Classifying objects with Machine Learning techniques

Digital Image Processing project

Falcone Federico - 86048   Garda Massimiliano - 85965

University of Brescia
`Prof. Nicola Adami`

Academic Year 2014/2015

Contents

Contents 1

Introduction

Google Image $\rightarrow$ Classifying images by keywords found on Internet

Our Project $\rightarrow$ Classifying objects using a dataset of images grouped by category.

Introduction

Our Project (1)

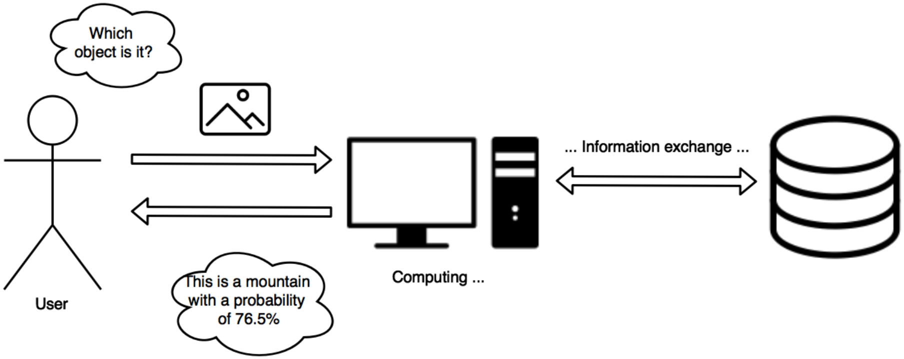
Which object is it?
User
This is a mountain with a probability of 76.5%
Computing ...
... Information exchange ...

Figure 1: Example of a query

Introduction

Our Project (2)

How the system should work. . .

- Download images from Google
- Subdividing images into categories
- Store images in a database.

. . . what it is necessary to implement

- Features extraction
- Machine Learning Techniques
- Taking probabilities into account

Contents 2

Features

Interesting questions before starting:

- **Global** vs **Local** features: are they so different?
- How to **describe** an image to compare it with another one?

Features

Global features

Global features describe the entire image in terms of:

- Shape
- Texture
- Color

Attention: Global features are sensitive to *clutter* and *occlusion*

Features

Global features: example with Color Histogram method

Problem: Whose histogram is this?

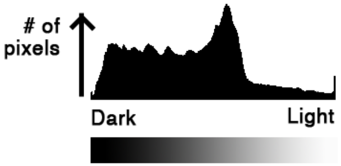

Figure 2: Color histogram of an image

Color Histogram method

This basic approach starts from the *histogram* of an image, counting the number of pixel values from 0 to 255

. . . but this method is very image-preprocessing bounded

Features

Local features

A **local feature** is an image pattern which differs from its immediate neighborhood.

It is usually associated with a *change* of an image property or several properties simultaneously, although it is not necessarily localized exactly on this change.

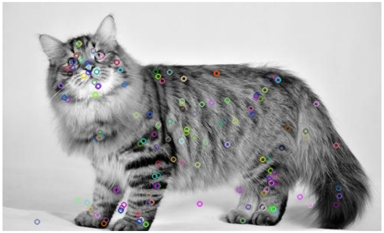

Figure 3: An example of keypoints of an image (MSER-SIFT)

Features detectors and descriptors

Base concepts

Features detectors

The first step of the feature extraction process is to find a set of distinctive **keypoints** that can be reliably localized under varying imaging conditions, viewpoint changes, and in the presence of noise.

Features descriptors

Once a set of interest regions has been extracted from an image, their content needs to be encoded in a descriptor that is suitable for discriminative matching.

Features detectors and descriptors

Base concepts

Features detectors

The first step of the feature extraction process is to find a set of distinctive **keypoints** that can be reliably localized under varying imaging conditions, viewpoint changes, and in the presence of noise.

Features descriptors

Once a set of interest regions has been extracted from an image, their content needs to be encoded in a descriptor that is suitable for discriminative matching.

Features detectors and descriptors

General Overview

In literature you can find a lot of features detection/description methods, such as:

Detectors

- HARRIS
- SIFT
- SURF
- ORB
- MSER
- FAST

Descriptors

- SIFT
- SURF
- ORB
- HOG
- BRIEF

Features detectors and descriptors

HARRIS detector

- Designed for **geometric stability**
- The keypoints it finds corresponds to **corner-like** structures
- It uses an approach based on two dominant orientation, using the two principal components of a matrix (the components are the maximum eigenvalues, using PCA transformation)
- If the image scale differs too much between the test images, then the extracted structures will be different, too

Features detectors and descriptors

HOG descriptor

The essential thought behind the **histogram of oriented gradients**[1] descriptor is that local object appearance and shape within an image can be described by the *distribution of intensity gradients* or *edge directions*.

The image is divided into small connected regions called **cells**, and for the pixels within each cell, a histogram of gradient directions is computed.

The descriptor is then the concatenation of these histograms.

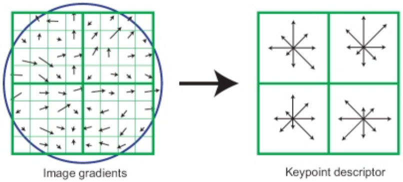

Figure 4: HOG descriptor

[1] Often used for Human Detection

Features detectors and descriptors

SIFT descriptor

SIFT = Scale-invariant feature transform

Main goal

We want to use features that are **invariant** to translation and scale changes

This descriptor aims to achieve robustness to **lighting variations** and **small positional shifts** by encoding the image information in a localized set of gradient orientation histograms.
The descriptor computation starts from a scale and rotation normalized region extracted with a detector method.

Features detectors and descriptors

SURF detector/descriptor

SURF = Speeded Up Robust Features

- Another features detector/descriptor that behaves similar to SIFT
- Instead of relying on Gaussian derivatives for its internal computations, it is based on simple 2D box filters [2]
- It computes several summary statistics over the image, such as $\sum dx$, $\sum dy$, $\sum |dx|$, $\sum |dy|$

[2] aka *Haar Wavelet*

Features detectors and descriptors

MSER detector

MSER = Maximally stable extremal regions

- Extracts **homogeneous intensity regions**
- These regions are stable over a range of imaging conditions and can still be reliably extracted under viewpoint changes.
  Since they are generated by a segmentation process, they are not restricted to elliptical shapes, but can have **complicated contours**.

Features detectors and descriptors

Examples of various algorithms

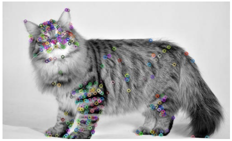
Figure 5: HARRIS-SIFT

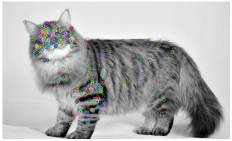
Figure 6: ORB-ORB

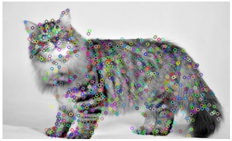
Figure 7: SIFT-SIFT

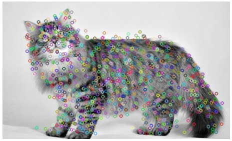
Figure 8: SURF-SURF

Machine Learning Techniques

- Which kind of approach is better?
- *Global Classifier* or *OneClass Classifier*?
- Too Many attributes: is it a bad thing?
- What *learner* choose?

Machine Learning Techniques

General overview

Unsupervised Learning

These methods take in input only the value X, that represent the observed data coming from the experiments (aka **attributes**)

Supervised Learning

These methods take in input the couple (X, Y), where X is the set of the attributes and Y is the value of the **class** attribute

Machine Learning Techniques

General overview

Unsupervised Learning

These methods take in input only the value X, that represent the observed data coming from the experiments (aka **attributes**)

Supervised Learning

These methods take in input the couple (X, Y), where X is the set of the attributes and Y is the value of the **class** attribute

Machine Learning Techniques

Different approaches: Global Classifier and OneClass Classifier

Global Classifier $\rightarrow$ The number of classes is the same as the number of images' categories.
This approach can classify an image using only one classifier trained over all categories.

OneClass Classifier $\rightarrow$ The number of classes is only two: 0 if the image belongs to that class (*category*) and 1 otherwise.
This approach can distinguish if the image belongs to that class or not.

Machine Learning Techniques

Different approaches: Global Classifier and OneClass Classifier

OneClass Classifier can be used as:

- **OneVsAll** Classifier, fitting n classifiers considering one category as "Positive" and the others as "Negative",
- **"*Fair*" OneVsAll** Classifier, fitting n classifiers as the previous point, but each classifier is trained with 50% of positive examples and 50% of negative examples [3],
- **OneVsOne** Classifier, fitting $\binom{n}{2}$ classifiers, where n is the number of categories.

Even if this kind of approach seems more expensive, it can lead to a greater level of scalability. Furthermore, it's less sensitive to outliers.

[3]this is the reason of the name "fair"

Machine Learning Techniques

Dimensionality Reduction

Dimensionality reduction is used to reduce the number of the variables (**attributes**), in order to catch the subset of variables that explain most the model we are going to create

PCA

Principal Component Analysis is a *linear* transformation between a large set of variables into a smaller one, using the *covariance matrix* and its *eigenvalues* to map the attributes in a new space, selecting the components using the size of the eigenvalues, in descent order.

Machine Learning Techniques

Dimensionality Reduction

Question: *Is our dataset suitable for PCA analysis?*

Attention

If the attributes are not correlated each other, the PCA analysis leads to variables with *low explained variance* and so they cannot capture the variability of the data

We noticed that reducing the number of components to 2, 3,. . . ,10,. . . and so on the overall explained variance ratio is low, with peaks of 30% maximum . . .

This is caused by the way the features are computed!

Machine Learning Techniques

Learning algorithms

In this project we focus the attention on these *learning algorithms*:

- **SVM** - *Support Vector Machines*
- Ensemble learning:
  - **Random Forest**©
  - **AdaBoost** - *Adaptive Boosting*

Machine Learning Techniques

Learning algorithms: SVM

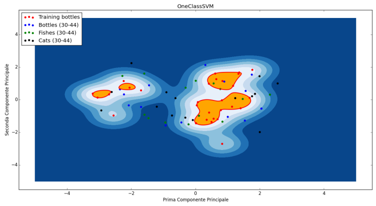

Figure 9: *OneClassSVM* trained on `bottles`

A *SVM* allows to find hyperplanes that are able to separate the data within two partitions; the **kernel** is a function that allows to separate positive examples from negative ones. Its expression can be linear, gaussian, polynomial, sigmoidal and so on . . .

Here we see that there isn't a perfect kernel function that contributes to create a strong separation between data. . .

Machine Learning Techniques

Learning algorithms: SVM

To test the configuration of SVM as a OneClass classifier, we apply the PCA dimensionality reduction in order to see the points (our image's attributes) in a *2D* space. . .

Warning

The shape of the kernel doesn't help finding a good separation between points, and the error rate is not low!
Even if we consider more than two components, the sum of the explained variance increases very slowly and we can't visualize in a clean way the boundaries and the shape of the kernel. . .

Machine Learning Techniques

Learning algorithms: Random Forest©

Random Forest© is an ensemble learning algorithm that, instead of using only one-time learners, builds a bag of learners sampling with replacement from the training set (*Bagging*).

Example

$n = 50$ trees are grown in order to reduce the probability of misclassification→this works if the failure rate of a single tree classifier is less than $\epsilon = 0.5$, from the *binomial distribution*

$$p_{err} = \sum_{i=26}^{50} \binom{50}{i} \epsilon^i (1-\epsilon)^{50-i}$$

This represents the probability that at least 26 learners out of 50 make the wrong prediction

It can be shown that, under the previous condition on ϵ, $p_{err} \ll \epsilon$.

Machine Learning Techniques

Learning algorithms: Random Forest$^{©}$

The prediction depends on the type of the **target** class attribute:

1. *Regression*: If the type is *continuous* then the RF$^{©}$ classifier will made a **mean** of the class attribute predicted by the ensemble of trees
2. *Classification*: If the type is *discrete*, such as in our situation, the output will be the **mode** of the predicted class values

$\rightarrow$Our classifiers are based on a *voting system*, as remarked in point (2)

Our Idea

Basic concept

We have taken some aspects into account for the classification task.

1. **Features**: we have decided to save features in the same folder of the images. In this file are saved only the features of the images in the folder (that represents one category). With this approach, if you will add new images in the folder, you can simply update the Features' file. A possible extension is to save these features in a database.
2. **Learning Algorithm**: we have decided to train a model with the approach "Belongs to / Not Belongs to". In this case we use the Features' file to train the model that can predict if a new image belongs or not to the category. We used k-fold validation because of robustness of this approach.

Our Idea

Basic concept

3. **Probability**: we know that features are not so precise to describe an image. As we said before, their performances depend on a large number of factors. Firstly, images that we have are not so clean from outliers and objects in the images are not so defined as we expect. So we have found a method to limit errors on classification...

Our Idea

How to Classify Object with Probability

Main Idea

The main idea is to classify an object by seeing which category has the **highest probability** that permits to be sure to say that the object belongs to it.

Our Idea

How to Classify Object with Probability

Procedure to guess the correct classification:

1. Extract features from the new image.
2. Do a k-fold validation from images contained into the folder(training and test images).
3. For each fold (training and test set) make a prediction of the images.
4. Save prediction/probability for every feature extracted, for each image.
5. Consider which category has the highest number of matching features and show the probability of belonging to this class.

Our Idea

How to Classify Object with Probability

We can obtain three different result:

1. The image is close to other images and only the correct category matches with a lot of features. In this case the result of classification is almost correct.
2. The image is not so close to other images. So the image could belong to many different categories. In this case it is done an evaluation of the categories with the highest probability and it is necessary to compare them to each other.
3. The image doesn't belong to any category. Maybe we have some features that matched, but the probability is not so high. It has been fixed some threshold to make sure that no category will be choose to categorize the image.

Contents 3

Case Study

Main outline

To implement our system we will follow these steps:

- Use of built-in functions provided by `OpenCV` to deal with features detection and description,
- explain the structure of our scripts written in `Python`,
- introduce the ML algorithms and the validation methods to build a strong learner.

OpenCV

Wrapper methods in OpenCV

OpenCV lets the programmer use some *wrapper methods* in order to get the features **detector** and **descriptor** from an image as it is shown in the code below:

Listing 1: detectAndExtract

```
class detectAndExtract:
  _detector = None
  _descriptor = None

  def __init__(self,detector,descriptor):
    self._detector = detector
    self._descriptor = descriptor

  def elabora(self.path):
    # Calculate features ...
    return out.tolist()
```

OpenCV

Wrapper methods in OpenCV

The class `detectAndExtract` is used in the framework. The parameters are `String` and methods return a list of all features detected.
One of the advantage to use this kind of configuration is that the method itself holds the detection and the description task.

Listing 2: An example of using detectAndExtract's method

```
obj = detectAndExtract(detector,extractor)
array = obj.elabora(path)
```

The FeaturesFile's Class

Files' creation and loading

Now we have to save features in a file in order to train our model with Machine Learning algorithms.
So we have decided to create this class that can save all features of a given keyword in a file.
It wouldn't have been a good idea if this file had been created every time ex-novo and so if the file exists, the class loads into an array the features contained in the file.

Features file

File is a `.csv` file and its name is composed like this:
`keywordsname_DETECTOR_DESCRIPTOR.csv`

We can have different files, one for every features extracted.

Image detectors and descriptors

Our choices

We noticed that the goodness of the features depends on the type of the images we are going to use...

We used these features:

```
detector_descriptors=[["MSER","SIFT"],["HARRIS","SIFT"],
["SIFT","SIFT"],["ORB","ORB"],["FAST","SURF"],["FAST","BRIEF"],
["HOG","HOG"]]
```

→ See the code for further and specific examples...

Machine Learning Implementation

Once features are saved in files, it's quite easy to apply Machine Learning Techniques on data. The dataset is divided into **attributes**, that characterize features, and **target**, a value that represent the class value[4].

But there is one more question: *What algorithm to choose?*

Python Libraries

We've used scikit-learn libraries for this part of Machine Learning. All informations are available on this site `http://scikit-learn.org`.

[4] 0 $\rightarrow$ if it belongs to, 1 $\rightarrow$ if it not belongs to

Machine Learning Implementation

What algorithm to choose?

We have tested algorithms explained in the first part, and we have noticed some important points we want to focus on:

- **OneClassSVM** $\rightarrow$ It's not so precise to classify new images and it badly classifies images that we know they belong to the same class [5].
- **AdaBoostClassifier** $\rightarrow$ This Decision Tree Classifier seems to work very well, because of its nature of being *iterative*: new models are influenced by performance of previously built ones. However it's slower than RandomForestClassifier[6].

[5] because it's difficult to find a suitable kernel shape

[6] provided that `n_estimators` is the same

Machine Learning Implementation

What algorithm to choose?

- **RandomForestClassifier** $\rightarrow$ This method is quicker than others and the accuracy is very high. In the next slides we will explain all the performance obtained by using it.

Random Forest Classifier

It's a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.

Machine Learning Implementation

Random Forest© Classifier

Listing 3: Parameters of RandomForestClassifier

```
clf = RandomForestClassifier(n_estimators=300,n_jobs=-1)
```

The parameters are the following:

- `n_estimators`: 300 → The number of trees grown in the forest.
- `criterion`: *gini* → The function to measure the quality of a split (Gini impurity or entropy).
- `n_jobs`: −1 → The number of jobs to run in parallel for both fit and predict. (−1 stands for: all cores).

The Features Learning's Class

To implement the algorithm and use it easily with the other part of the code, we have created `FeaturesLearning` Class. Given positive and negative instances, it can train a model using a `cross_validation.StratifiedKFold` [7] and then it returns an array of probabilities for each fold that indicates belonging to the class.
This procedure is invoked in the `Main Class` for each category in the database, given an image to classify.

Just for the sake of argument, we try to run the **OneVsOne** classifier version, but is usable only with few categories. . .

[7] to preserve the proportion between positive and negative examples

The Features Learning's Class

Code's overview (1)

First of all we import all necessary libraries from scikit-learn

Listing 4: Library imported in FeaturesLearning Class

```
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import confusion_matrix,
  classification_report
from sklearn.cross_validation import cross_val_score
from sklearn import decomposition
from sklearn import cross_validation
```

These are necessary to train model and show results.

The Features Learning's Class

Code's overview (2)

Then in Class FeaturesLearning we have these methods.

Listing 5: Methods of Fearures Learning Class

```
class FeaturesLearning(object):
  """docstring for FeaturesLearning"""
  X_positive = None
  X_negative = None
  clf = RandomForestClassifier(n_estimators=300,n_jobs=-
  def __init__(self, X_positive, X_negative,ffe):
    # Constructor Method ...
  def trainModel(self,path):
    # To train model with cross_validation kfold ...
  def _prediction(self,featuresFile,path,class_test=None
    # To make a prediction ...
```

Our Idea Implementation

Overview (1)

We have constructed three different classes that can permit us to separate the part of Digital Image from the part of Machine Learning and Files' management. In this way we can modified a single part and let the others work as well as before.
For example we can substitute the part of Files' management[8] with a part of Database's reading/writing[9].

[8]We haven't explain this part in this presentation because we think it's not important to the goal of our project

[9]this was the initial idea:we adopted the described approach for a matter of time, speed and performance

Our Idea Implementation

Overview (2)

We decided to implement some `dict()` in order to create a sort of counter for each category and a data structure to keep the predictions' probabilities. . .

Example

`{'old_phone': 9, 'ball': 18, 'bicycle': 11, 'bottle': 8, 'table': 9}` (numbers represent the count of matches for that specific category)

Probability

To measure the probability of belonging to a particular class we used the method `predict_proba(y_test)` of the RF© classifier; the predicted class for `y_test` is the one with $p_i \geq 0.5$

Explanation

The predicted class probabilities of an input sample is computed as the mean predicted class probabilities of the trees in the forest. The class probability of a single tree is the fraction of samples of the same class in a leaf.
If the ensemble is composed of n trees, then the prediction is made counting how many trees predicted class value "0" and class value If $n = 300$ and 200 trees predicted "0" then $p_0 = \frac{200}{300} \approx 0.67$ and $p_1 = 1 - p_0 \approx 0.33$

To control this threshold in the program, we used a global variable, `threshold_prob`, to make the classification task more/less strict

Results

As an example, we reported here the results we obtained with a test images, using `threshold_prob=0.5` and `threshold_class=3`, with the approaches we introduced in slide 22

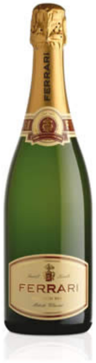

Figure 10: Our test image, `ferrari.jpeg`

Results (2)

"*Fair*" **OneVsAll** classifier

```
The image is a pencil with probabilities of 0.620444444444 and 3 of 6 matching
The image is a shark with probabilities of 0.569666666667 and 2 of 6 matching
The image is a ball with probabilities of 0.547333333333 and 1 of 6 matching
The image is a bicycle with probabilities of 0.590333333333 and 4 of 6 matching
The image is a clock with probabilities of 0.552666666667 and 3 of 6 matching
The image is a fish with probabilities of 0.614 and 1 of 6 matching
The image is a cat with probabilities of 0.6072 and 5 of 6 matching
The image is a old_phone with probabilities of 0.576666666667 and 4 of 6 matching
The image is a bottle with probabilities of 0.693866666667 and 5 of 6 matching

The image is a table with probabilities of 0.678666666667 and 2 of 6 matching
```

Results (3)

OneVsOne classifier

```
Threshold on positives :  at least 3 with probability >= 0.5
{'old_phone':  12, 'ball':  9, 'bicycle':  16, 'bottle':  13, 'table':  6}

{'old_phone':  0.66155555555555567, 'ball':  0.65526881720430119, 'bicycle':  0.64967213114754108,'bottle':

0.75313725490196071, 'table':0.60030303030303034}
```

Warning: we limited the number of categories because building $\binom{10}{2}$ classifiers is very expensive...

Contents 4

Final Considerations

- Is database approach faster?
- The influence of preprocessing (*example*$\rightarrow$ in literature, saliency maps for object extraction. . .)
- Other type of dimensionality reduction methods, but. . . be careful!
- Deep knowledge of the features detector/description algorithms could lead to a better classification. . .