Universidad Austral

Proyecto 2: clasificación con machine learning

12/8/22

Alejandro Silvestri

Visión artificial

Contenido

Introducción

El [proyecto anterior](#) se ocupaba de la detección de contornos, delegando la clasificación a la función **`matchShapes`**. El presente proyecto continúa el anterior, eliminando la función `matchShapes` y realizando una clasificación basada en aprendizaje automático (machine learning).

A diferencia de otros proyectos, éste consta de 3 aplicaciones Python que se ejecutan por separado:

1. generador de descriptores
2. entrenador
3. clasificador

El generador de descriptores se usará para construir el dataset. El entrenador consumirá ese dataset y producirá un modelo entrenado de clasificación. El clasificador usará el modelo entrenado para predicción.

Proyecto

Los alumnos elegirán al menos tres objetos reconocibles por su contorno, y elaborarán un dataset de muestras y sus correspondientes etiquetas. Cada muestra consiste en los siete invariantes de Hu como mínimo; los alumnos podrán agregar otras características.

Entrenarán un algoritmo de aprendizaje automático (usualmente un clasificador basado en *decision trees*) sobre el dataset de descriptores y guardarán en un archivo el modelo resultante.

Un sistema final, similar al proyecto anterior, detectará los objetos en la imagen de la webcam en tiempo real y los clasificará según las predicciones del modelo entrenado en lugar de usar *matchshapes*.

A continuación se sugieren maneras de encarar el generador de descriptores, el entrenador y el clasificador. Los alumnos que lo deseen pueden hacerlo de otras maneras.

Generador de descriptores

Conviene comenzar definiendo un diccionario de etiquetas, usualmente una tabla numerada de 1 en adelante (la etiqueta), con su descripción (la forma), por ejemplo:

1. cuadrado
2. triángulo
3. estrella

En adelante, las tablas y el código manejarán el número de la etiqueta en lugar de su descripción.

El dataset a construir consiste en una serie de muestras, cada una con una etiqueta. En este caso las muestras son los 7 valores numéricos de los invariantes de Hu correspondientes a un contorno, y la etiqueta es un valor entero asignado a esa forma. Los invariantes de Hu de una muestra tienen esta forma:

```
[  6.53608067e-04,   6.07480284e-16,  9.67218398e-18, 1.40311655e-19,
  -1.18450102e-37,   8.60883492e-28, -1.12639633e-37  ]
```

Un dataset de invariantes de Hu es un array de dos dimensiones, con una fila para cada contorno analizado y 7 columnas para sendos invariantes de Hu. El ejemplo ilustra un dataset de dos muestras:

```
[
 [ 6.53608067e-04,   6.07480284e-16,  9.67218398e-18, 1.40311655e-19,
  -1.18450102e-37,   8.60883492e-28, -1.12639633e-37  ],
 [ 6.53608067e-04,   6.07480284e-16,  9.67218398e-18, 1.40311655e-19,
  -1.18450102e-37,   8.60883492e-28, -1.12639633e-37  ]
]
```

El generador de descriptores es una aplicación similar al proyecto 1, que en lugar de clasificar sólo detecta el contorno, y cuando el usuario pulsa la tecla espacio imprime en terminal sus invariantes de Hu.

Con esta aplicación se extraen invariantes de Hu de múltiples imágenes diferentes con la misma forma. Por ejemplo se puede presentar un mismo cuadrado en diferentes posiciones, rotaciones y escalas, y se puede complementar con otros cuadrados ligeramente diferentes (por ejemplo dibujados a mano alzada).

Luego de evaluar todas las imágenes, se copian los invariantes de Hu desde la terminal y se pegan en una hoja de cálculo, y se le agrega una columna con la etiqueta correspondiente. El procedimiento se repite para cada forma que se quiera distinguir. Una buena idea de cara a la etapa siguiente es mezclar aleatoriamente las filas.

Entrenador

Esta aplicación entrena la máquina y genera el modelo para inferencia. Opera exclusivamente sobre el dataset, que puede estar guardado en archivos con algún formato adecuado para ser consumido por el algoritmo de machine learning, o bien se puede pegar directamente en el código con forma de array de dos dimensiones.

El código puede ser como sigue, requiere instalar scikit-learn:

```
from sklearn import tree
```

```
from joblib import dump, load

# dataset
X = [
 [6.53608067e-04, 6.07480284e-16, 9.67218398e-18, 1.40311655e-19, -1.18450102e-37, 8.60883492e-28, -1.12639633e-37],
 [6.07480284e-16, 9.67218398e-18, 1.40311655e-19, -1.18450102e-37, 8.60883492e-28, -1.12639633e-37, 6.53608067e-04],
 [6.53608067e-04, 6.07480284e-16, 9.67218398e-18, 1.40311655e-19, -1.18450102e-37, 8.60883492e-28, -1.12639633e-37],
 [1.40311655e-19, -1.18450102e-37, 8.60883492e-28, -1.12639633e-37, 6.53608067e-04, 6.07480284e-16, 9.67218398e-18],
 [8.60883492e-28, -1.12639633e-37, 6.53608067e-04, 6.07480284e-16, 9.67218398e-18, 1.40311655e-19, -1.18450102e-37],
 [6.53608067e-04, 6.07480284e-16, 9.67218398e-18, 1.40311655e-19, -1.18450102e-37, 8.60883492e-28, -1.12639633e-37],
 [9.67218398e-18, 1.40311655e-19, -1.18450102e-37, 8.60883492e-28, -1.12639633e-37, 6.53608067e-04, 6.07480284e-16],
]

# etiquetas, correspondientes a las muestras
Y = [1, 1, 1, 2, 2, 3, 3]

# entrenamiento
clasificador = tree.DecisionTreeClassifier().fit(X, Y)

# visualización del árbol de decisión resultante
tree.plot_tree(clasificador)

# guarda el modelo en un archivo
dump(clasificador, 'filename.joblib')

# en otro programa, se puede cargar el modelo guardado
clasificadorRecuperado = load('filename.joblib')
```

La aplicación entrena el modelo, y al finalizar lo guarda en un archivo para que pueda ser utilizado para predicción por otra aplicación.

Clasificador

Esta aplicación es una versión mejorada del proyecto anterior. Opera sobre la webcam. Básicamente reemplaza la clasificación con `matchShapes`, por la predicción del modelo entrenado.

Al iniciar debe cargar el modelo desde su archivo, y luego, sobre cada contorno, computar su descriptor y alimentar el modelo para predicción:

```
from joblib import load

# carga el modelo
clasificador = load('filename.joblib')

etiquetaPredicha = clasificador.predict(invariantesDeHu)
```

La interfaz de usuario puede ser idéntica a la del proyecto anterior. La etiqueta predicha es un número entero, se puede usar el diccionario para obtener la descripción de texto correspondiente.

Referencias

- 1.10. Decision Trees — scikit-learn 1.1.2 documentation
- 9. Model persistence — scikit-learn 1.1.2 documentation
- Visualize a Decision Tree in 4 Ways with Scikit-Learn and Python | MLJAR
- Installing scikit-learn