IN4320 Machine Learning Computer Exercise

Version 0.5

May 14, 2019

Multiple Instance Learning: image classification

In this exercise we will make an image classifier, using a Multiple Instance Learning approach. To keep it simple, we will use a relatively tiny dataset, with simple features and a simple classifier.

We will try to distinguish images of apples from bananas. Below are two examples of images, the first containing an apple, the second containing a banana. Notice that the backgrounds in the images are quite similar.

Note that it is very important to *explain* how you come to an answer. For some boring questions you don't need to give an answer, and I will put `No answer needed` in the question.

To make an image classifier based on MIL, we have to make several steps:

1. Define the instances that constitute a bag. Here we will make use of a mean-shift image segmentation to segment an image into subparts.

(600+, 3) vector 120 bags

2. Define the features that characterize each instance. Here we will just use the (average) red, green and blue color.

3. Define the MIL classifier. Here we will use a naive approach, that is using a standard classifier trained on the individual instances.

4. Define the combination rule that combines the predicted labels on the individual instances to a predicted label for a bag.

In the coming exercise we will go though it step by step.

Note: for the mean-shift algorithm and the Liknon classifier (also called, the L1 support vector classifier), I supplied Matlab functions. If you want to use another programming language, feel free, but you have to find (or make) your own implementations.

1 The Naive MIL classifier

(a) `No answer needed` Get the data `sival_apple_banana.zip` and some additional Matlab functions `additionalcode.zip` from Brightspace. The data should contain two folders, one with `apple`, the other with `banana`. The additionalcode contains a function to segment an image `im_meanshift`, and a function to convert a cell-array of bags to a Prtools dataset `bags2dataset`.

1 Implement a script that reads all images from a given directory. You can use the Matlab functions `dir` and `imread`.

(b) Next implement a function 1 `extractinstances` that segments an image using the Mean Shift algorithm (using `im_meanshift`), computes the average red, green and blue color per segment, and returns the resulting features in a small data matrix.

2 width describe

Notice that the number of segments that you obtain depends on a `width`-parameter that you have to supply to `im_meanshift`. Set this parameter such that the background and the foreground are segmented reasonably well. 3 Would it be better to oversegment, or undersegment the images? 4 What value of the width parameter did you set?

(c) Create a function `gendatmilsival` that creates a MIL dataset, by going through all apple and banana-images, extracting the instances per

image, and storing them in a Prtools dataset with `bags2dataset`. Note that, in addition to the class labels, also the bag identifiers are stored in the dataset. If you are interested, you can retrieve them using `bagid = getident(a,'milbag')`.

How many bags did you obtain? How many features do the instances have? How many instances are there per bag? Make a scatterplot to see if the instances from the two classes are a bit separable.

(d) Create a function `combineinstlabels` that accepts a list of labels, and outputs a single label obtained by majority voting.

(e) Now we are almost ready to classify images... First we have to train a classifier; let's use a Fisher classifier for this. Now apply the trained classifier to each instance in a bag, classify the instances (using `labeld`), and combine the label outputs (using your `combineinstlabels`) to get a bag label.

How many apple images are misclassified to be banana? And vice versa? Why is this error estimate not trustworthy? Estimate the classification error in a trustworthy way!

(f) Invent at least two ways in which you may improve the performance of this classifier. Argue why it may improve the performance.

2 MILES

The classifier that we used in the previous section was very simple. In this section we implement one of the most successful classifiers, called MILES. Also have a look at the article "MILES: Multiple-instance learning via embedded instance selection." by Chen, Yixin, Jinbo Bi, and James Ze Wang, IEEE Transactions on Pattern Analysis and Machine Intelligence, (2006): 1931-1947.

(a) Implement a function `bagembed` that represents a bag of instances $\mathbf{B}_i$ by a feature vector $\mathbf{m}(\mathbf{B}_i)$, using equation (7) from the article.

How large will this feature vector $\mathbf{m}(\mathbf{B}_i)$ become for our apple-banana problem?

1 description

(b) Make a Prtools dataset with the vectors $\mathbf{m}(\mathbf{B}_i)$ and their corresponding labels y_i. Choose a sensible value for σ such that not all numbers become 0 or 1. Train on this large dataset a L_1-support vector classifier (or, more correctly called, LIKNON): `liknonc`.

2 sigma

(c) Test the LIKNON classifier on this dataset: how many errors is this classifier making? Is this classifier better than the naive MIL classifier trained in the previous section? What can you do to make this MILES classifier perform better?

1

2 YES +description

3

3 Another MIL classifier

(a) Finally, implement your own MIL classifier. Any classifier may do, *except* for the Naive approach[1] and the MILES classifier (obviously). It may be something you invented yourself, or some classifier from literature.

Explain what MIL classifier you are implementing, give the code, and compare its performance with that of the Naive classifier (i.e. the Fisher classifier with a majority vote), and the performance of the MILES classifier.

[1] No, just replacing the Fisher classifier in the first section is not enough!