Image Recognition : Cats and Dogs case study

André Silva
ei12133@fe.up.pt

Diogo Pinto
ei11120@fe.up.pt

Rui Grandão
ei11010@fe.up.pt

Faculty of Engineering
University of Porto
Porto, Portugal

Abstract

Image labeling is one of the main tasks performed in the field of computer vision. Due to its relevance, many efforts are nowadays being done to improve the performance of state-of-the-art classification algorithms. We perform the task of distinguishing images with dogs from ones with cats, by selecting the best fit classifier for the task from a set of ones that are widely used in data science tasks, and with it we achieve an above average performance in a Kaggle competition in which the task is inspired.

1 Introduction

One of the most important tasks in computer vision is to be able to identify the contents of a given picture. This has proven to be a difficult task, but major breakthroughs proved that it is attainable [6].

One relaxation of that objective is to be able to distinguish images between a smaller set of classes of artifacts conveyed in them. In that sense, Kaggle launched a competition in September, 2013 [1] that challenged the community to perform the task of distinguishing computationally pictures with dogs from the ones with cats.

The document follows by further detailing the problem, then a revision over the classifiers used for the task, as well as the parameters used, and after that an evaluation of the results obtained and conclusions.

2 Image classification system

The system inputs a data set of 25000 pictures, with variable dimensions, from which half correspond to pictures with dogs, and another half to pictures with cats.

Both dogs and cats distribute over a large set of races, which diverge a lot, not only in terms of shape, but also in color. The pictures also diverge in terms of scale, rotation and transposition, and include other artifacts (e.g. people) that further complexify the problem.

Thus, we start by performing a small step of normalization, by converting the pictures to black and white.

2.1 Feature detection and description

The first step to describe an image through a set of features is to detect the most relevant points in it (normally they correspond to corners or edges). Then, those points have to be described in some way, according to their neighborhood, so that we are able to identify patterns in the distribution of key points over images.

To accomplish both of these tasks we used SIFT algorithm [7]. It describes each key point in an image through a vector of 128 features that provide information regarding numerous aspects of the key point, taking into account the neighborhood of them.

2.2 Feature matcher

2.2.1 Fast approximate nearest neighbor search

In order compare the descriptors from different images, we use the Fast Approximate Nearest Neighbor algorithm [8]. This matcher is provided to the Bag of Words algorithm, as described in Section 2.3.1.

2.3 Image representation

2.3.1 Bag of words

In order to use each image as a sample in the training of a classification algorithm, we must first normalize their representation into a comparable fixed-size vector of features. In order to obtain that, the following steps were performed:

- aggregate all the extracted descriptors from the images;
- perform k-means over those descriptors, and extract the corresponding k centroids;
- using this vocabulary definition, compute how close each of the images' descriptors are to each of the words;
- aggregate this information into a vector.

Using this, we are able to construct a matrix one image per row, and one feature per column.

2.4 Classification

In this section it is enumerated the different classifiers used by the system, in the task of deciding if an image contains a cat or a dog.

2.4.1 K-nearest neighbor with majority vote

Neighbors-based classifiers follows an instance-based learning paradigm: instead of constructing a model it stores instances of the training data. Classification is then obtained from a majority voting between the nearest samples to the sample being classified.

2.4.2 Gaussian Naive Bayes

It is a supervised learning algorithm that applies the Bayes' theorem assuming every pair of features are independent. Their probability follows a Gaussian function. The details on the algorithm used to update the feature means and variance can be found at the STAN-CS-79-773 [2] technical report.

2.4.3 Support vector machine

In order to perform classification, we used the binary classification approach that tries to assign labels to a set of instances by adopting the current standart support vector machines proposed by Corinna Cortes and Vapnik [3].

The classifier implements the "one-against-one" approach proposed by Knerr et al. [5] for multi-class classification and it uses a radial basis kernel function [9].

2.4.4 AdaBoost

The *Adaptive Boosting* ensemble algorithm was introduced by Freund and Schapire in 1995. It creates a series of weighted weak classifiers (according to the accuracy they obtain, and performs boosting in samples wrongly classified. Our system uses an extension proposed in 2009, by Zhu et al. [10], in order to support multi-class classification.

2.4.5 Random forest

Random decision forests [4] were initially proposed by Ho. in 1995 as a meta classifier for the decision tree classifiers that were vastly used due to their execution speed. The random forest was presented as being more accurate and expandable in order to obtain better scores for the training and test data. This estimator fits decision trees into sub-sets of fixed sized from the data domain and takes advantage of averaging to attenuate overfitting of the system, therefor resulting in a more accurate and robust classification.

2.5 Evaluation

In order to obtain the best classifier for the task, all the classification algorithms described in section 2.4 were fed into a function that performed the following steps:

- split the data in 70% for training and test sets and 30% for test;
- perform k-fold cross validation over the 70% of data where:
  - all the classifiers are fit to the training portion of data;
  - those classifiers are given a score based on the accuracy they have in the validation set.
- the classifier that obtained the highest score is then selected.

Afterwards, the selected classifier is fitted into all the training data and predicts the output for the new examples.

3 Results

3.1 General analysis of the obtained results

As we can see by the scores published from subsections 3.2.1 to 3.2.5, regarding a vocabulary with one hundred words, the Random Forest algorithm had the best results during cross-validation testing.
Whilst this algorithm had the best results of the bunch, the K-Nearest Neighbours algorithm was the most notable algorithm when it comes to predicting if an image had a cat present, as we can see from its precision during testing. This information is also easibly obtained from Table 1 regarding the K-Nearest Neighbours confusion matrix where we can see a low value of false postives for cat predictions.
Overall the precision was between 65% and 75% although this could be futurely enhanced, as testing with different vocabulary and feature sizes would enable us to fine-tune the parameters and get better precision values. The problem is that these algorithms are very resource heavy and time consuming so its hard to test all the values.

3.2 Vocabulary size: 100 words

3.2.1 Results of cross-validation for the K-Nearest Neighbours algorithm:

Score: 0.6524
Confusion Matrix:

Table 1: Confusion Matrix

		Predicted	
		Cat	Dog
Actual	Cat	1808	1942
	Dog	665	3085

Classification Report:

Table 2: K-Nearest Neighbours Scores

	Precision	Recall	F1-Score	Support
Cat	0.73	0.48	0.58	3750
Dog	0.61	0.82	0.7	3750
Average	0.67	0.65	0.64	7500

3.2.2 Results of cross-validation for the Gaussian Naive Bayes algorithm:

Score: 0.6241
Confusion Matrix:

Table 3: Confusion Matrix

		Predicted	
		Cat	Dog
Actual	Cat	2151	1599
	Dog	1220	2530

Classification Report:

Table 4: Gaussian Naive Bayes Scores

	Precision	Recall	F1-Score	Support
Cat	0.64	0.57	0.60	3750
Dog	0.61	0.67	0.64	3750
Average	0.63	0.62	0.62	7500

3.2.3 Results of cross-validation for the Support Vector Machine algorithm:

Score: 0.6899
Confusion Matrix:

Table 5: Confusion Matrix

		Predicted	
		Cat	Dog
Actual	Cat	2745	1005
	Dog	1321	2429

Classification Report:

Table 6: Support Vector Machine Scores

	Precision	Recall	F1-Score	Support
Cat	0.68	0.73	0.70	3750
Dog	0.71	0.65	0.68	3750
Average	0.69	0.69	0.69	7500

3.2.4 Results of cross-validation for the AdaBoost algorithm:

Score: 0.6971
Confusion Matrix:

Table 7: Confusion Matrix

		Predicted	
		Cat	Dog
Actual	Cat	2688	1062
	Dog	1210	2540

Classification Report:

Table 8: AdaBoost Scores

	Precision	Recall	F1-Score	Support
Cat	0.69	0.72	0.70	3750
Dog	0.71	0.68	0.69	3750
Average	0.70	0.70	0.70	7500

3.2.5 Results of cross-validation for the Random Forest algorithm:

Score: 0.7240

Confusion Matrix:

Table 9: Confusion Matrix

		Predicted	
		Cat	Dog
Actual	Cat	2963	787
	Dog	1283	2467

Classification Report:

Table 10: Random Forest Scores

	Precision	Recall	F1-Score	Support
Cat	0.70	0.79	0.74	3750
Dog	0.76	0.63	0.70	3750
Average	0.73	0.72	0.72	7500

4 Future work

There is still plenty of room for improvement. In the current method, other possibilities include performing grid search over different parameter values for the classifiers, therefore finding the best-performing instances of each for the current task.

Another option would be to use Deep Convolutional Neural Networks [6]. These work very well in determining image features and correlating them, and deal with highly non-linear classification problems.

5 Conclusion

With this work we demonstrated a common classification work-flow in computer vision tasks. We reviewed the overall details behind the classification algorithms used, and assessed their performance in the task of labeling pictures with dogs and with cats.

Thus, this work lays out the foundation to tackle bigger and complexer classification problems in computer vision field, serving as future reference.

References

[1] Kaggle dogs vs. cats. `https://www.kaggle.com/c/dogs-vs-cats`. Accessed: 2015-12-19.

[2] Tony F. Chan, Gene H. Golub, and Randall J. LeVeque. Updating formulae and a pairwise algorithm for computing sample variances. Technical Report STAN-CS-79-773, Department of Computer Science, Stanford University, November 1979. URL `http://i.stanford.edu/pub/cstr/reports/cs/tr/79/773/CS-TR-79-773.pdf`.

[3] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, September 1995. ISSN 0885-6125. doi: 10.1023/A:1022627411411. URL `http://dx.doi.org/10.1023/A:1022627411411`.

[4] Tin Kam Ho. Random decision forests. In *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1*, ICDAR '95, pages 278–, Washington, DC, USA, 1995. IEEE Computer Society. ISBN 0-8186-7128-9. URL `http://dl.acm.org/citation.cfm?id=844379.844681`.

[5] S. Knerr, L. Personnaz, and G. Dreyfus. Single-layer learning revisited: a stepwise procedure for building and training a neural network. In FranÃğoiseFogelman SouliÃľ and Jeanny HÃľrault, editors, *Neurocomputing*, volume 68 of *NATO ASI Series*, pages 41–50. Springer Berlin Heidelberg, 1990. ISBN 978-3-642-76155-3. doi: 10.1007/978-3-642-76153-9_5. URL `http://dx.doi.org/10.1007/978-3-642-76153-9_5`.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[7] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[8] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pages 331–340. INSTICC Press, 2009.

[9] J. P. Vert, K. Tsuda, and B. Scholkopf. A primer on kernel methods. *Kernel Methods in Computational Biology*, pages 35–70.

[10] Ji Zhu, Hui Zou, Saharon Rosset, and Trevor Hastie. Multi-class AdaBoost. *Statistics and Its Interface*, 2(3):349–360, 2009. ISSN 19387989. doi: 10.4310/sii.2009.v2.n3.a8. URL `http://dx.doi.org/10.4310/sii.2009.v2.n3.a8`.