Fundamentos de Aprendizaje Automático 2016/2017

PRÁCTICA Nº 4

1. Objetivo

Tras el desarrollo y selección de algoritmos, la aplicación de métodos de clasificación y aprendizaje automático en aplicaciones reales puede incluir una fase de combinación de algoritmos. En ocasiones las mejores soluciones a un problema de clasificación no las da un único algoritmo, sino la combinación de varios clasificadores.

Como ejercicio en esta línea, en esta práctica se diseñará e implementará la combinación de varios clasificadores. A diferencia de las prácticas anteriores, ésta no conlleva el desarrollo de nuevos algoritmos, sino la gestión experimental de algoritmos y módulos software como piezas de construcción, utilizando implementaciones disponibles en la librería scikit-learn y/o el software desarrollado en las anteriores prácticas.

2. Tareas

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

- *1ª semana*: implementación y pruebas de conjuntos de clasificadores (clase ClasificadorEnsemble).
- *2ª semana:* implementación de conjuntos de clasificadores utilizando la librería scikit-learn y comparación de resultados.

Tanto la implementación propia de conjuntos de clasificadores, como la librería scikit-learn se probarán sobre el conjunto de datos ***digits*** con el que ya se ha trabajado en prácticas anteriores y tres conjuntos de datos que el propio estudiante elegirá entre los disponibles en el catálogo de UCI (https://archive.ics.uci.edu/ml/datasets.html). Se podrán utilizar también datos de cualquier otra procedencia, previa confirmación con el profesor de prácticas.

Se detallan a continuación las dos tareas a realizar.

1. Conjuntos de clasificadores (clase ClasificadorEnsemble)

Se implementarán los métodos entrenamiento y clasifica de una nueva clase ClasificadorEnsemble, que heredará de la clase abstracta Clasificador.

Formaremos un *ensemble* clasificador con los siguientes clasificadores:

- Naive Bayes.
- Regresión logística.
- Vecinos próximos.

Se utilizará la votación por mayoría simple como modo de agregación de clasificadores. Se podrá utilizar, a elección del estudiante, o bien la implementación de scikit-learn de estos clasificadores, o bien las implementaciones de las prácticas 1 y 2. Debe tenerse en cuenta que los métodos Regresión Logística y Vecinos Próximos se han implementado únicamente para trabajar con atributos continuos. También conviene recordar que el algoritmo de regresión logística para problemas multiclase, necesita utilizar el *wrapper* ClasificadorMulticlase.

2. Librería scikit-learn

Desde la versión 0.18 de scikit-learn, el submódulo `sklearn.ensemble` contiene la clase `VotingClassifier` que implementa un *ensemble* de clasificadores de acuerdo a diferentes estrategias de votación, estando la votación por mayoría entre ellas (`voting='hard'`). El constructor de la clase `VotingClasifier` recibe como parámetro (`estimators`) una lista de tuplas con tantos elementos como clasificadores formen el *ensemble*. El primer elemento de cada tupla, lo formará una cadena para identificar el clasificador, y el segundo elemento de cada tupla será una instanciación del clasificador.

```
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import VotingClassifier

X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
y = np.array([1, 1, 1, 2, 2, 2])


clf1 = LogisticRegression(random_state=1)
clf2 = GaussianNB()


model = VotingClassifier(estimators=[
        ('lr', clf1), ('gnb', clf2)], voting='hard')
```

Una vez instanciada la clase `VotingClassifier` se puede invocar a las funciones `fit` y `predict` tal y como se venía haciendo con los clasificadores de las prácticas anteriores:

```
model.fit(X_train, y_train)
predicted= model.predict(X_test)
```

Para más información, consúltese la ayuda de la clase: http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html. Se puede realizar la práctica utilizando la clase `VotingClassifier` de scikit-learn 0.18, aunque esta versión no sea la disponible en los laboratorios de prácticas.

Para versiones de **scikit-learn anteriores a la 0.18** (**como en los laboratorios**), la clase `VotingClassifier` no está disponible dentro de scikit-learn. Sin embargo, el autor de este código, Sebastian Raschka, tiene pública la implementación del paquete `mlxtend` en un repositorio de Github[1][2]. Este paquete contiene funciones y clases útiles para el análisis de datos y reconocimiento de patrones. Entre ellas, está la clase `EnsembleVoteClassifier`, precursora de la clase `VotingClassifier`, dentro del submódulo `mlxtend.classifier`. Para poder utilizar la clase `EnsembleVoteClassifier` se deberán seguir los siguientes pasos:

1. Descargar la versión 0.4.2 del paquete `mlxtend` del repositorio Github: https://github.com/rasbt/mlxtend/releases/tag/v0.4.2
2. Descomprimir el fichero y copiar la subcarpeta `mlxtend` en el mismo directorio donde se encuentre el Notebook.

Ahora es posible importar la clase `EnsembleVotingClassifier` como:

```
from mlxtend.classifier import EnsembleVoteClassifier
```

El funcionamiento de la clase `EnsembleVoteClassifier` es idéntico al de la clase `VotingClassifier`, aunque cambian ligeramente los parámetros del constructor:

```
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from mlxtend.classifier import EnsembleVoteClassifier

clf1 = LogisticRegression(random_state=1)
clf2 = GaussianNB()


X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
y = np.array([1, 1, 1, 2, 2, 2])

model = EnsembleVoteClassifier(clfs=[clf1,clf2,clf3], voting='hard')
```

[1] http://sebastianraschka.com/Articles/2014_ensemble_classifier.html
[2] https://github.com/rasbt/mlxtend

Una vez instanciada la clase `EnsembleVoteClassifier` se puede invocar a las funciones `fit` y `predict`:

```
model.fit(X_train, y_train)
predicted= model.predict(X_test)
```

3. Lenguaje de programación

El lenguaje a utilizar para el desarrollo de la práctica será Python.

4. Fecha de entrega y entregables

Domingo 18 de diciembre de 2016 a las 23.59h. Se deberá entregar un fichero comprimido .zip con nombre **`FAAP4_<grupo>_<pareja>.zip`** (ejemplo `FAAP4_1461_1.zip`) y el siguiente contenido:

1. **Ipython Notebook (.ipynb)** con las instrucciones necesarias para realizar las pruebas descritas en el apartado 2 y el correspondiente análisis de resultados. El Notebook debe estructurarse para contener los siguientes apartados:

Apartado 1	Efectividad del *ensemble* de clasificadores sobre los diferentes conjuntos de datos, comparada con los clasificadores individuales (p.e. una tabla o diagrama de barras por cada conjunto de datos). Utilizar una validación cruzada con 5 grupos como estrategia de particionado. Incluir comentarios analizando los resultados observados. Incluir un breve comentario describiendo cualquier detalle específico que el estudiante haya particularizado o desarrollado más allá de las instrucciones dadas en este enunciado, o que el estudiante considere de interés. Entre otros aspectos, será necesario incluir la descripción de los conjuntos de datos utilizados.
Apartado 2	Resultados de la clasificación mediante conjuntos de clasificadores en los cuatro conjuntos de datos utilizando la librería scikit-learn. Utilizar una validación cruzada con 5 grupos como estrategia de particionado. Comparar estos resultados con los obtenidos en la implementación propia.
Apartado 3	Obtener la matriz de confusión para el conjunto ***digits*** usando el *ensemble* y determinar qué dígitos son más difíciles de clasificar

en general. Se puede utilizar la función `confusion_matrix` de scikit-learn[3]. Este análisis puede hacerse sobre la última de las particiones de la estrategia de particionado.

(Opcional) Identificar tres de los patrones del conjunto ***digits*** que el *ensemble* de clasificadores no es capaz de clasificar correctamente. Tratar de determinar a qué puede deberse este error en la clasificación analizando las imágenes de estos dígitos.

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

3. Código Python (**ficheros .py**) necesario para la correcta ejecución del Notebook incluida la subcarpeta `mlxtend` si no se ha utilizado la clase `VotingClassifier`.

[3] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html