones.

7.4. Actividades evaluables

La actividad evaluable asociada a este tema y al módulo 3 de la asignatura va a constituir en un ejercicio de clasificación y otro de regresión usando los datos de los alquileres vacacionales ofrecidos en la plataforma AirBnB que ya se usaron en la actividad del tema 3, ver sección 3.7. Para ello habrá que crear un notebook con celdas de tipo code y de tipo markdwon, en las de este último tipo se incluirán comentarios de lo que se esté haciendo en las de tipo code o sobre los resultados.

Para el estudio estadístico y limpieza de datos, ver subsección 3.7.2, se pueden tomar lo que ya se hiciera en la actividad del tema 3.

La fecha de entrega de la actividad aparece en la plataforma.

7.4.1. Clasificación

La tarea de clasificación consistirá en clasificar los datos según el tipo de alojamiento, definido en el campo room_type, a partir del resto de características. Es decir, en room_type estarán codificadas las clases y en el resto de campos los atributos.

Para el estudio estadístico y limpieza de datos, ver subsección 3.7.2, se puede tomar lo que ya se hiciera en la actividad del tema 3.

La clasificación se realizará usando LinearSVC y SVC con sus parámetros por defecto, para el entrenamiento y validación se seguirá lo descrito en los dos primeros puntos de la subsección

3.7.3, teniendo en cuenta que ahora son dos modelos. En cuanto al punto 3, afinación de hiperparámetros, se realizará para SVC, usando el kernel RBF, buscando los mejores valores para γ y C usando GridSearchCV. En este caso no se van a dar sugerencias del rango de valores a probar, por lo que hay que realizar búsquedas de manera que cada una refine la anterior. Proponemos seguir las indicaciones de la sección 3.2 de A Practical Guide to Support Vector Classification, bastará con dos búsquedas. Para crear las listas de parámetros se puede hacer a mano o usando numpy.logspace. Las listas están equiespaciadas en escala logarítmica, si se realizara una tercera búsqueda se podría usar una escala lineal. La idea es escala logarítmica para rangos elevados y lineal para rangos no elevados.

7.4.2. Regresión

La tarea de regresión consistirá en considerar el precio por noche como variable dependiente, y el resto de campos como variables independientes. Es decir, se tratará de predecir los valores del campo price a partir de todos los demás. Convendrá repasar capítulo 2 de Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition.

Puntos a tener en cuenta:

- Convendrá estudiar estadísticamente las variables price, minimun_nights y calculated_host_listings_count para ver si hay outliers que posiblemente sea conveniente eliminar.

- Probar con LinearSVR y SVR con parámetros por defecto, realizando el entrenamiento para cada caso usando todas las variables independientes y todas las independientes excepto neighbourhood, ya que esta variable es categórica y tiene muchos valores, lo que podría afectar a los resultados. Si los resultados aconsejan eliminar neighbourhood el siguiente punto se realizará sin tener en cuenta dicha variable.

- Para el algoritmo que mejor resultado haya dado en el punto anterior realizar una afinación de hiperparámetros como se comentó en el apartado de clasificación:

    - C si fuera LinearSVR, realizando 3 búsquedas.
    - γ y C si fuera SVR, realizando 2 búsquedas.

7.4.3. Valoración

Se valorará la creación del código que realice los requisitos enumerados, la presencia de comentarios de lo que hace el código y sus resultados y las comparaciones entre los resultados obtenidos por los distintos algoritmos, así como algún gráfico que muestre los resultados. Para terminar incluir un apartado de conclusiones. Tanto los comentarios como las conclusiones no es necesario que sea extensos, sino que describan de forma concisa.

7.5. Otras actividades opcionales

Leer los artículos que se citan en el libro y en las notas al final del capítulo. Leer el apéndice C del libro sobre el problema dual de SVM.

Tema 8

Selección de variables y reducción de la dimensionalidad.

8.1. Material de estudio

Los materiales de estudio de este tema son el capítulo 8 de Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition y adquirir un conocimiento de las herramientas que en ellos se comenta principalmente a partir de la documentación de scikit-learn. Iremos detallando el uso de estos materiales en las secciones siguientes.

Los códigos de ejemplo para este tema ya han sido descargados en temas anteriores.

8.2. Comentarios

La selección de variables y reducción de dimensionalidad es un conjunto de técnicas para disminuir el número de características a considerar en un problema de Aprendizaje Automático.

En el capítulo 8 del libro se da primero una introducción a estas cuestiones.

Primero se comenta con ejemplos el problema de la maldición de la dimensionalidad y a continuación se ven los principales enfoques: la proyección y el manifold learning, que se podría traducir como aprendizaje de variedades.

A continuación se pasa a estudiar la técnica conocida como Análisis de Componentes Principales (PCA) que es una técnica de aprendizaje nosupervisado. Esta técnica está basada en la descomposición de una matriz en sus componentes principales, si se quiere profundizar en los aspectos matemáticos de SVD consultar el capítulo 58 de Handbook of Linear Algebra, second edition. El libro comenta las implementaciones disponibles en scikit-learn, ver en su documentación 2.5. Decomposing signals in components (la primera sección 2.5.1) y 6.5. Unsupervised dimensionality reduction, así como los enlaces en estas páginas a las APis (PCA,IncrementalPCA y KernelPCA) y ejemplos de las clases comentadas en el libro.

La siguiente técnica comentada en el libro es Locally Linear Embedding (LLE) que es una técnica no lineal. La documentación de scikit-learn de esta técnica y de algunas de las las comentadas en la siguiente sección del libro, Other Dimensionality Reduction Techniques, se hallan en

2.2. Manifold learning y en los enlaces a sus APis (LLE, MDS, Isomap y t-SNE).

Termina el capítulo comentado otras técnicas, varias de las cuales se han enumerado en el párrafo anterior y además LDA y Random projection, estas última es material complementario no evaluable.

Como ampliación, no evaluable, se pueden leer el resto de secciones de 2.5. Decomposing signals in components, de la 2.5.2 a la 2.5.7.

8.3. Actividades autoevaluables

Realizar los ejercicios que se encuentran en la sección Exercices del capítulo 8 Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow. En el apéndice A de Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition y en el notebook 08_dimensionality_reduction.ipynb están las soluciones.

8.4. Actividades evaluables

La actividad evaluable va a consistir principalmente en la comparación de clasificaciones realizadas empleando técnicas de reducción de dimensionalidad.

Se suministra el notebook load_faces.ipynb, disponible en la carpeta code de los documentos de Alf, en el que se cargan las imágenes de caras obtenidas con el método de scikit-learn fetch_olivetti_faces. En el notebook se carga el dataset y se hace uso de display(Markdown(faces.DESCR)) para mostrar la documentación asociada al dataset, esta documentación es la misma que aparece en el enlace anterior, utilizar este método puede proporcionar información interesante si los creadores del dataset se han preocupado en proporcionarla. En las celdas siguientes las imágenes se centran y se define un método para mostrar algunas de ellas.

La actividad consiste en:

1. Comprimir las imágenes aplicando PCA de manera que se preserve el 95 % de la varianza y mostrar las 8 primeras imágenes originales y descomprimidas. Crear un nuevo dataset con las imágenes comprimidas. ¿Qué número de píxeles quedan al comprimir?

2. Lo mismo que en 1. usando LDA.

3. Dividir los datos en conjunto de entrenamiento y test de manera que el conjunto de test sea un 20 % del total, tanto para los datos originales como para los comprimidos con PCA y LDA. Repasar la documentación de scikit-learn referida a SVM para ver si es conveniente aplicar transformaciones a los datos.

4. Realizar una afinación de hiperparámetros para LinearSVC y SVC con kernel 'rbf' para la clasificación de los datos originales, realizando la búsqueda de parámetros como se comentó en las subsecciones 7.4.1 y 7.4.2 (cambiando SVR por SVC). Usar para el parámetro 'cv'

de GridSearchCV el valor 10. Medir los tiempos en milisegundos de las dos ejecuciones de GridSearchCV. Se recomienda usar multiproceso.

5. Realizar 4. usando los datos comprimidos mediante PCA.

6. Realizar 4. usando los datos comprimidos mediante LDA.

7. Comparar los parámetros resultantes para los mejores estimadores obtenidos en 4., 5. y 6. así como sus accuracies y los tiempos de ejecución totales de las búsquedas y la media de las búsquedas de cada GridSearchCV teniendo en cuenta el número de valores de los parámetros en cada GridSearchCV.

8.4.1. Valoración

Se valorará la creación del código que realice los requisitos enumerados, la presencia de comentarios de lo que hace el código y sus resultados y las comparaciones entre los resultados obtenidos, así como una tabla que muestre los resultados. Para terminar incluir un apartado de conclusiones. Tanto los comentarios como las conclusiones no es necesario que sea extensos, sino que describan de forma concisa.