Actas de las XXXIX Jornadas de Automática, Badajoz, 5-7 de Septiembre de 2018
https://doi.org/10.17979/spudc.9788497497565.0998

TUTORIAL PARA EL RECONOCIMIENTO DE OBJETOS BASADO EN CARACTERÍSTICAS EMPLEANDO HERRAMIENTAS PYTHON

Jose-Raul Ruiz-Sarmiento, Javier Monroy, Francisco-Angel Moreno, Javier Gonzalez-Jimenez
Dpto. Ingeniería de Sistemas y Automática, Instituto de Investigación Biomédica de Málaga (IBIMA), Universidad de Málaga, Campus de Teatinos, 29071, Málaga
{*jotaraul, jgmonroy, famoreno, javiergonzalez*}*@uma.es*

Resumen

El reconocimiento de objetos es algo innato en el ser humano. Cuando las personas miramos una fotografía, somos capaces de detectar sin esfuerzo elementos como animales, señales, objetos de interés, etc. En el campo de la visión por computador este proceso se lleva a cabo mediante herramientas de Inteligencia Artificial, con el fin de obtener información sobre el contenido de una imagen. Esta tarea, aunque ampliamente investigada, aún sigue siendo un campo de estudio activo debido a los grandes retos que conlleva: la detección de objetos en distintas condiciones luminosas, con posibles oclusiones, distintos tamaños y perspectivas, etc. Este artículo describe las tareas a completar en el desarrollo de un sistema de reconocimiento de objetos exitoso, y proporciona al lector una serie de directrices prácticas sobre como realizarlas. El trabajo viene acompañado de una serie de scripts Python para experimentar con las diferentes técnicas descritas, pretendiendo servir de apoyo en tareas docentes o de iniciación a cualquier entusiasta en la materia.

Palabras clave: reconocimiento de objetos, inteligencia artificial, aprendizaje automático, *Python*

1. INTRODUCCIÓN

El objetivo de un sistema de reconocimiento de objetos es identificar elementos en una escena como pertenecientes a ciertas categorías (*e.g.* mesa, silla, tazón, plato, etc.) a partir de una imagen de la misma. Las personas realizamos esta tarea de manera innata, pero resulta muy compleja su definición algorítmica para poder ser implementada con éxito en un computador [11]. Dentro de la Visión por Computador [6], el Aprendizaje Automático o Artificial (*Machine Learning*, *ML*) es uno de los enfoques más empleados a la hora de abordar el diseño de estos sistemas [2]. Los métodos basados en *ML* tratan de, sobre la base de una serie de datos de entrenamiento, ajustar sus parámetros internos para reconocer exitosamente nuevos objetos.

Son diversas las técnicas y enfoques que han sido propuestos a la hora de modelar los objetos a reconocer. Entre los más populares se encuentran los basados en características extraídas de los propios objetos, aunque también los hay que emplean su apariencia, conocimiento sobre ellos (información semántica), o incluso modelos geométricos (*e.g.* modelos *CAD*). Cabe señalar que, actualmente, las técnicas de reconocimiento basadas en Aprendizaje Profundo (*Deep Learning*) están consiguiendo grandes resultados (*e.g.* [7, 13]). No obstante, estas técnicas requieren de un *hardware* con altas prestaciones y gran cantidad de datos de entrenamiento, los cuales no están siempre disponibles [4]. Es por ello que en este artículo nos vamos a centrar en los sistemas de *ML* basados en características, entre los que destacan los que emplean descriptores de ciertos puntos de interés [22], como es el caso de HOG [5], SIFT [9] o SURF [8], o los que describen geométrica (tamaño, orientación, forma, etc.) y visualmente (color) la región de la imagen correspondiente al objeto en sí [1].

Previa a su etapa de funcionamiento, estos sistemas necesitan completar una fase de diseño. La Fig. 1 ilustra los pasos necesarios para ello, incluyendo: un análisis del conjunto de datos con el que se cuenta, una división de este en datos de entrenamiento y de evaluación, un preprocesamiento de los datos donde los objetos son caracterizados, el ajuste del método en sí, y la evaluación del mismo. Dada la popularidad de estas técnicas, actualmente hay disponibles multitud de implementaciones que permiten manipular y testear métodos de *ML* para el reconocimiento de objetos [3, 12, 15, 17, 19]. Desafortunadamente, lo que no es tan frecuente es encontrar soluciones simples y bien documentadas para completar los pasos necesarios en el diseño de un sistema exitoso.

En este trabajo se proporcionan una serie de directrices para completar con éxito la fase de diseño de un sistema de reconocimiento de objetos basado en características. La finalidad del mismo es eminentemente práctica, por lo que una serie de *scripts Python*, disponibles en `https://github.com/jotaraul/object_recognition_in_python`, complementan las des-

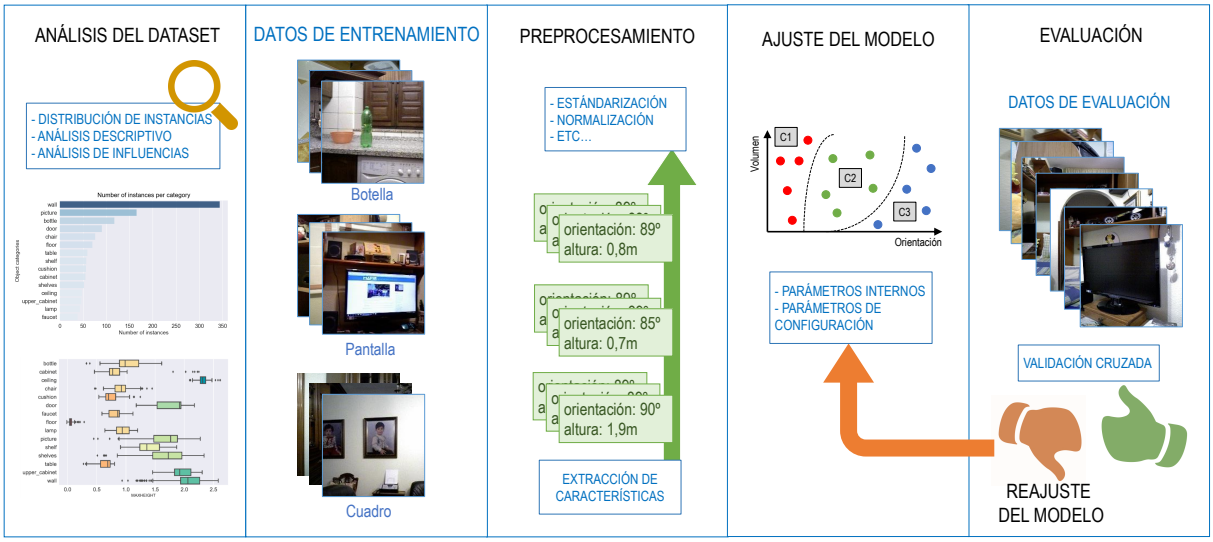

Figura 1: Pasos típicos a seguir en la etapa de diseño de un sistema de reconocimiento de objetos.

cripciones y análisis contenidos en este artículo. Concretamente, los scripts proporcionados se basan en las siguientes librerías: *Pandas* para la carga y el procesamiento de conjuntos de datos [10], *Seaborn* para las representaciones gráficas [20], y *Scikit-learn* para el ajuste de métodos de reconocimiento (*e.g.* árboles de decisión, máquina de vectores de soporte, regresión logística, k-vecinos, clasificador bayesiano ingenuo, etc.) [12].

Para ejemplificar las tareas realizadas en cada paso del diseño se emplea el conjunto de datos (o *dataset*) *Robot@Home* [14]. Este *dataset* contiene imágenes RGB-D [18], que proveen información tanto de intensidad como de profundidad de la escena, capturadas por un robot móvil en distintas casas. *Robot@Home* provee etiquetas asignando a regiones de estas imágenes la categoría de objeto a la que pertenecen [16], y además, facilita una descripción de cada uno de estos objetos que incluye características tanto geométricas como visuales.

En el resto del artículo, la Sec. 2 introduce brevemente las librerías *Python* empleadas[1]. La Sec. 3 describe buenas prácticas a la hora de manejar y analizar el conjunto de datos con el que se cuente, mientras que la Sec. 4 comenta cómo describir los objetos y preparar estas descripciones para ser empleadas en el ajuste del método de reconocimiento. La Sec. 5 describe cómo ajustar, evaluar y validar métodos empleando distintas técnicas y métricas, mientras que la Sec. 6 cierra el artículo con una breve discusión sobre el alcance del mismo.

[1]Con el fin de no dificultar la lectura, las librerías *Python* y componentes/funciones empleadas sólo se comentan en puntos clave del artículo. Esta información está disponible en los *scripts* facilitados, los cuales, para una fácil interpretación, siguen la misma estructura que la de los pasos aquí descritos.

2. LIBRERÍAS PYTHON

Pandas. Esta librería pretende facilitar el trabajo con conjuntos de datos y proveer los componentes necesarios para implementar modelos estadísticos. Entre sus utilidades permite: leer y escribir datos entre estructuras de datos cargadas en memoria y ficheros con distintos formatos (CSV y ficheros de texto, Microsoft Excel, SQL databases, y HDF5), y reestructurar, pivotar, o mezclar conjuntos de datos, entre otras. Con el objetivo de ser rápida y eficiente, está optimizada para maximizar su rendimiento con partes críticas implementadas en Cython o C. *Pandas* también provee modelos estadísticos de regresión lineal, pero como los mismos desarrolladores comentan (última página en [10]), su idea es trabajar estrechamente con los homónimos de *Scikit-learn* para aunar esfuerzos y proveer métodos cohesionados a la comunidad.

Seaborn. Esta librería escrita en *Python* y basada en *matplotlib*, provee una interfaz de alto nivel para mostrar gráficas estadísticas atractivas visualmente. Una de sus características más relevantes es que incluye soporte para estructuras de datos de *Pandas*, por lo que pueden usarse en el mismo código sin necesidad de conversión de tipos de datos. El objetivo de *Seaborn* es hacer de la visualización una parte central en la exploración y comprensión de los datos. Algunas de las funcionalidades que ofrece son: varios temas y paletas de colores para crear representaciones gráficas vistosas, funciones para visualizar y comparar distribuciones de una o dos variables, herramientas para ajustar y mostrar modelos de regresión lineal, o funciones para visualizar matrices de datos.

Scikit-learn. Posiblemente la librería *Python* más popular de *ML*. Esta engloba una gran variedad de algoritmos para problemas tanto supervisados (donde los datos vienen con los atributos que se quieren predecir, *e.g.* las categorías de los objetos) como no supervisados (datos sin dichos atributos). Su objetivo principal es hacer llegar estos algoritmos a personas no expertas en la materia a través de un lenguaje de alto nivel y de propósito general como es *Python*. Para ello ponen énfasis en su facilidad de uso, alto rendimiento, buena documentación, y consistencia de sus APIs. Entre otras funcionalidades permite entrenar métodos de predicción, convertirlos en modelos persistentes en disco, y evaluarlos empleando distintas técnicas y métricas.

3. MANEJO Y ANÁLISIS DEL CONJUNTO DE DATOS

El primer paso en el diseño de un sistema de reconocimiento es el estudio del conjunto de datos de partida (ver Fig. 1). Un conjunto de datos útil en este punto ha de contener instancias de objetos pertenecientes a las categorías que se quieran reconocer. Estas instancias vienen principalmente en forma de observaciones (información sensorial) de los objetos, y/o características extraídas de los mismos.

El análisis del *dataset* es un paso fundamental, permitiendo comprender mejor la información que este contiene, diseñar una estrategia para atacar el problema con más posibilidades de éxito, e incluso detectar limitaciones sobre las conclusiones que podremos alcanzar al evaluar el método [21]. Concretamente, se pretende obtener información sobre: la distribución de las instancias de cada categoría de objetos (balance), el comportamiento de las características empleadas para describirlos, y la capacidad discriminatoria de estas. La realización de este análisis puede ahorrar tiempo de desarrollo al detectar anticipadamente posibles errores que producirían un ajuste erróneo del método. En este punto se ha empleado la librería *Pandas*, especialmente indicada para la carga de datos de gran magnitud y la obtención de esta información básica de manera eficiente.

3.1. Distribución de Instancias

El objetivo de este análisis es determinar el balance existente entre el número de instancias y categorías de objetos en el conjunto de datos. Para el caso de *Robot@Home*, los resultados del análisis arrojan 2,309 instancias de objetos pertenecientes a 180 categorías, aunque este número no es real si se usa con el fin de reconocer objetos, ya que aparecen categorías con pocas instancias (o incluso sólo una). Este es el caso por ejemplo de las clases *fax*, *radiador*, *yogur*, o *balón de yoga*. Además, como se intuye de ese resultado y tal y como muestra la Fig. 2, el conjunto de datos no está balanceado, hecho que habrá que tener en cuenta durante el ajuste del método ya que, según el elegido, su rendimiento puede verse afectado. No obstante, si fuera el caso, el conjunto de datos puede balancearse para solventar este problema.

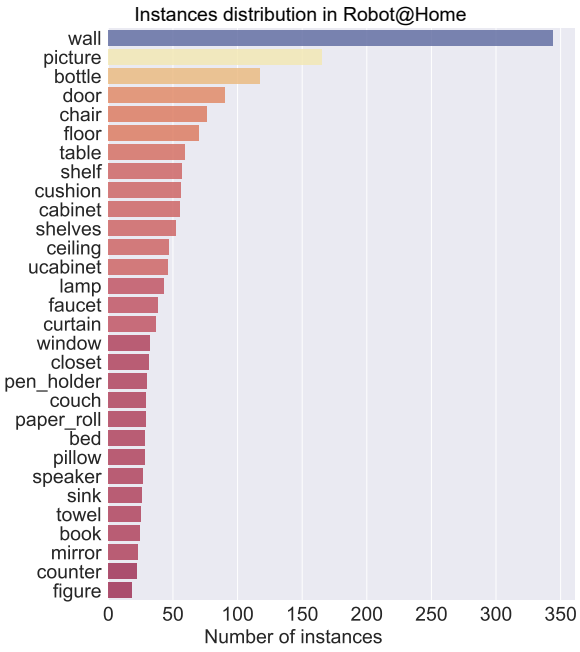

Figura 2: Número de instancias de las 30 categorías de objetos que más se repiten en *Robot@Home*

3.2. Análisis Descriptivo

Este análisis busca describir las diferentes características a utilizar para categorizar a los objetos, permitiendo comprender la tendencia central, dispersión y forma de sus distribuciones, validar inicialmente su uso, y detectar comportamientos erróneos (típicamente provenientes de fallos en la implementación).

La Tab. 1 muestra la descripción básica de algunas características para el caso de *Robot@Home* (computada por *Pandas*). El tipo de comprobaciones que se pueden hacer son, por ejemplo, que la altura mínima (altura de la parte del objeto más cercana al suelo) es de media menor (lógicamente) que la máxima, que en el conjunto de datos hay más objetos con orientación vertical que horizontal (por su valor medio), o como la distribución del volumen de los objetos está positivamente sesgada con un predominio de objetos *pequeños* (según los percentiles). También se puede apreciar como

Cuadro 1: Descripción básica de la tendencia central, dispersión y forma de las distribuciones de 6 características en *Robot@Home* .

Característica	Altura min.	Altura max.	Orientación	Volumen	Planaridad	Tono
Media	$0.54m.$	$1.44m.$	66.15^o	$0.80m^3.$	0.21	117.37
Desv. típtica	$0.59m.$	$0.66m.$	32.33^o	$1.57m^3.$	0.12	54.50
Mínimo	-0.50$m.$	-0.01$m.$	0.01^o	$0.00m^3.$	0.00	24.65
25 %	$0.00m.$	$0.87m.$	57.21^o	$0.03m^3.$	0.11	75.20
50 %	$0.41m.$	$1.54m.$	84.76^o	$0.23m^3.$	0.19	108.95
75 %	$0.95m.$	$2.00m.$	88.36^o	$0.83m^3.$	0.30	154.26
Máximo	$2.37m.$	$2.61m.$	89.99^o	$13.16m^3.$	0.49	299.10

según los valores mínimos y máximos del tono (o matiz) de los objetos, el rojo puro no aparece en los datos (se corresponde con un tono igual a 0). Este tipo de información nos permite comprobar que los valores de las variables dadas/computadas tienen sentido, así como aprender algo más sobre los objetos observados.

Para cada categoría en el *dataset* también puede mostrarse de forma visual la distribución de estas características. Por ejemplo, la Fig. 3 muestra, empleando herramientas de *Seaborn* , las distribuciones de las características planaridad (arriba) y altura del centroide (abajo). En las gráficas, las cajas coloreadas muestran donde se encuentra el 50 % de los datos, mientras que sus límites inferior y superior se corresponden con el percentil 25 y 75, respectivamente. La linea interior vertical señala la media, mientras que las rayas exteriores verticales señalan mediciones alejadas hasta 1.5 veces la distancia entre los percentiles 25 y 75. Por último, las mediciones más alejadas de estas se representan como puntos y se consideran valores anómalos u *outliers*. Intuitivamente, para que una característica permita discriminar entre categorías, esta debe tomar valores concentrados (poca dispersión) y diferentes para cada una de ellas. Podemos ver como en el caso de la planaridad la dispersión es considerable, y los valores que toman no muestran una gran diferencia en función de las categorías. Podemos ver como el caso de la altura del centroide es opuesto, con una dispersión bastante menor, y una mayor diferenciación entre clases. Esto indica que, a priori, la segunda característica tiene un mayor poder discriminatorio y ayudará en mayor medida en el reconocimiento.

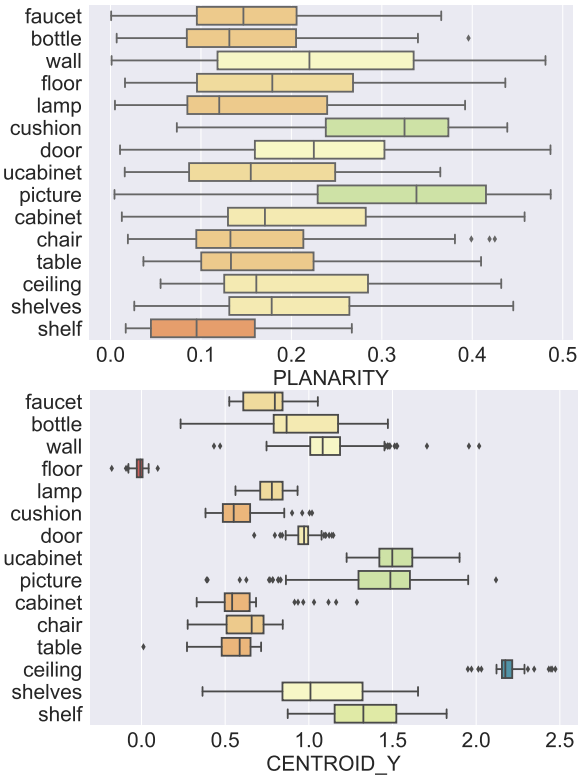

Figura 3: Distribución de las características *PLANARITY* (planaridad) y *CENTROID_Y* (altura del centroide desde el suelo) según la categoría de los objetos.

3.3. Análisis de Influencia

Este análisis realiza una estimación del poder de discriminación de las características con respecto a las categorías de los objetos, esto es, como de prometedoras son a la hora de discernir las distintas categorías con su uso. Este estudio se hace de manera individual, ya que será el método de reconocimiento el encargado de combinarlas. Un método simple y efectivo es realizar a cada característica un test *chi-cuadrado* (χ^2), el cual hace la hipótesis de que no existe relación entre una característica y las categorías de los objetos, construye un modelo como si tal relación no existiera, y devuelve una medición del error obtenido al comparar el modelo con los datos reales. La ejecución del test también devuelve un *p-valor* que, de tomar valores bajos, valida estadísticamente su resultado. De este modo, para que una característica tenga poder discriminatorio interesa que el test *chi-cuadrado* devuelva un resultado alto con un *p-valor* reducido. Cabe mencionar que para poder realizar este test las características han de *binarizarse*. También se puede realizar con características tomando más de

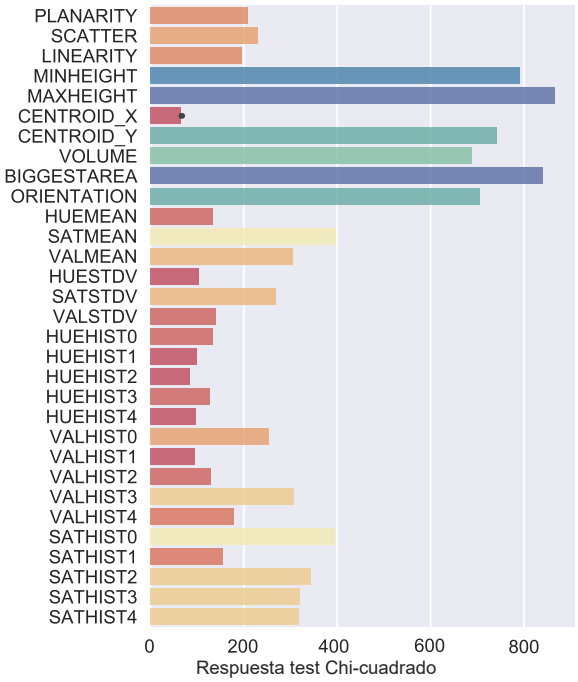

Figura 4: Resultados del test *chi-cuadrado* para todas las características en *Robot@Home* .

dos valores, pero esto requiere una etapa posterior de chequeo o *post-hoc*, por lo que en este trabajo vamos a mostrar la versión más básica.

La Fig. 4 muestra los resultados de este test para todas las características en *Robot@Home* . Podemos ver como hay 6 características que obtienen un valor claramente mayor que el resto (color azul/verde), rechazando fuertemente la hipótesis de que no hay relación entre ellas y las categorías de los objetos. Tras estas, hay otras variables (color naranja) con un valor también alto para este estadístico, mientras que las restantes (en tonos rojo) son poco prometedoras. En todos los casos los *p-valores* asociados son bastante inferiores a 0.05, valor umbral considerado típicamente para dar validez estadística a dicho rechazo.

4. PREPROCESAMIENTO

En la sección anterior se ha completado un primer estudio del poder discriminatorio de las características describiendo los objetos. Este paso se ha podido realizar directamente ya que *Robot@Home* proporciona, entre otra información, los objetos ya descritos conforme a las características que aparecen en la Fig. 4. Si el conjunto de datos con el que se trabaja no facilitara esta información, o el resultado del estudio concluyera que las características no son apropiadas, habría que completar los datos con un conjunto de características variado y realizar de nuevo los análisis. En este sentido, y tal y como se comprueba en la sección de evaluación, el conjunto de características de *Robot@Home* es bastante completo, ya que describe el tamaño, la forma, la posición espacial y la apariencia de los objetos.

Como se ha mencionado, los objetos pueden describirse a partir de características de distinta naturaleza, incluyendo características HOG, SIFT, o SURF, o incluso las de Haar. En algunos de estos estos casos los análisis anteriores no serían aplicables, mientras que en otros tendrían que ser adaptados para que tengan sentido y produzcan resultados válidos.

En cualquier caso, los métodos de aprendizaje automático suelen beneficiarse si las características son pre-procesadas para estandarizar su valores, esto es, que sigan una distribución Gaussiana de media cero y desviación típica uno. Esto se debe a la forma en la que son entrenados, donde si una característica tiene una varianza superior a la de otra, esta podría dominar el ajuste del modelo y comprometer su validez. *Scikit-learn* incluye distintas técnicas de estandarización que incluso permiten trabajar con datos dispersos o con valores anómalos. Esta librería también permite, mediante herramientas de su módulo *preprocesing*, realizar transformaciones no lineales de las características, normalizar los datos (escalar cada descripción para que tenga norma unitaria), binarizar, o incluso generar características polinomiales para añadir complejidad al modelo.

5. AJUSTE Y EVALUACIÓN DEL MÉTODO

Una vez los descriptores de los objetos han sido procesados, ya están listos para participar en la fase de ajuste del modelo (ver Fig. 1). En esta fase, los parámetros internos del método se ajustan en función de los datos de entrenamiento de tal manera que, dada una nueva descripción de un objeto, sea capaz de inferir su categoría con éxito.

Métricas. Ya ajustado el método, los datos de evaluación se emplean para comprobar si el ajuste de sus parámetros es correcto. Esta comprobación se hace en función a cierta métrica, o un conjunto de estas, y se considera que el método es válido cuando cumple con los requisitos de éxito para la aplicación concreta en la que quiera emplearse. Las métricas más populares son exactitud (*accuracy*), ratio entre el número de objetos correctamente reconocidos y el número de objetos en el *dataset*, precisión (*precision*), que expresa la habilidad del método para no etiquetar un objeto como perteneciente a una categoría que no es realmente la suya, y exhaustividad (*recall*) que mide la capacidad para reconocer como perteneciente a

una categoría un objeto que realmente pertenece a ella.

Estas métricas pueden calcularse a nivel de instancia (objeto), con lo cual suelen llevar el prefijo *micro*, o a nivel de categoría de objeto, *macro*. También hay métricas que combinan las anteriores, como es el caso del *F1 score*, un promedio ponderado de la exhaustividad y la precisión. La métrica a emplear va a depender de los requisitos que deba cumplir el sistema.

Validación cruzada. Por otro lado, cómo dividir el conjunto de datos en subconjuntos de entrenamiento y de evaluación no es trivial y, si no se hace correctamente, la validez de las conclusiones sobre el éxito del método puede verse comprometida. Para ello suelen emplearse técnicas de *validación cruzada*. Estas técnicas persiguen la validación de un modelo mediante un análisis estadístico que refuerza los resultados obtenidos dándoles un cierto grado de generalidad, limitando la posibilidad de obtener un buen resultado debido simplemente al azar en la división de los datos en *entrenamiento* y *evaluación*. Básicamente, consiste en realizar una iteración donde el conjunto de datos se divide en dos grupos, empleando uno de ellos para el ajuste y el otro para la evaluación. Para reducir la variabilidad del resultado obtenido, esta iteración vuelve a realizarse usando distintos grupos, y los resultados de validación son combinados para estimar el rendimiento predictivo del método.

Existen distintas técnicas de validación cruzada que definen distintas maneras en las que dividir los datos, y consideran un distinto número de iteraciones a realizar. En conjuntos de datos donde no es factible comprobar todas las divisiones posibles, una de las técnicas más usadas es la de k-grupos, donde en cada iteración se dividen aleatoriamente los datos en k grupos distintos del mismo tamaño, empleando uno de ellos para evaluar el método y el resto para ajustarlo. El éxito del método se obtiene tras realizar un número elevado de iteraciones (cuanto más alto, más seguros estaremos de que el resultado es válido) y promediar los resultados obtenidos en cada una de ellas.

Métodos. *Scikit-learn* ofrece una gran variedad de métodos de clasificación que se pueden emplear para el reconocimiento de objetos. Entre ellos se encuentran: árboles de decisión, bosques aleatorios, árboles-extra, máquina de vectores de soporte, regresión logística, k-vecinos, o el clasificador bayesiano ingenuo. Este trabajo no persigue evaluar concienzudamente cada uno de ellos, pero sí se van a emplear para ilustrar ciertos hechos. Por ejemplo, la Fig. 5 muestra cómo varía la métrica *exactitud* en función del número de iteraciones de validación cruzada completadas (datos obtenidos

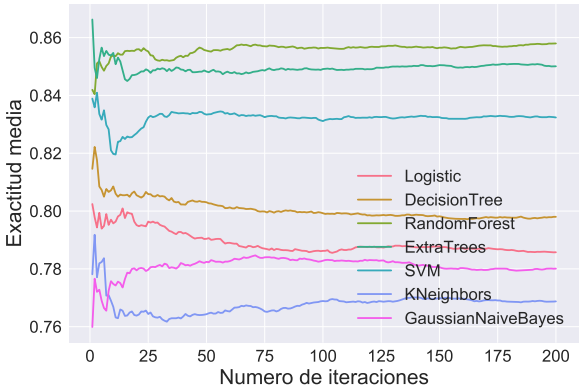

Figura 5: Exactitud de una serie de métodos para el reconocimiento según el número de iteraciones de validación cruzada realizadas.

empleando las 15 categorías de objetos con mayor número de instancias y todas las características disponibles tras estandarizarlas). Esto es, el eje vertical muestra la media de esta métrica para las iteraciones completadas hasta el momento. Se puede ver cómo, con un número bajo de iteraciones, el resultado para un mismo método oscila considerablemente, por lo que no se puede considerar fiable. Una gráfica de este tipo puede mostrarnos el número de iteraciones necesario para que la métrica *se estabilice* y sea realmente informativa. En el caso que nos ocupa, podemos ver como a partir de 100 iteraciones los valores se mantienen estables para la mayoría de los métodos.

Cada uno de los métodos, aparte de sus parámetros internos a ajustar durante el entrenamiento, también cuenta con una serie de parámetros de configuración que influyen directamente a su rendimiento, y que se ajustan en función de las peculiaridades de la aplicación. Para el ajuste de estos parámetros también se puede utilizar validación cruzada, añadiendo los parámetros de configuración a ajustar a las variables *número de iteraciones* y *número de grupos*.

Matriz de confusión. Una forma gráfica de comprobar distintos aspectos del rendimiento de un método es usar la llamada *matriz de confusión*. Esta matriz indexa en sus filas las categorías de los objetos en el conjunto de datos de evaluación, mientras que las columnas indexan las categorías a las que han sido asignados por el método. Así, cuanto mayor concentración haya en la diagonal de la matriz, más exitoso será. Esta matriz también nos permite observar para qué categorías de objetos el modelo tiene un rendimiento inferior, de cara a poder reajustar la fase de diseño para solventarlo. La Fig. 6 muestra la matriz de confusión usando el método de bosques aleatorios, construida acumulando los resultados reportados por este en 200 iteraciones de validación cruzada (para

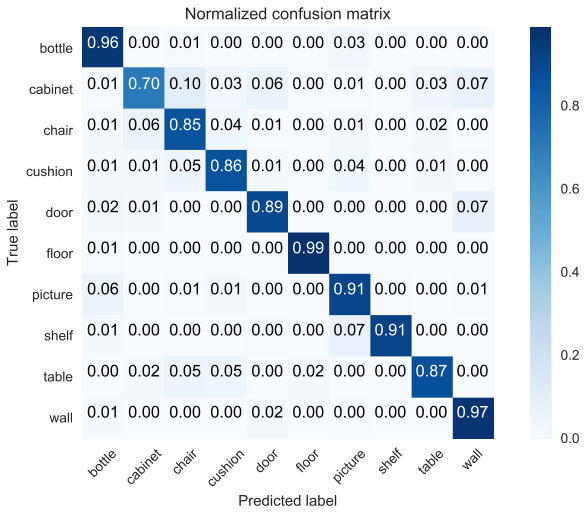

Figura 6: Matriz de confusión para las 10 categorías con mayor número de instancias empleando *bosques aleatorios*.

mejorar la visibilidad, se muestran sólo las 10 categorías con más instancias). Podemos ver cómo el comportamiento del método en general es bastante bueno (valores próximos a 1 en la diagonal), y que la categoría más *conflictiva* es *armario*. La categoría con la que más lo confunde es *silla*, seguida de *pared*, por lo que si se quiere mejorar estos resultados sería necesario añadir alguna característica que permita discriminar mejor entre estas categorías.

Características. En la Sec. 3 se analizó el poder discriminatorio de las características en *Robot@Home* . El uso de características poco discriminatorias afecta de distinta forma a los métodos: algunos son capaces de aislarlas y darles menos peso, no afectando a su rendimiento, mientras que otros sí sufren con su uso. La Fig. 7 muestra la evolución de la exactitud de los métodos empleados en este trabajo dependiendo del número de características usadas (empleando de nuevo 200 iteraciones de validación cruzada). Este número siempre se refiere a las características más prometedoras según se mostró en la Fig. 4. Otro factor a tener en cuenta es el del tiempo empleado en el ajuste de los modelos, el cual puede ser crítico según la aplicación. Por ejemplo, mientras que el ajuste empleando 6 características requiere en total 32*seg.*, usando las 32 asciende a 72*seg.* (datos tomados con un procesador Intel(R) Core(TM) i7-3820 CPU @ 3.60GHz y 16GB de memoria RAM).

6. DISCUSIÓN

Este trabajo ha proporcionado una serie de directrices para el diseño de métodos basados en aprendizaje automático capaces de reconocer objetos. El texto lo completan *scripts Python*, disponibles en `https://github.com/jotaraul/object_recognition_in_python`, que permiten probar de manera práctica cada uno de los pasos descritos para completar la fase de diseño. Empleando herramientas de las librerías *Pandas* , *Seaborn* y *Scikit-learn* se puede analizar un conjunto de datos, dividirlo entre datos de entrenamiento y evaluación, describir los objetos en él para el ajuste del modelo de reconocimiento, ajustarlo, evaluarlo y validarlo, sirviendo este trabajo de hilo conductor para el diseño de métodos exitosos. Pensamos que el artículo puede ser útil en tareas docentes, así como para cualquier entusiasta de la materia que quiera construir su propio reconocedor.

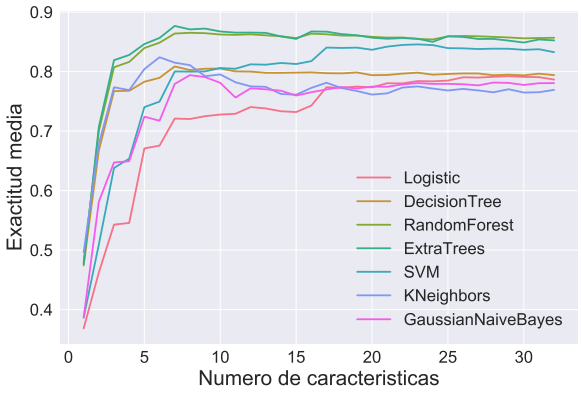

Figura 7: Exactitud de los métodos dependiendo del número de características empleadas.

Agradecimientos

Este trabajo se ha desarrollado en el marco del proyecto WISER (DPI2017-84827-R), financiado por el Ministerio de Ciencia e Innovación contando con fondos del Fondo Europeo de Desarrollo Regional (FEDER).

English summary

A TUTORIAL ON OBJECT RECOGINITION USING PYTHON TOOLS

Abstract

When people look at pictures, we are able to effortlessly detect elements like animals, signals, objects of interest, etc. Object recognition is a computer vision technique that, through tools from Artificial Intelligence, aims to carry out such an action with the goal of retrieving information from the content of an image. This task, although widely investigated, is an active research fied due to the challenges that it has to face: the detection of objects in different lighthing conditions, with possible occlusions, different sizes and perspectives, etc. This paper describes the typical actions to be completed in the development of a successful object recognition system, and provides the reader a

number of practical guidelinesabout how to address them. The work is accompanied by a Python script (available at `https://github.com/jotaraul/object_recognition_in_python`*) to put them into practice using state-of-the-art techniques (from libraries like Pandas or Scikit-learn), and aims to be a valuable resource for education, or as initiation to any enthusiastic in the topic.*

Keywords: object recognition, artificial intelligence, machine learning, *Python*.

Referencias

[1] A. Anand, H. S. Koppula, T. Joachims, and A. Saxena. Contextually guided semantic labeling and search for three-dimensional point clouds. *In the International Journal of Robotics Research*, 32(1):19–34, January 2013.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[3] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[4] Daniel Camilleri and Tony Prescott. Analysing the limitations of deep learning for developmental robotics. In *Conference on Biomimetic and Biohybrid Systems*, pages 86–94. Springer, 2017.

[5] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[6] Javier Gonzalez-Jimenez. *Visión por Computador (in Spanish)*. Thomson Paraninfo, 1999.

[7] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask r-cnn. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[8] Jan Knopp, Mukta Prasad, Geert Willems, Radu Timofte, and Luc Van Gool. Hough transform and 3d surf for robust three dimensional classification. In *Proceedings of the 11th European Conference on Computer Vision: Part VI*, ECCV'10, pages 589–602, 2010.

[9] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision*, 60(2):91–110, November 2004.

[10] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proc. of the 9th Python in Science Conference*, pages 51 – 56, 2010.

[11] M. Oliveira, L. Seabra Lopes, G. H. Lim, S. H. Kasaei, A. D. Sappa, and A. M. Tomé. Concurrent learning of visual codebooks and object categories in open-ended domains. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 2488–2495, Sept 2015.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, November 2011.

[13] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, July 2017.

[14] J. R. Ruiz-Sarmiento, C. Galindo, and J. González-Jiménez. Robot@home, a robotic dataset for semantic mapping of home environments. *The International Journal of Robotics Research*, 36(2):131–141, 2017.

[15] J. R. Ruiz-Sarmiento, C. Galindo, and J. González-Jiménez. A survey on learning approaches for probabilistic graphical models. application to scene object recognition. *International Journal of Approximate Reasoning*, 83(C):434–451, April 2017.

[16] J. R. Ruiz-Sarmiento, Cipriano Galindo, and Javier González-Jiménez. OLT: A Toolkit for Object Labeling Applied to Robotic RGB-D Datasets. In *European Conf. on Mobile Robots*, 2015.

[17] J.R. Ruiz-Sarmiento, C. Galindo, and J. Gonzalez-Jimenez. Modelado del contexto geométrico para el reconocimiento de objetos. *XXXVIII Jornadas de Automática*, 2017.

[18] J.R. Ruiz-Sarmiento, C. Galindo, J. Gonzalez-Jimenez, and J.L. Blanco. Navegación reactiva de un robot móvil usando kinect. *Actas ROBOT 2011*, 2011.

[19] J.R. Ruiz-Sarmiento, F.A. Moreno, J. Monroy, and J. Gonzalez-Jimenez. mVision, a Toolbox for Computer Vision Courses. In *The 12th annual International Technology, Education and Development Conference (INTED2018)*, 2018.

[20] seaborn: statistical data visualization. `https://seaborn.pydata.org/`. [Online; accessed 14-June-2018].

[21] Hoerl Roger W., Snee Ronald D., and De Veaux Richard D. Applying statistical thinking to Big Data problems. *Wiley Interdisciplinary Reviews: Computational Statistics*, 6(4):222–232, 2014.

[22] Jianguo Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. In *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)*, pages 13–13, June 2006.