Reconocimiento de Patrones, Máster en Visión Artificial
Universidad Rey Juan Carlos
Curso 2016-2017
Práctica 1: Clasificadores generativos

1. Normas

La práctica es individual.

La memoria de la práctica junto con los programas python se entregarán en un fichero .zip en el Campus Virtual de la asignatura. **La fecha tope de entrega de la práctica es el día del examen de la asignatura - en enero**.

En la portada de la memoria deben aparecer el nombre y DNI del autor. No existe restricción alguna en cuanto al formato y extensión de la memoria aunque más de 10 páginas con 10pt de letra se puede considerar ya excesivo. Por otra parte se recuerda que la calidad de la memoria es un aspecto de importancia similar a los programas.

Importante: En la memoria se deben listar los sitios web de donde se haya sacado código usado en las prácticas (exceptuando los ejemplos de código de la documentación de sklearn).

2. Planteamiento

Esta práctica tiene varios objetivos:

1. Entender el esquema de implementación de un clasificador para entrenarlo y ejecutarlo.
2. Entender los clasificadores generativos paramétricos y no paramétricos
3. Introducir al alumno en el manejo de *scikit-learn* en python.
4. Entender de forma práctica el "No free-lunch theorem".

3. Desarrollo: implementación y evaluación de clasificadores generativos

Vamos a usar los diferentes elementos de Reconocimiento de Patrones implementados en el paquete *scikit-learn* de python (`http://scikit-learn.org/`). Se podrán utilizar los que se encuentran implementados en el paquete y habrá que implementar los que no se encuentren disponibles.

En la práctica **se pide** evaluar el rendimiento de diferentes posibilidades de representación de $p(\mathbf{x}|\alpha_i)$ (donde α_i es la etiqueta de la clase i):

1. Clasificador paramétrico basado en Gaussianas (no es el naïve Bayes de *sklearn*).
2. Clasificador paramétrico basado en Mezcla de Gaussianas, GMM (la estimación de GMMs está implementada en *sklean*).
3. Clasificador no paramétrico basado en ventanas de Parzen (la evaluación del kernel está implementada en *sklearn.*
4. Clasificador no paramétrico basado en los K vecinos más próximos, K-NN (implementado en *sklearn*).

Para la evaluación de la práctica trabajaremos sobre datos en 2 dimensiones. Se tendrán varias clases con función de densidad, $p(\mathbf{x}|\alpha_i)$, mezcla de Gaussianas y de parámetros conocidos. Conociendo las funciones de densidad reales seremos capaces de estimar un clasificador ideal (el de mínimo error) y compararlo con el comportamiento de clasificadores entrenados a partir de datos muestreados de las funciones de densidad conocidas.

En el Moodle de la asignatura se proporcionará un script, `test_practica.py`, en python con el esqueleto del código que hay que desarrollar. En el script (junto con otros auxiliares) se encuentrará implementada la función que establece las distribuciones de clase reales y la obtención de muestras de esa distribución para entrenar y probar.

En esta parte se pide implementar una clase por cada clasificador a probar en la que tengamos los métodos:

- **fit(X, y) ≡ entrenar clasificador**. Este método tendrá como entrada a X, los N datos de entrenamiento de d dimensiones un array de numpy $N \times d$ e y, las N etiquetas de clase asociadas a cada dato en otro array de numpy. El resultado de llamar a `fit` será que el objeto clasificador guardará en sus atributos lo necesario para clasificar nuevos datos.
- **predict(X) ≡ clasificar**. Este método tendrá como entrada a X, los M datos a clasificar de d dimensiones como un array de numpy $M \times d$. Este método devolverá un array de numpy con las M etiquetas estimadas.

Por tanto habrá que implementar las siguientes clases (la clase GaussianBayes se proporcionará en el Moodle de la asignatura como guía):

1 **GMMBayesClassifier** debe de realizar la búsqueda del mejor valor del número de componentes Gaussianas g (ver documentación de GMM en *sklearn*).

2 **ParzenClassifier:** La implementación de `ParzenClassifier.fit` debe de realizar la búsqueda del valor de h mediante validación cruzada de 5 grupos (5 fold).

3 **KNNClassifier**. La implementación de `KNNClassifier.fit` debe de realizar la búsqueda del valor de K mediante validación cruzada de 5 grupos (5 fold).

Y a continuación:

4 Generar muestras de datos de entrenamiento y pruebas para los conjuntos de gaussians 1, 2, 3 y 4 (ver el código de `test_practica.py`).

5 Entrenar los distintos clasificadores con las muestras de entrenamiento.

6 Probar el clasificador ideal (el construido con las GMMs conocidas) y los implementados en los datos de entrenamiento. Calcular error de clasificación, matrices de confusión y las métricas de rendimiento que se consideren oportunas (*sklearn* tiene muchas implementadas).

Se pide responder a las siguientes preguntas:

- ¿Cuál es el mejor clasificador para el conjunto de Gaussianas 1? Razonar la respuesta.
- ¿Cuál es el mejor clasificador para el conjunto de Gaussianas 2? Razonar la respuesta.
- ¿Cuál es el mejor clasificador para el conjunto de Gaussianas 3? Razonar la respuesta.
- ¿Ocurre algo raro con el conjunto de Gaussianas 4? Razonar la respuesta en términos del desbalanceo de datos (clases con diferente número de datos de entrenamiento)
- ¿Qué se te ocurre que se podría hacer para mejorar los resultados en el caso de Gaussianas 4? Si has probado tu solución ¿Ha mejorado el rendimiento del clasificador?

4. Valoración de la práctica

La puntuación máxima en cada apartado será la siguiente (si un apartado no se hace puntua 0):

- Evaluación del **GaussianBayes**, hasta 1 punto.
- Programación y evaluación del **GMMBayesClassifier**, hasta 2 puntos.
- Programación y evaluación del **ParzenClassifier**, hasta 2 puntos.
- Programación y evaluación del **KNNClassifier**, hasta 2 puntos.
- Respuesta a las preguntas del enunciado, hasta 1 punto.

- Memoria hasta 1 punto.
- Código hasta 1 punto.

y el tener la máxima puntuación en cada apartado dependerá de los siguientes criterios:

- Funcionamiento de los clasificadores.
- Limpieza del código
- Calidad de la memoria y la discusión de los resultados.
- Extensión de las pruebas realizadas (diferentes pruebas realizadas, utilización de datos no provistas junto con el enunciado, comparación con los clasificadores ya implementados en sklearn, etc.).

5. Datos proporcionados

Se proporcionarán los scripts necesarios en el Moodle de la asignatura.