Selected Topics in Machine Learning Coursework 1

Gustavo Brito Rodríguez
Imperial College London
01198785
gb1216@ic.ac.uk

Alberto García Matachana
Imperial College London
01205901
ag4116@ic.ac.uk

Abstract

This report explores different codebook generation techniques for building a visual vocabulary on a subset of the Caltech101 dataset. We then use such codebooks for training and testing a Random Forest network and discuss how hyperparameter tuning affects the accuracy and complexity of the classifier. Finally, the RF performance is compared to a state of the art classifier.

1. Question 1: K-Means Codebook

1.1. The Data Set

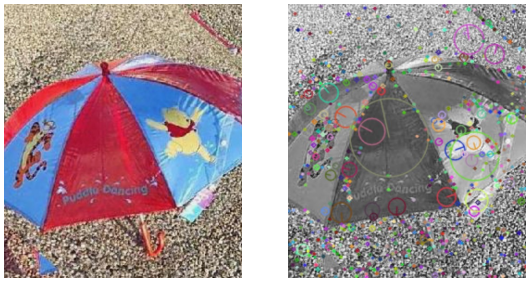
The data used for this coursework belonged to the *Caltech101* data set, a collection of images separated into 101 different categories with an average of 50 images each [4]. Out of these, 15 images from 10 different categories were randomly selected to form the training set X_{tr} and another 15 images from the same 10 categories were selected to form the testing set X_{tst}.

1.2. Obtaining Descriptors

This question's objective was to obtain a codebook that would allow for a new representation for the data, namely in a bag-of-words histogram.

The first step to do this was to obtain a set of descriptors from the X_{tr} for clustering. Initially, Scale-Invariant Feature Transform (SIFT) was used to detect and compute the descriptors from local features of images in the training set [6], given that it has been proven to provide superior accuracy over other local-descriptor methods [7].

A total of 65196×128 descriptor vectors were obtained from the 150 images in the training set. Consequently, PCA is applied to reduce the computational complexity involved on the clustering algorithm for vocabulary creation [10]. In order to account for 99% of the variance, the dimension was reduced from 128 to 109. K-Means clustering was applied on both the 128 (original) and 109 (PCA) dimensional descriptors with 100 clusters. This took a total of 367 seconds on the original descriptors and 266 seconds on the PCA descriptors. This is not an improvement given that PCA took 131 seconds to be performed. Nevertheless, in 1.3, we ran K-means repeatedly for bigger number of clusters, increasing the run-time clustering complexity and therefore making PCA dimensionality reduction worth using.



(a) (b)

Figure 1: Sample X_{tr} Image with SIFT features

1.3. Creating Vocabulary

Clustering was applied to obtain a discretised vocabulary which provided some invariance to small changes within the appearance of objects, since using all 65,196 PCA features as the vocabulary would lead to great computational complexity. Nevertheless, it is not possible to immediately know the optimal number of clusters to separate the data into. For that reason, Gaussian Mixture Models, with a varying number of clusters, were used for clustering. Then, the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) were calculated in order to decide on an optimal number of clusters [1, 11]. Gaussian Mixture models were necessary because they allow for the log-likelihood of the model to be calculated, otherwise impossible with K-means clustering. The formulas for the AIC and BIC can be found in Appendix A.1. The graph plotting the AIC and BIC against the number of clusters used is shown on Figure 2.

The AIC and BIC are a measure of a model's quality in comparison to the other models, i.e., they are a method of model selection. The best model will have the lowest values of AIC and BIC. As can be seen in the equations in Appendix A.1, both of these reward a model for how well it fits the data, but penalises it for having a high number of parameters (clusters). Additionally, the BIC also penalises a model for using a large number of data points. From the graph, it can be seen that $K = 250$ is the minimum, so it was the number of clusters chosen for training. It has been found that BIC can find the true solution if it is present in the training set as $n \to \infty$ [2, 3]. Nevertheless, this is very rarely the case in practice, and for finite n, BIC can have a substantial risk of selecting a very bad model from the

candidate set [13]. 65,196 descriptors are a relatively small size in comparison to ones found in the literature, sometimes well over 500,000 [9, 7]. Nevertheless, both the AIC and BIC in this experiment yielded a minimum value of 250 clusters, confirming it was a good choice.

Doing the clustering on the Gaussian Mixture Models from 50 to 450 clusters with steps of 50 was the most time consuming part of training, taking a total of 134 minutes.

K-means clustering was then applied on the original SIFT features with 250 clusters and the centroids obtained were used as the visual vocabulary. The original SIFT features (non PCA) were used here in order to improve the algorithm's performance.

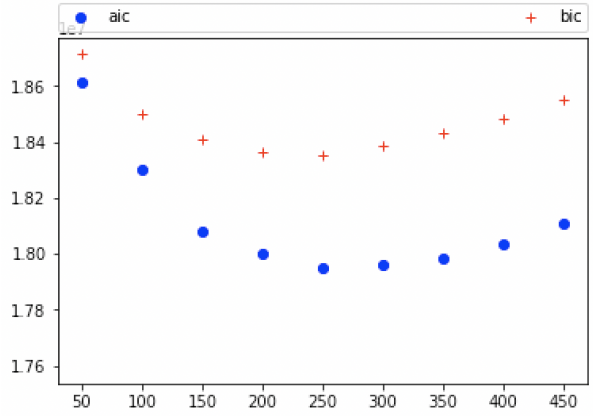


Figure 2: AIC and BIC against number of clusters.

1.4. Histogram Representation

The centroids found in 1.3 were then used to obtain a histogram (bag-of-words) representation of both X_{tr} and X_{tst}. These would then be used to train the classifier in 2.1. The histogram representation using 250 clusters after applying K-means on the training set for two sample images (found in Appendix A.2) of the same class is shown on Figure 3. Bag-of-Words representation corresponding to same two images for $K = \{100, 400, 600\}$ can also be found in Appendix A.2.

Matching the SIFT descriptors to the code-words in the vocabulary was done through the `NearestNeighbours` function from `sklearn`. The different Nearest Neighbour (NN) algorithms available were tested to see how they affected classification performance in 2.1.

Once the vocabulary was obtained, finding the histogram representations for both X_{tr} and X_{tst} took a total of 257 seconds (225 seconds for SIFT detection and computation and 32 seconds for NN matching).

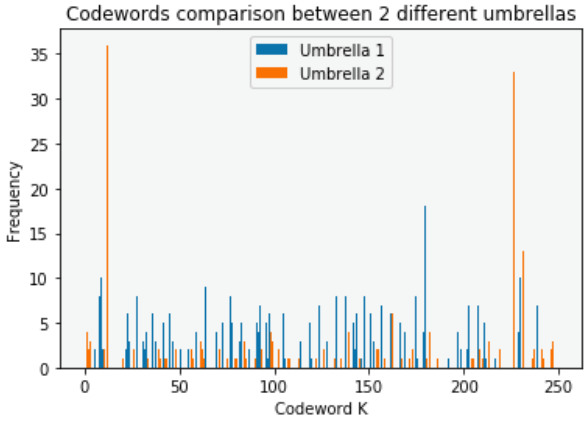


Figure 3: Bag-of-Words representation using 250 code-words for two sample X_{tr} images.

2. Question 2: Random Forest Classifier

2.1. Training and Testing the Classifier

The classifier architecture that was used in this part of the coursework was Random Forest (RF). In order to confirm that 250 code-words would lead to optimum classification performance (refer to 1.3), the algorithm was also trained with vocabularies of size 100, 200, 300, 400, 600. The function used to create the RF was `RandomForestClassifier` from `sklearn` and, initially, the default parameters were used. The percentage accuracy can be seen on Table 1 below, against vocabulary size.

Vocabulary size	Percentage Accuracy
100	65.33
200	65.82
250	66.67
300	66.39
400	66.00
600	60.67

Table 1: RF accuracy for varying vocabulary size

As can be seen, a vocabulary size of 250 resulted in optimum performance. The reason for this being that a smaller vocabulary would not be able to fit the data in an appropriate manner; but, a bigger vocabulary would result in overfitting of the training data set. This falls in line with the results obtained in 1.3.

The next step was to tune the RF hyper-parameter values. For this, Random Search Cross Validation (RSCV) was used. The idea behind Cross Validation (CV), is to separate the training set X_{tr} into N folds and to train the algorithm N times, but, each time, leaving one of the folds out of the training set to use as a validation set. The performance of

the validation set is averaged across the N iterations and gives a good idea of how well the training set modelled the data to avoid overfitting.

To perform hyper-parameter tuning when the algorithm is fed a number of different possible hyper-parameter values, Random Search Cross Validation performs cross validation on a randomly-sampled number of the possible combinations. This was done by using the `RandomizedSearchCV` function from `sklearn`. There were a total of 6480 possible combinations and the different hyper-parameters possibilities used can be found in Appendix B.1. The RSCV was performed with 1000 iterations and 3 folds, namely, 1000 random combinations out of the 6480 were tested and the one that achieved the highest accuracy was used to train the RF. More iterations will cover a wider search space and more CV folds reduce the chances of overfitting, but raising each will increase the run time. The RSCV run-time was 63 minutes for the aforementioned parameters. The sub-optimal values found are specified in Appendix B.1.

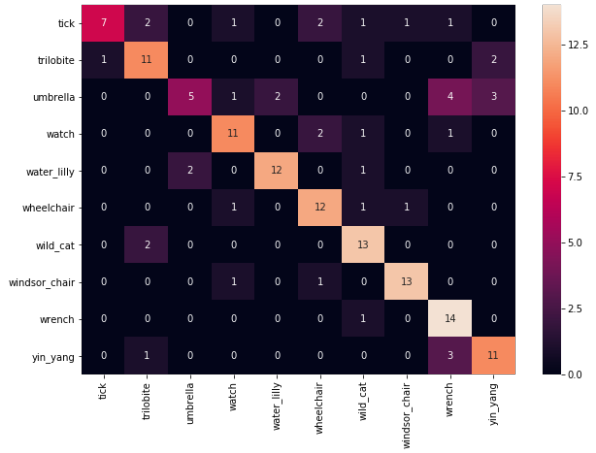


Figure 4: Confusion Matrix for RF classifier with 72.67% accuracy.

The hyper-parameter values found through RSCV were then used to train the RF and the accuracy improved from 66.67% to 72.67%. A significant improvement of 6%. The confusion matrix for the improved RF classifier is shown on Figure 4. Sample successes and failures (false positive and false negative) taken from this matrix can be found in Appendix B.1.

As mentioned in 1.4, all the algorithms available for Nearest Neighbour classification (`balltree`, `kdtree`, `brute`) of SIFT descriptors to vocabulary code-words were tested to observe their impact on performance. Notwithstanding, the histograms produced and therefore accuracy achieved did not change.

Training the RF classifier with the tuned hyper-parameters took 6 minutes once the histogram representations had been obtained and testing took 0.6 seconds. The most lengthy processes were finding an optimal number of clusters to use to represent the data and the hyper-parameter cross validation.

2.2. Exponential χ^2 Support Vector Machine

The χ^2 kernel has been seen to outperform Random Forest classifiers in the literature [9, 5]. In this section, the performance of the RF classifier from 2.1 was compared to a Support Vector Machine with an exponential χ^2 Kernel. Homogeneous additive kernels, such as this one, are designed to compare histograms, and are particularly useful in computer vision since most descriptors are, in fact, histograms (as are the ones in this coursework) [12].

This classifier was trained with the same vocabulary as the RF and performance showed a significant improvement from 72.67% to 78.00%. A change in performance, similar to that seen in [9]. The confusion matrix for this classifier can be seen in Figure 10.

3. Question 3

3.1. Random Forest for Image Representation

In this question, the use of a Random Forest to create a vocabulary was tested. In this method, each leaf of each tree is treated as a separate visual word. Each local SIFT descriptor sampled from an image traverses each tree from the root down to a leaf. Each tree assigns a unique leaf index to the visual descriptor instead of the class label associated during training.

Theoretically, the run-time complexity of the RF was expected to be $O(T\sqrt{D}N\log(k))$, as long as well-balanced trees where created, where T is the number of trees in the forest, D is the descriptor dimension, N is the size of X_{tr} and k is the number of nodes. This is much smaller than the K-means complexity, $O(DNk)$ [8].

After performing the experiment, as expected, computational run-time of creating the RF (3s) was significantly lower to that of a K-means clustering algorithm (367s). Additionally, labeling a descriptor with balanced trees (1s) required less complexity than NN classification with the K-means' centroids (32 seconds), especially, if the maximum depth of the trees is maintained low (limited at 70).

This new bag of words representation was compared with the same RF classifier and the accuracy obtained was 76.00% (refer to C.1 for confusion matrix), which was an improvement on the 72.67% accuracy of the K-means vocabulary. Using RF for vocabulary creations makes use of the class labels (supervised learning) which allows for within class variance to be reduced, unlike with the unsupervised learning in K-means clustering. The number of trees used in the forest was kept at 20. Significantly lower than in the classifier, but, leading to much smaller training run-time (167s for 1000 trees), as run-time complexity increases proportionally with the number of trees, with little improvement on accuracy (76.00% vs 76.33%).

References

[1] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, December 1974.

[2] K. P. Burnham and D. R. Anderson. Model selection and multimodel inference : a practical information-theoretic approach, 2002.

[3] K. P. Burnham and D. R. Anderson. Multimodel inference. *Sociological Methods & Research*, 33(2):261–304, nov 2004.

[4] Caltech. Caltech101. `http://www.vision.caltech.edu/Image_Datasets/Caltech101/`. Accessed: 2020-01-28.

[5] Y.-G. Jiang, C.-W. Ngo, and J. Yang. Towards optimal bag-of-features for object categorization and semantic video retrieval, 07 2007.

[6] D. G. Lowe. Object recognition from local scale-invariant features, September 1999.

[7] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, Oct 2005.

[8] F. Moosmann, E. Nowak, and F. Jurie. Randomized clustering forests for image classification, September 2008.

[9] C. H. nad Harald Sack. Does one size really fit all?: Evaluating classifiers in bag-of-visual-words classification, September 2014.

[10] K. Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, Nov 1, 1901.

[11] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[12] S. Vempati, A. Vedaldi, A. Zisserman, and C. V. Jawahar. Generalized rbf feature maps for efficient detection, 2010.

[13] S. Vrieze. Model selection and psychological theory: A discussion of the differences between the akaike information criterion (aic) and the bayesian information criterion (bic). *Psychological methods*, 17:228–43, 02 2012.

A. Question 1

A.1. Creating Vocabulary

The Akaike Information Criterion is given by:

$$\text{AIC} = 2k - \ln(\hat{L}) \tag{1}$$

where k is the number of parameters (clusters) used and $\ln(\hat{L})$ is the maximum value of the model's log-likelihood.

The Bayesian Information Criterion is given by:

$$\text{BIC} = \ln(n)k - \ln(\hat{L}) \tag{2}$$

where n is the number of data points, k is the number of parameters and $\ln(\hat{L})$ is the maximum value of the model's log-likelihood.

A.2. Histogram Representation

(a) (b)

Figure 5: Sample X_{tr} Images: Umbrella 1 and Umbrella 2

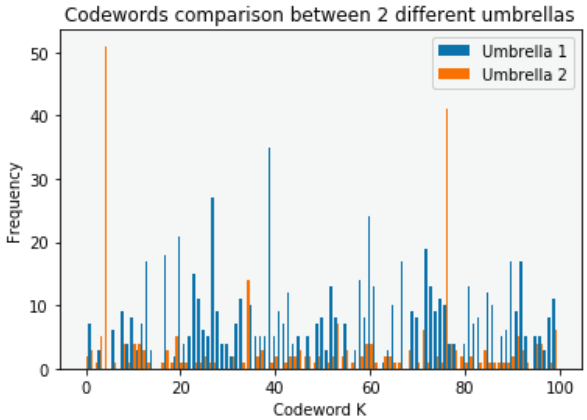


Figure 6: Bag-of-Words representation using 100 codewords for two sample X_{tr} images.

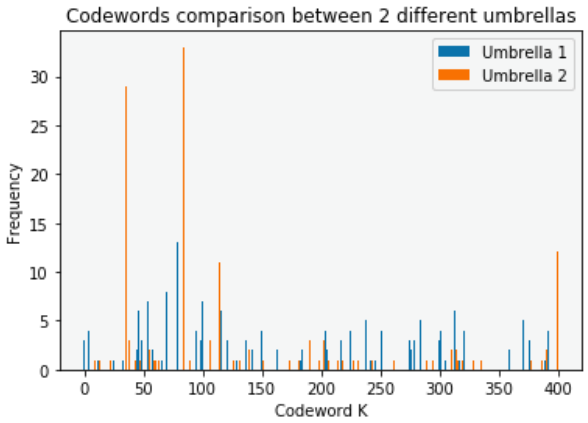


Figure 7: Bag-of-Words representation using 400 codewords for two sample X_{tr} images.

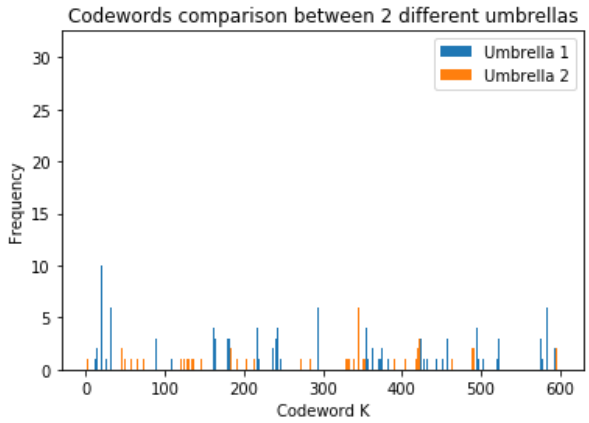


Figure 8: Bag-of-Words representation using 600 codewords for two sample X_{tr} images.

B. Question2

B.1. Training and Testing Classifier

List of possible argument values for `RandomForestClassifier` from `sklearn` used in the RSCV:

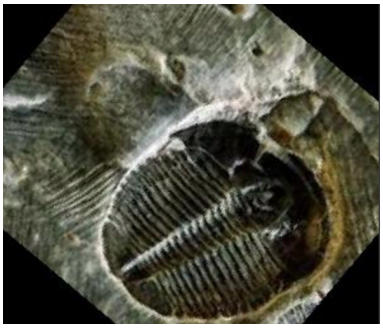
- bootstrap = {True, False}
- max_depth = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, None}
- max_features = {'sqrt', 'log2', 'auto'}
- min_samples_leaf = {1, 2, 4}
- min_samples_split = {2, 5, 10}
- n_estimators = {200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000}

The underlined values were the ones which achieved the highest accuracy during RSCV. The 'bootstrap' argument, dictates the method for sampling data points (with or without replacement). 'max_depth' sets the maximum number of levels in each decision tree. 'max_features' sets the maximum number of features to consider for splitting a node. 'min_samples_leaf' sets the minimum number of data points allowed in a leaf node. 'min_samples_split' sets the minimum number of data points placed in a node before the node is split and, finally, 'n_estimators' sets the number of trees to be used in the forest.

The following figure shows a sample success and two sample failures from the best-performing RF classifier in 2.1.

(a) True Positive

(b) False Positive



(c) False Negative

Figure 9: 9a: Correctly identified 'trilobite'. 9b 'yin_yang' identified as 'trilobite'. 9c 'trilobite' identified as 'wild_cat'
.

B.2. Exponential χ^2 Support Vector Machine

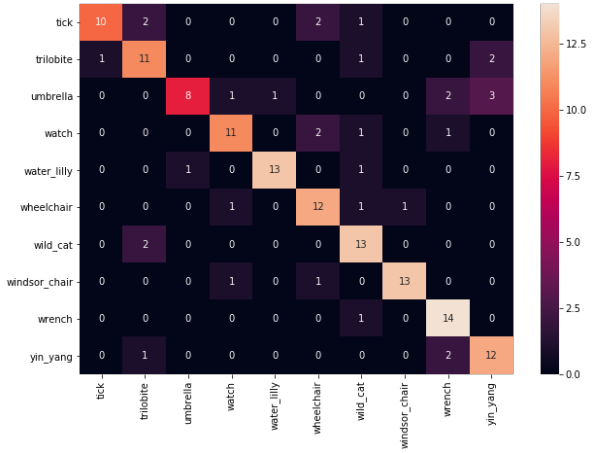


Figure 10: Confusion Matrix for χ^2 kernel SVM classifier with 78.00% accuracy.

C. Question 3

C.1. Random Forest for Image Representation

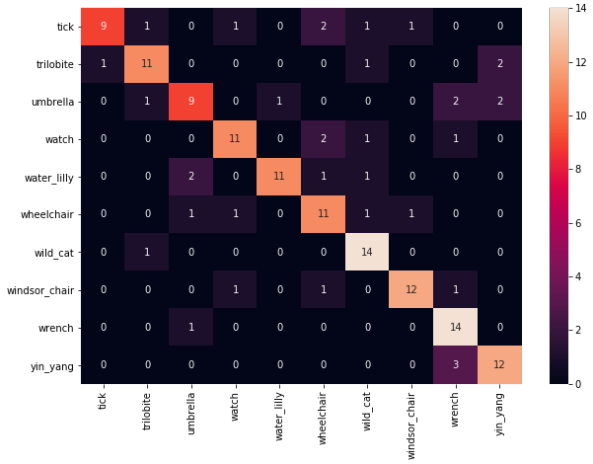


Figure 11: Confusion Matrix for RF vocabulary and RF classifier with 76.00% accuracy.