Использование библиотеки scikit-image

Януш Виктор

ВМК МГУ

1. Описание библиотеки
2. Примеры
3. В применении к машинному обучению
4. Итоги

Содержание презентации

О библиотеке

- Scikit-image — библиотека для работы с изображениями.
- В ней реализовано много готовых алгоритмов.
- Преимущество этой библиотеки в том, что ей легко пользоваться в связке с уже известными numpy/scipy.
- Саму библиотеку можно найти по адресу scikit-image.org.

Содержание презентации

Загрузка изображений

Загружать изображения с scikit-image очень просто:

```
import skimage.io as io
image = io.imread(path)
```

Поддерживаются все распространенные форматы (bmp, png, jpg, jpeg и т.д.).

Машграф с помощью scikit-image

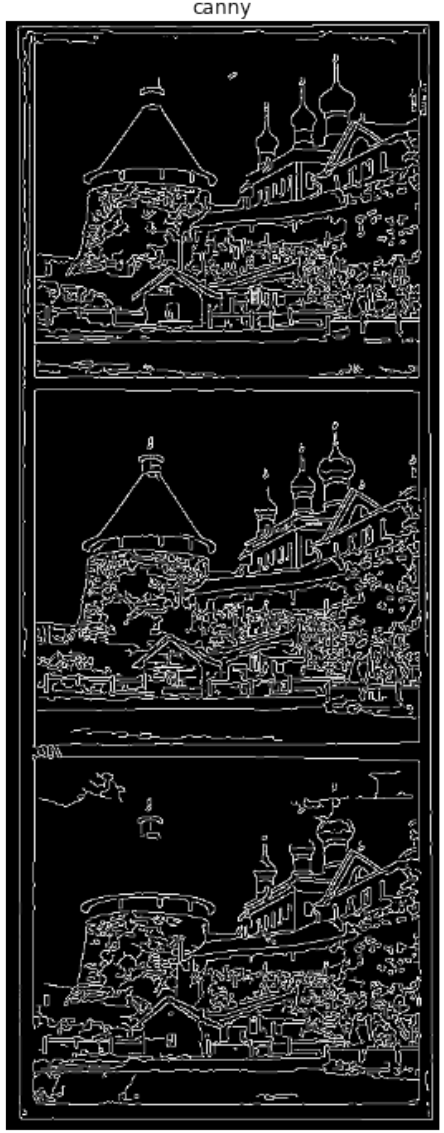
Как реализовать алгоритм canny с помощью scikit-image? Очень легко!

```
from skimage.feature import canny
borders = canny(image, low_threshold=20, high_threshold=100)
```

Пример работы canny

original

canny



Выравнивание гистограмм

В scikit-image есть определенные наборы тестовых картинок.
Попробуем применить выравнивание гистограмм к снимку поверхности Луны:

```
from skimage import data, exposure
from skimage.morphology import disk
image = data.moon() # Картинка из scikit-image
global_equalized = exposure.equalize_hist(image)
local_equalized = rank.equalize(image, selem=disk(30))
```

Пример выравнивания гистограмм

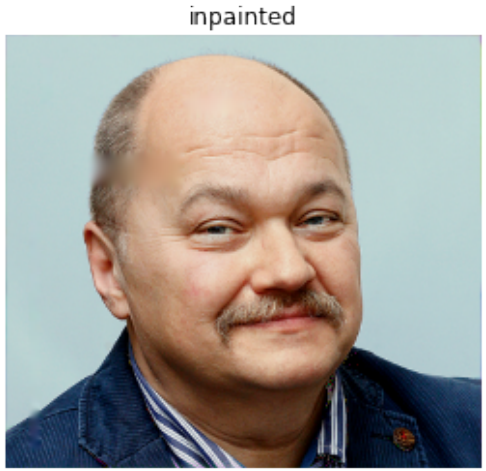
Пример Inpainting

Иногда так случается, что в изображении затираются определенные части. Процесс их восстановления — inpainting. Посмотрим как это делать в **scikit-image**:

```
image = io.imread('mash.bmp')
mask = np.zeros(image.shape[:-1])
mask[20:60, 30:40] = 1
mask[50:70, 150:200] = 1
mask[100:130, 70:150] = 1
mask[150:350, 20:30] = 1
image_with_defect = image.copy()
image_with_defect[np.where(mask)] = 0

from skimage.restoration.inpaint import inpaint_biharmonic
image_inpainted = inpaint_biharmonic(image_with_defect,
                                     mask,
                                     multichannel=True)
```

Пример Inpainting



Пример нахождения контуров

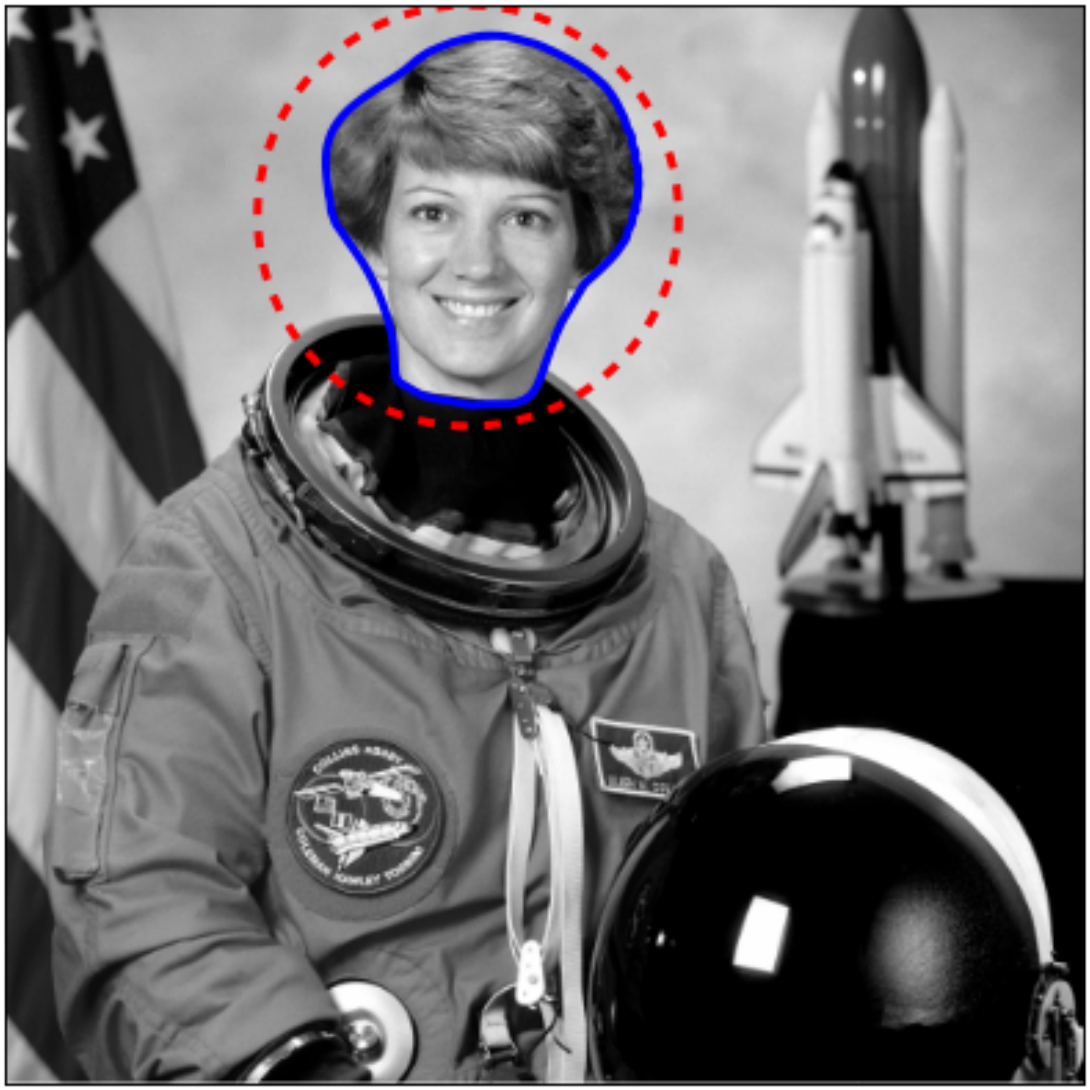
На изображениях бывают объекты сложной формы, про которые известно, где они примерно находятся (например bounding box), но нет четкого описания их формы. Тогда можно применить модель active contour.

```
from skimage.filters import gaussian
from skimage.segmentation import active_contour
im = skimage.color.rgb2gray(data.astronaut())

s = np.linspace(0, 2*np.pi, 400)
x = 220 + 100*np.cos(s)
y = 100 + 100*np.sin(s)
init = np.array([x, y]).T

im = gaussian(im, 3)
snake = active_contour(im, init, alpha=0.015,
                       beta=10, gamma=0.001)
```

Пример нахождения контуров

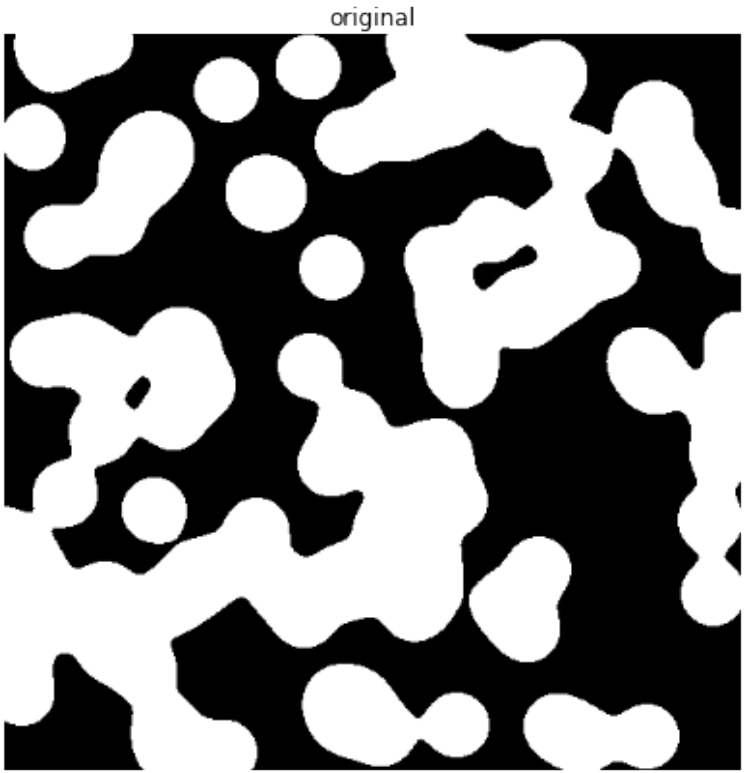


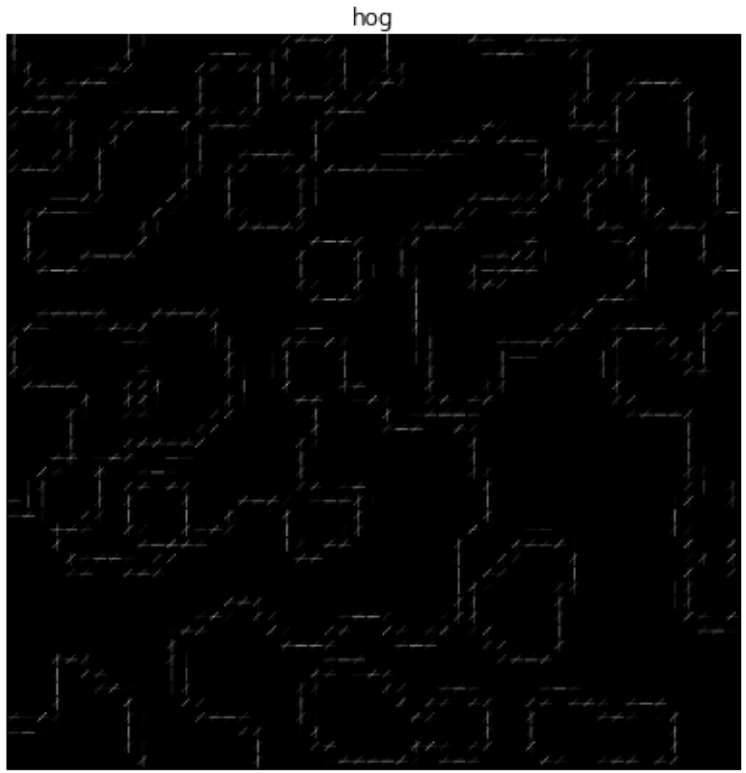
Машграф возвращается

Как реализовать нахождение гистограмм ориентированных градиентов (HOG)? Оно уже реализовано!

```
from skimage.feature import hog
im = data.binary_blobs()
features, hog_image = hog(im, orientations=8,
                          pixels_per_cell=(10, 10),
                          cells_per_block=(1, 1),
                          visualise=True)
```

Пример нахождения контуров





Содержание презентации

Классификация MNIST

- Попробуем как-то использовать библиотеку **scikit-image** для машинного обучения.
- Самый простой датасет с картинками — MNIST.
- Первое, что приходит в голову — размножение данных и преобразование признаков.
- Применим метод опорных векторов (SVM).

Загрузка данных

Загрузим данные и запустим на них SVM, никак не преобразуя их:

```
from sklearn.datasets import fetch_mldata
from sklearn.utils import shuffle
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score

mnist = fetch_mldata('MNIST original')
X_train, y_train = mnist.data[:60000], mnist.target[:60000]
X_test, y_test = mnist.data[60000:], mnist.target[60000:]
shuffle(X_train, y_train)

clf = LinearSVC(C=1.0)
clf.fit(X_train, y_train)
accuracy_score(y_test, clf.predict(X_test)) # выдает 0.8883
```

Размножение данных

Тут можно делать много всего. Попробуем повернуть каждую цифру на случайный небольшой угол от -15 до 15 градусов.

```
from skimage.transform import rotate
rotated_X_train = []
for x in X_train:
    angle = np.random.rand() * 30 - 15
    rot_img = rotate(x.reshape(28, 28), angle)
    rotated_X_train.append(rot_img.flatten())
rotated_X_train = np.array(rotated_X_train)

new_X = np.concatenate([X_train, rotated_X_train], axis=0)
new_y = np.concatenate([y_train, y_train])

clf = LinearSVC(C=1.0)
clf.fit(new_X, new_y)
accuracy_score(y_test, clf.predict(X_test)) # выдает 0.9042
```

HOG дескрипторы

Извлечем HOG-дескрипторы для каждой картинки:

```
def make_hog(X):
    hog_X = []
    for x in X:
        img = x.reshape(28, 28)
        features = hog(img, orientations=8,
                       pixels_per_cell=(4, 4),
                       cells_per_block=(1, 1))
        hog_X.append(features.flatten())
    return np.array(hog_X)

hog_X_train, hog_X_test = make_hog(new_X), make_hog(X_test)

clf = LinearSVC(C=1.0).fit(hog_X_train, new_y)
y_pred = clf.predict(hog_X_test)
accuracy_score(y_test, y_pred) # выдает 0.975
```

DAISY дескрипторы

Добавим DAISY дескрипторы к HOG.

```
def make_daisy(X):
    daisy_X = []
    for x in X:
        img = x.reshape(28, 28)
        descs = daisy(img, step=7, radius=7, rings=2,
                      histograms=6, orientations=8)
        daisy_X.append(descs.flatten())
    return np.array(daisy_X)

best_X_train = np.hstack([hog_X_train, make_daisy(new_X)])
best_X_test  = np.hstack([hog_X_test , make_daisy(X_test)])

clf = LinearSVC(C=1.0).fit(best_X_train, new_y)
y_pred = clf.predict(best_X_test)
accuracy_score(y_test, y_pred) # выдает 0.9822
```

Бонус

В качестве бонуса можно попробовать подобрать параметры SVM:

```
clf = LinearSVC(C=10.0).fit(best_X_train, new_y)
y_pred = clf.predict(best_X_test)
accuracy_score(y_test, y_pred) # выдает 0.9858
```

Содержание презентации

Итоги

- scikit-image — удобная библиотека для работы с изображениями
- Ее можно успешно применять для машинного обучения при работе с данными в виде картинок
- В библиотеке реализовано еще очень много разных возможностей и алгоритмов (фильтры, скелеты, сегментация, убирание шума, и т.д.)