NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE

Assignment 5

TDT4173: Machine Learning and Case-Based Reasoning

Submitted by
Vittorio Triassi

November 2019

1 Introduction

a) A brief overview of the whole system. This includes a short explanation on how to run it as well as what libraries your system is using.

For the purpose of the following assignment we decided to implement every task using `scikit-learn`. The main reason is that we already had the chance to use such library by implementing a few models in the previous assignment, and that is why we thought it would be a good idea to stick with it in this case as well. It is also true that several are the libraries available to carry out the same tasks. With a better understanding of the architectures and of course of the syntax, it would have been nice to use such libraries. The report is divided into four parts: (1) feature engineering (2) character classification (3) character detection and finally (4) conclusions are drawn.

In the *feature engineering* part, we decided to first perform the standardization of our features and then applied the principal component analysis. As far as the *character classification* we used two classifiers provided by the `scikit-learn` library, more specifically, we used a Support Vector Classifier and a Random Forest Classifier. In the *character detection* we performed the sliding window tecnique instead.

In order to be able to run the code, it is necessary to use *Jupyter*, since a *Jupyter notebook* written in Python was used. In the case that a `scikit-image` error is thrown, it will be necessary to uncomment the following line in the notebook: `!pip install scikit-image`. To evaluate the performance of our models, `metrics` from `scikit-learn` was used.

2
Feature Engineering

b) Explain how each of the feature engineering techniques you selected work. Why did you select these techniques? Justify your answer.

To carry out the *feature engineering* part, we decided to: *standardize* the features and apply the *principal component analysis*. The main reason why we chose the following tecniques is that almost all the machine learning estimators need standardized datasets. When we talk about *standardizing* our dataset, we are saying nothing but removing the mean and scaling to unit variance. In our task, we used `StandardScaler` from `sklearn.preprocessing`. What we do is to center and scale each feature. Another reason why scaling the features is a good practice is that if a feature has a variance that is way larger than the others, it might dominate the objective function ending up in an estimator not able to learn as expected [2]. The second tecnique used is the *principal component analysis* (PCA). PCA uses Singular Value Decomposition to project high dimensional data to a lower dimensional representation. In order for PCA to work, the input has to be centered before applying the SVD [1]. In our task, `PCA` from `sklearn.decomposition` was used.

c) Were there any techniques that you wanted to try, but for some reason were not able to? Explain.

When talking about feature engineering, several are the tecniques that can be used. Moreover, when deciding which tecnique is worth using, the final choice really depends on the problem we are trying to address. In our case, it would have been a good exercise to perform manual cropping, but it is also true that we are trying our best to automate the way we can get insights from our data and manual cropping would have been time consuming. Since we are dealing with images, it would have been definitely interesting to understand how SIFT would have performed on our data. Usually, they can provide us very good properties.

3
Character Classification

d) After having looked at the dataset, did you have any initial idea about what kind of model that might work well? Explain your reasoning.

Since we were dealing with a problem that involved images, our first thought was to use specific networks that perform well on classification tasks when the input is 2D. It would have been nice to first use a classic Artificial Neural Network (ANN), that would have been later extended to a better Convolutional Neural Network (CNN).

e) Give a description of the two models you elected to use. The description must include a brief explanation on how they work. Why did you select these models?

The models we used in the following task were a Support Vector Machine (SVM) and a Random Forest. In SVMs, we try to find a hyperplane that is able to separate two classes of the data by finding the largest margin. SVMs have a good accuracy compared to other tecniques but are very hard to tune because they involve a lot of parameters. Random Forest is a classifier that fits a number of decision tree classifiers on sub-sets of the dataset and averages the results to improve the accuracy of the model without running into overfitting. In our code, we used `SVC` from `sklearn.svm` and `RandomForestClassifier` from `sklearn-ensemble`. We chose the following models because we wanted to test less trending approaches to solve an interesting problem such as image classification. Usually, CNNs are preferred over other tecniques.

f) A critical evaluation of your two models. How are you measuring their performance? How did they do? Which model gave the best results? Include at least five predictions in the report (both good and bad).

Between our two models, SVM performed better, achieving an accuracy around 80% on the test data. The second model, the Random Forest, performed slightly worse

achieving 70%. We can be quite satisfied with the first model since we did not tuned its parameters so much and still we obtained good results. As stated before, SVMs can be hard to tune and at least in our case, we have simply used the default kernel (`rbf`) and tried a few values for `gamma`. Both models were run after having performed the feature scaling. In Figure 3.1, five predictions from both classifiers are shown. As we can see, the SVM is more precise in its predictions, while the RFC gets more often confused.

```
Random predictions for SVM      Random predictions for RFC
Target: 17 / Prediction: [17]   Target: 13 / Prediction: [11]
Target: 17 / Prediction: [17]   Target: 0 / Prediction: [0]
Target: 3 / Prediction: [3]     Target: 4 / Prediction: [12]
Target: 19 / Prediction: [19]   Target: 14 / Prediction: [14]
Target: 7 / Prediction: [7]     Target: 4 / Prediction: [4]
```

Figure 3.1: SVM and RFC predictions.

In the Appendix, the classification reports of both classifiers are shown (see A.1 and A.2).

g) Were there any additional models that you would have liked to try, but for some reason were not able to? Explain.

As stated in the *Introduction*, we decided to stick with `scikit-learn`. It would have been interesting to implement a *Convolutional Neural Network* (CNN) by using proper libraries such as Tensorflow or even better: Keras. The reason why CNN would have been a good fit is mainly that with such networks we do not "destroy" information by flattening everything as we did. This enables us to achieve way better perfomance in terms of accuracy on the test set.

4
Character Detection

h) Test your character detector on detection-1.jpg and detection-2.jpg and show the result in the report. Feel free to find or create additional images to test your detector, if you are so inclined.

i) Give an evaluation of your detection system. How does it perform?

The detection component has better performance on the first image since all the characters are on the right scale and are shown vertically, so no rotated characters appear. In the second one it struggles more since we did not consider specific improvements for the rotations. Also, uppercase letters seem to be recognized better than lowercase.

j) Describe any improvements you made to your detector. Discuss how you can improve your system further.

Several are the improvements that are possible when trying to address an object detection task. In our case, two different images were provided. In the first image, even without specific modifications, it is possible to detect the characters. On the other hand in the second one, we notice that a few characters are rotated. Unless our detector is implemented in such a way that considers also other rotations, it is not able to detect such angles. Other possible improvements might be related to the scale of the objects we want to detect. For instance, in our detector we set a window size of 20×20 pixels. If we are trying to detect something on a different scale, it will be a problem and our detector will not be as effective as it is on the aforementioned scale.

5 Conclusion

k) What is the weakest and strongest component of your OCR system (feature engineering, character classification, and character detection)? Explain your answer.

After having carried out all the tasks, we can say that the weakest component is definitely the *character detector*. The reason is that there are several cases that were not really considered, such as characters that might be rotated respected to the classic vertical representation and also the scale of the image itself. On the other hand, the strongest component might be the SVM classifier, which despite it is not the state-of-the-art model for image classification, performs very well.

l) What went good/bad with the project? Any lessons learned?

The assignment itself has been a very good exercise to have practical insights on a real problem such as image classification and detection. Something that I have personally experienced is that when there is no deep knowledge of a specific library, it gets difficult to properly use all the optimizations that can be easily offered. That was also the main reason we did not use libraries like Tensorflow or Keras that offer way better support and describe several concepts at a higher level compared to how we did in sk-learn. The accuracy we obtained with our models are not good enough to say that they can be used to deploy anything. It is also true that we did not use all of our dataset in the training step and we saved 20% of it for the test data.

References

[1] Scikit-learn. sklearn.decomposition.pca. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html. Accessed: 2019-11-14.

[2] Scikit-learn. sklearn.preprocessing.standardscaler. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html. Accessed: 2019-11-14.

A

Classification Reports

SVM classification report:

	precision	recall	f1-score	support
0	0.69	0.92	0.79	157
1	0.77	0.33	0.47	30
2	0.91	0.83	0.87	47
3	0.80	0.54	0.64	52
4	0.72	0.89	0.80	144
5	0.93	0.54	0.68	24
6	0.88	0.55	0.68	42
7	0.79	0.66	0.72	41
8	0.72	0.79	0.75	80
9	0.77	0.43	0.56	23
10	0.92	0.52	0.67	23
11	0.90	0.83	0.86	53
12	0.81	0.81	0.81	36
13	0.78	0.90	0.84	105
14	0.73	0.92	0.81	99
15	0.97	0.76	0.85	38
16	1.00	0.11	0.19	19
17	0.64	0.87	0.73	100
18	0.90	0.89	0.89	98
19	0.92	0.87	0.89	87
20	0.87	0.52	0.65	25
21	1.00	0.70	0.82	30
22	1.00	0.67	0.80	18
23	0.85	0.50	0.63	22
24	0.80	0.53	0.64	15
25	1.00	0.47	0.64	15
accuracy			0.78	1423
macro avg	0.85	0.67	0.72	1423
weighted avg	0.80	0.78	0.77	1423

Figure A.1: SVM classification report.

RFC classification report:

	precision	recall	f1-score	support
0	0.61	0.89	0.72	157
1	1.00	0.20	0.33	30
2	0.75	0.77	0.76	47
3	0.55	0.23	0.32	52
4	0.63	0.87	0.73	144
5	1.00	0.29	0.45	24
6	0.94	0.38	0.54	42
7	0.85	0.71	0.77	41
8	0.67	0.78	0.72	80
9	1.00	0.09	0.16	23
10	1.00	0.30	0.47	23
11	0.78	0.72	0.75	53
12	0.83	0.67	0.74	36
13	0.75	0.82	0.78	105
14	0.57	0.91	0.70	99
15	0.89	0.66	0.76	38
16	1.00	0.05	0.10	19
17	0.62	0.80	0.70	100
18	0.87	0.89	0.88	98
19	0.70	0.84	0.76	87
20	1.00	0.08	0.15	25
21	0.95	0.70	0.81	30
22	1.00	0.56	0.71	18
23	1.00	0.36	0.53	22
24	0.80	0.27	0.40	15
25	1.00	0.27	0.42	15
accuracy			0.70	1423
macro avg	0.84	0.54	0.58	1423
weighted avg	0.75	0.70	0.67	1423

Figure A.2: RFC classification report.