Introducción al Aprendizaje Automático

Grado en Ingeniería Informática (Esp. Computación)

Práctica 3
Clasificación

Objetivos

El objetivo de esta primera práctica es implementar algunos de los algoritmos de clasificación estudiados en la clase de teoría, comparando los resultados de nuestra implementación con los que producen otras librerías comerciales, tanto a nivel de resultado como de eficiencia.

Para llevar a cabo la práctica, además de usar nuestra propia implementación, utilizaremos dos librerías de Python muy populares en el ámbito del machine learning: Turi Create[1] y scikit-learn[2].

Desarrollo de la práctica

1. Codifica una función que aplique el algoritmo de aprendizaje del perceptrón, siguiendo las directrices que se explicaron en clase.

2. Desarrolla un programa principal que haga uso de estas dos funciones, y que las aplique sobre el siguiente conjunto de datos:

x1	x2	y
1	0	0
0	2	0
1	1	0
1	2	0
1	3	1
2	2	1
2	3	1
3	2	1

3. Codifica ahora una función que aplique el algoritmo de clasificación de regresión logística, siguiendo también las directrices vistas en clase, y aplícalo al mismo conjunto de datos (umbral 0.5 para clasificación).

4. Construye un conjunto de funciones que reciban dos vectores $(y, \hat{y})$ y que construyan la matriz de confusión para este conjunto de resultados. Construye ahora un conjunto de funciones que reciben esta matriz de confusión (en el formato que hayas definido) y

[1] https://github.com/apple/turicreate
[2] https://scikit-learn.org/stable

calculen las métricas de calidad que hemos estudiado en clase: exactitud, recall, precisión, F_1 y F_b, selectividad y especificidad.

5. Con las funciones desarrolladas en el apartado 4, obtén los resultados de calidad de los clasificadores desarrollados (perceptrón y logística).

6. A continuación, vamos a trabajar con un ejemplo de análisis de sentimientos real. Se trata de determinar si la evaluación que se hace de una película es positiva o negativa. Los datos se tomarán de un conjunto de 50.000 revisiones, tomadas de la base de datos IMDB[3]. Para construir el clasificador, no utilizarás tu propia implementación, sino que utilizarás la que está disponible en Turi Create.

CONSEJOS

a) En primer lugar, crearás un *SFrame* que contendrá el fichero completo
b) Posteriormente, tendrás que conocer la frecuencia de cada una de las palabras que aparecen en las críticas. El módulo *text_analytics* de Turi Create proporciona funcionalidad para hacer este cálculo (específicamente, la función *count_words*). Consulta la documentación para ver cómo obtener esta cuenta y cómo añadir el resultado a tu *SFrame*.
c) Una vez dispongas de los datos de entrada y salida, construirás el clasificador utilizando la función *logistic_classifier*. Consulta la documentación para conocer su funcionamiento.

7. A continuación, evaluarás el modelo que has obtenido en el apartado 6. Puedes consultar la documentación de la función *model* para ver cómo hacerlo con Turi Create, aunque puedes desarrollar tu propio código aplicando el modelo que has obtenido en 6 sobre el conjunto de datos. Una vez obtenidas todas las predicciones, puedes obtener la matriz de confusión y las métricas que has desarrollado en el apartado 4.

8. Ahora vas a obtener la curva ROC para el modelo desarrollado en el apartado 6. Recuerda que has de ir modificando el umbral de clasificación y, en cada caso, obtener las predicciones, la matriz de confusión, la especificidad y la sensibilidad. Dibuja la gráfica con la curva.

9. En esta práctica vamos a probar un modelo de red neuronal para llevar a cabo una tarea de reconocimiento de imágenes, a saber, el reconocimiento de dígitos a partir de imágenes caligráficas (*dataset* MNIST). El procedimiento a seguir se ha visto en la clase de teoría. Procederemos a realizar los siguientes pasos: (a) cargar el *dataset*, (b) redimensionarlo para que se ajuste a las entradas de la red y para que la salida sea categórica (c) Definir la arquitectura de red (784:120relu:64relu:10softmax) (d) compilar la red y entrenarla.

Intenta jugar con algunos de los hiperparámetros de la red para intentar mejorar la predicción. Presentarás un informe de los modelos que has probado. Para llevar a cabo esta evaluación, puede ser interesante que particiones el dataset en *training* y *test*.

10. Para terminar esta práctica, vamos a construir un modelo de red neuronal para llevar a cabo una tarea de regresión, la que vimos en la práctica anterior. Vamos a ensayar la siguiente arquitectura de partida para modelar el precio de las viviendas:

[3] https://moodle.uco.es/m2122/mod/resource/view.php?id=290889

- Una capa de entrada de tamaño 38 (el número de columnas del conjunto de datos)
- Una capa oculta de tamaño 128 con una función de activación ReLU y un parámetro de *dropout* de 0,2
- Una capa oculta de tamaño 64 con una función de activación ReLU y un parámetro de *dropout* de 0,2
- Una capa de salida de tamaño 1 sin función de activación

Para entrenar la red, hay que usar como función de pérdida el error cuadrático medio, y es recomendable usar el optimizador Adam.

Intenta jugar con los hiperparámetros para obtener el modelo con menor error de prueba. Puedes hacer dos particiones (train/test) para valorar la calidad de tus diferentes configuraciones.