Fundamentos de Aprendizaje Automático 2019/2020

PRÁCTICA Nº 2

1. Objetivo

En esta práctica continuaremos el estudio de los clasificadores implementando dos nuevos algoritmos: *vecinos próximos* y *regresión logística*. Siguiendo la filosofía de la anterior práctica se deben comparar sus resultados con los que proporciona la librería de *scikit-learn* (http://scikit-learn.org/stable/). También se trabajará con las representaciones gráficas de algunos conjuntos de datos especialmente preparados y se continuará aplicando el análisis ROC para comparar clasificadores.

2. Tareas

La planificación temporal sugerida y las tareas a llevar a cabo son las siguientes:

- *1ª semana:* Implementar el algoritmo de *vecinos próximos* (clase **ClasificadorVecinosProximos**) para realizar una tarea de clasificación de los siguientes conjuntos de datos, probando con diferentes valores de vecindad (K=1, 3, 5, 11, 21 y 51). Ambos conjuntos se encuentran en Moodle.

  - ✓ Conjunto de datos ***online shoppers*** (https://archive.ics.uci.edu/ml/datasets/Online+Shoppers+Purchasing+Intention+Dataset#)
  - ✓ Conjunto de datos ***wdbc*** (http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/).

  En el uso del algoritmo de vecinos próximos es recomendable normalizar los atributos de forma que cada uno de ellos tenga media 0 y desviación típica 1. La normalización es importante puesto que evita problemas de escala. Para llevar a cabo la normalización se añadirán nuevos métodos, tal y como se describe en el apartado 3. Por otro lado, hay que tener en cuenta que en el dataset *online shoppers* aparecen atributos nominales, por lo que debemos utilizar una distancia apropiada para ellos. La distancia más sencilla para atributos nominales es la que devuelve un 0 si los valores son iguales y un 1 si son distintos. Sin embargo, si el atributo sigue un orden se puede intentar mantener ese orden en el cálculo de la distancia. Un ejemplo sería el atributo *month* en el citado conjunto.

- *2ª semana*: Implementar el algoritmo de *regresión logística* (clase **ClasificadorRegresionLogistica**), aplicando el método de maximización de la verosimilitud visto en las clases de teoría. Comparar los resultados obtenidos para los conjuntos de datos ***online_shoppers*** y ***wdbc*** con el algoritmo de *regresión logística* y con el de *vecinos próximos*. Ten en cuenta los atributos

nominales del primer conjunto, para transformarlos mediante una codificación *OneHotEncoder* como la vista en la práctica uno.

- *3ª semana*: Analizar los dos conjuntos de datos anteriores con la librería de scikit-learn para vecinos próximos y con la correspondiente para regresión logística, según los detalles del apartado 4. Seguidamente, para analizar características como las regiones y las fronteras de decisión, se van a representar gráficamente los conjuntos de datos ***example1***, ***example2***, ***example3*** y ***example4*** empleando la función descrita en el apartado 5. Para obtener los valores con los que trabajar debes aplicar los algoritmos de Vecinos Próximos y Regresión Logística a los cuatro conjuntos. Puedes usar la implementación propia o la de scikit learn para esta tarea.

  Ahora que se han implementado tres clasificadores se puede extender el Análisis ROC que iniciamos en la primera práctica. Para usar los mismos datos, debes ejecutar Naive Bayes con los dos nuevos conjuntos: online shoppers y wdbc. Posteriormente realiza el Análisis ROC para comparar los resultados en estos conjuntos con los tres clasificadores.

3. Normalización de datos

Para la normalización de los datos, se implementarán dos métodos nuevos en la clase que consideres más apropiada, justificando la elección:

- `calcularMediasDesv(self,datostrain)`: esta función calculará las medias y desviaciones típicas de cada atributo continuo a partir de los datos de entrenamiento contenidos en la matriz `datostrain`.
- `normalizarDatos(self,datos)`: esta función normalizará cada uno de los atributos continuos en la matriz `datos` utilizando las medias y desviaciones típicas obtenidas en `calcularMediasDesv`.

4. Scikit-learn

Vecinos Próximos

Scikit-learn implementa dos clasificadores basados en vecindad. Para los propósitos de esta práctica interesa ***KNeighborsClassifier***. La versión básica de este algoritmo considera todos los vecinos por igual. Sin embargo, en algunas circunstancias es preferible ponderar el valor de los mismos y dar más importancia a los que estén más cercanos, dentro de la vecindad. El control de la opción básica o ponderada se establece con el parámetro weight. Si **weight='uniform'** se utiliza la opción básica (que corresponde también al valor por defecto) y si **weight='distance'** se asigna un peso proporcional a la inversa de la distancia. También se podría pasar una función propia para la distancia.

NOTA: Aprovecha esta característica de Scikit-Learn para incluirla en la implementación propia, mediante un parámetro *weight* que se pasa al constructor *ClasificadorVecinosPróximos*.

Regresión Logística

Scikit-learn proporciona una implementación de la *Regresión Logística* a través de la función ***LogisticRegression***.

5. Representación de fronteras de decisión para problemas de clasificación binarios con dos atributos

El fichero plotModel.py proporciona la implementación de la función plotModel:

```
plotModel(x,y,clase,clf,title,diccionarios)
```

`x`: valores del primer atributo para los ejemplos de entrenamiento
`y`: valores del segundo atributo para los ejemplos de entrenamiento
`clase`: etiquetas de los ejemplos de entrenamiento
`clf`: instanciación de un clasificador que implemente la clase abstracta Clasificador
`title`: string con el título de la gráfica
`diccionarios`: diccionario de datos

Por ejemplo, para el conjunto de datos *example1*.data se podría invocar a la función de la siguiente forma:

```
from plotModel import plotModel
import matpotlib.pyplot as plt
plotModel(dataset.datos[ii,0],dataset.datos[ii,1],dataset.datos
[ii,-1]!=0,clasificador,"Frontera",dataset.diccionarios)
```

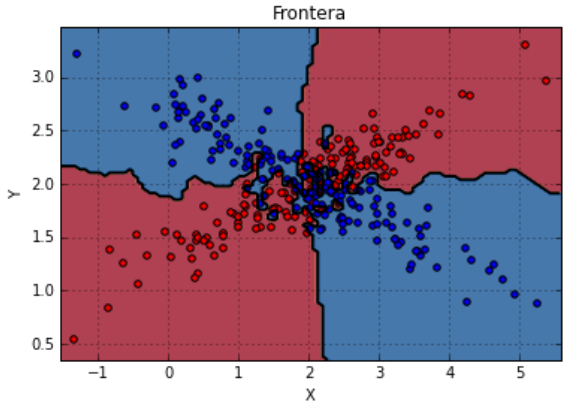
Donde clasificador es una instanciación de la clase ClasificadorVecinosProximos:
`clasificador=Clasificador.ClasificadorVecinosProximos()`

`ii` son los índices de los patrones de entrenamiento de la última partición de la estrategia de particionado escogida:
`ii=estrategia.particiones[-1].indicesTrain`

Para visualizar los puntos y la frontera de decisión del conjunto *example1.data* basta con invocar a plotModel desde Jupyter o un IDE. Para visualizar solo los puntos, sin las fronteras se puede incluir el siguiente código tras la invocación de plotModel.

```
plt.figure()
plt.plot(dataset.datos[dataset.datos[:,-1]==0,0],
         dataset.datos[dataset.datos[:,-1]==0,1],'bo')
plt.plot(dataset.datos[dataset.datos[:,-1]==1,0],
         dataset.datos[dataset.datos[:,-1]==1,1],'ro')
```

Se obtendría entonces la representación:



NOTA: la función plotModel invoca al método clasifica del clasificador sin proporcionar la etiqueta de los patrones en el parámetro *datostest*. Tener esto en cuenta a la hora de implementar la función clasifica de los métodos ***ClasificadorVecinosProximos*** y ***ClasificadorRegresionLogistica***.

6. Fecha de entrega y entregables

Semana del 18 al 22 de Noviembre de 2019. La entrega debe realizarse antes del comienzo de la clase de prácticas correspondiente. Se deberá entregar un fichero comprimido .zip con nombre **`FAAP2_<grupo>_<pareja>.zip`** (ejemplo `FAAP2_1461_1.zip`) y el siguiente contenido:

1. **Ipython Notebook (.ipynb)** con las instrucciones necesarias para realizar las pruebas descritas en los apartados anteriores y el correspondiente análisis de resultados. El Notebook debe estructurarse para contener los siguientes apartados:

Apartado 1	Resultados de la clasificación mediante vecinos próximos (implementación original) para diferentes valores de vecindad en los conjuntos de datos propuestos. Obtener los resultados tanto para datos normalizados como sin normalizar, con el objetivo de justificar el rendimiento del algoritmo en base a estas características.
Apartado 2	Resultados de la clasificación mediante regresión logística en los conjuntos de datos propuestos. Probar con diferentes valores para la constante de aprendizaje y el número de pasos.
Apartado 3	Resultados de la clasificación utilizando los algoritmos de Scikit-Learn para vecinos próximos y regresión logística. Comparación con los resultados de la implementación propia. Representación gráfica de los conjuntos de datos example1.data, example2.data, example3.data y example4.data para los dos

	algoritmos. Interpretación de dichas gráficas.
Apartado 4	Aplicar el análisis ROC para comparar los clasificadores Naive Bayes, Vecinos Próximos y Regresión Logística en los conjuntos online shoppers y wdbc. Comentar los resultados

2. **Ipython Notebook exportado como html.**

3. Código Python (**ficheros .py**) necesario para la correcta ejecución del Notebook

7. Puntuación de cada apartado

La valoración de cada apartado será la siguiente:

- ✓ **Vecinos próximos**: 3,5 puntos
- ✓ **Regresión logística**: 3,5 puntos
- ✓ **Scikit-Learn**: 1 punto
- ✓ **Representación gráfica**: 1 punto
- ✓ **Análisis ROC**: 1 punto