Machine Learning challenge: Intermediate report

Stefano Claes

December 11, 2020

Contents

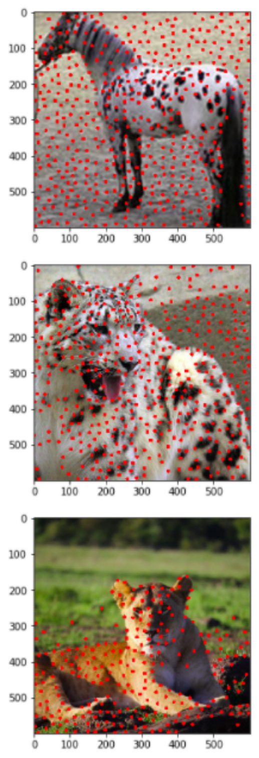
Abstract

Following document is a report on the programming pipeline that has been used to predict several animals species. We will discuss the pipeline from beginning to the end where we do the last classification of data. Note that there is a significant difeerence between the notebooks. THere will be a start_here notebook where you can track the progress that I made and the things I have tried. There will be also another notebook called start_here_academic which is a cleaned up version that is acceptable to read and see a pipeline.

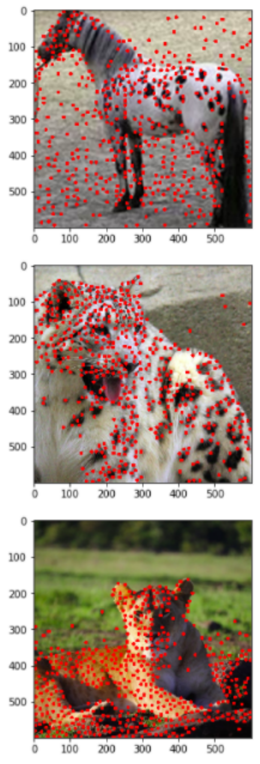
1 Visual bag of words

1.1 Detector

The visual bag of words has been constructed by tweaking the corner detection until an improvement was seen in the scores we got form the classifiers. Unfortunately we have no table with results of this data, which we will do for the final report. What we can show is the visual results we got from the corner detection and the way it enhanced our visual representation on animals.



(a) Shi tomasi corner detection, distance = 20



(b) Shi tomasi corner detection, distance = 13

Figure 1: 2 800 by 800 pictures

As you can see the dots are less scattered all over the place and centered on the animal, this is important because the background contains little or no info about the animal. You can argue that for example having coral in the background would be a good way to pinpoint the animal class fish, hence we have only land animals we do not have to worry about this.[1] Another way we tried to reduce background noise was by cropping the images to 800 by 800. As so we eliminated the fact the Shi-Tomaso would find noise in the background,this is possible because most animals are fitted in the center of the picture. Another thing we eliminated by cropping the image was detecting sharp corners on the black borders which were introduced by resizing and keeping the same aspect ratio.

1.2 Descriptor

As a descriptor we use a tweaked version of DAISY which uses a minimum distance of 13 for the detection of Shi-Tomasi corners. Yet again some numbers miss here which we will contain for the final report.

2 Clustering

For the clustering we use the mini-batch k-means algorithm that has been implemented in sci-kit learn. The way k-means operates and work is as following[1]:
In our set of pictures we will need an amount of pictures that will be addressed as the model of a type of animal. We will use the parameter C to represent them. The amount of model animals, $|C| = k$ where k is our cookbook size will be the amount of clusters we need to fine-tune. Eventually we will call all pictures of animals of our set X. The formula that we need to perform for k-means is the following:

$$min \sum_{x \epsilon X} ||f(C,x) - x||^2$$

The formula f(C,x) will in terms of our data return the nearest cluster of animals where the animal x belongs to. The algorithm we use to cluster our data is described below.

[1]We do have parrots which are often seen in cages.

Algorithm 1: Mini-batch k-means

```
1  Given: k, mini-batch size b,iterations t, dtat set X;
2  Initialize each c ϵ C with an x picked randomly from X;
3  $i \leftarrow 0$ ;
4  for $i$ = 1 to $t$ do
5      $M \leftarrow b$ ;                          // Examples picked randomly form X
6      for $x \epsilon M$ do
7          d[x] $\leftarrow f(C, x)$ ; // Cache the center nearest to x
8        for $x \epsilon M$ do
9            $c \leftarrow d[x]$ ;                  // Get cached center for this x
10           $v[c] \leftarrow v[c] + 1$ ;           // Update per-center counts
11           $\eta \leftarrow \frac{1}{v[c]}$;      // Get per-center learning rate
12           $c \leftarrow (1 - \eta) \cdot c + \eta \cdot x$ ;   // take gradient step
13           ;
```

To translate this algorithm to our own usage in the pipeline we could say the following:
1) X = The data that DAISY extracted from each of our pictures
2) C = The codebook size we use

2.1 Codebook fine-tune

If we look to our data we can make following statements about clustering, there are 4035 pictures of animals and 12 types of animals. Our ideal C or codebooksize would be 12, one cluster per animal. While this being true this raises another question, the animals get identified by 500 points of interest and some pictures are close ups on animals while other embody the entire animal. Is it a good practice to classify them as another cluster or not? A lot can be speculated but ultimately we will rely on the graph we can extract from multiple runs which you can see below.

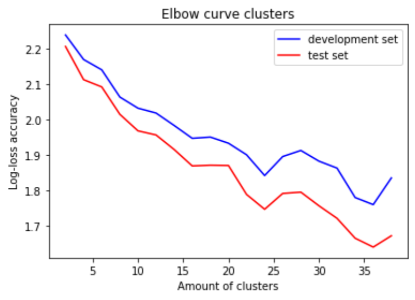


Figure 2: Performance codebook ranging from 2 to 40

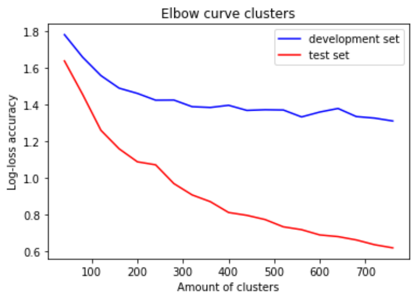


Figure 3: Performance codebook ranging from 40 to 800

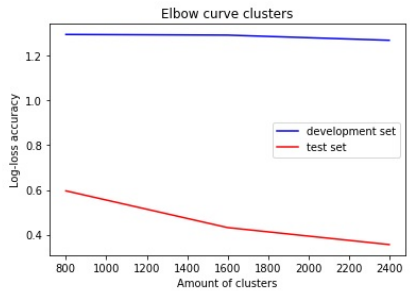


Figure 4: Performance codebook ranging from 800 to 2400

As we can see using the logistic regression classifier and a test-dev split of 0.3 the elbow point occurs between 100 and 200 clusters. When using more clusters we tend to over-fit and just add random noise into consideration.[2]

[2]Because of not understanding this curve very good a codebook size of 2500 was used to get the kaggle score of 1.33590

3 Classifier

As a classifier we used a logistic regression classifier. This gave us a score of 1.23 when using our own metrics and gave 1.33590 on the kaggle scoreboard. For the final report we will address more parameters on this classifier, to continue with the optimal findings of this pipeline we wil use a codebooksize of 150 to generate the cross-val-score and the learning curve.

3.1 Cross-val-score based on parameter C

The parameter C stands for regularization, this means that it will try to void over-fitting. By increasing C to a large number we will decrease this strength and vice versa.

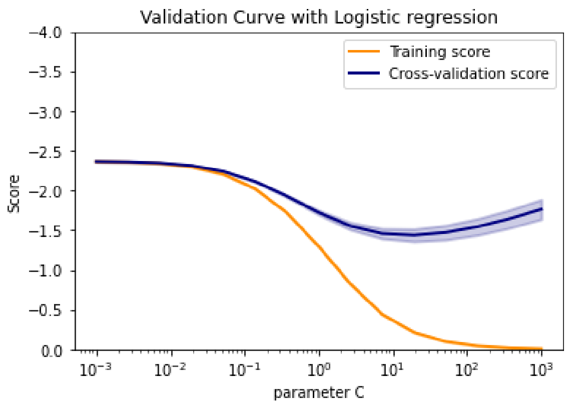


Figure 5: Cross-validation score based on changing parameter C

When looking at the graph we clearly see that the optimum value for the parameter C which is the inverse of the regularization constant is around the value of 10. Optionally we could make a plot that zooms in around this area.

3.2 Learning curve

As last step in the pipeline we would like to know how much samples we would like to use to train our classifier, we do this by using the learning curve that is defined in sklearn.

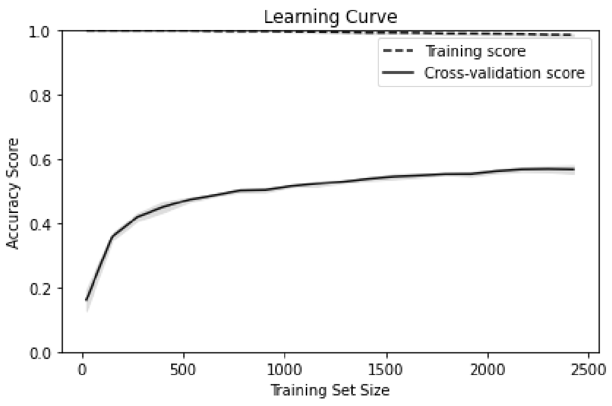


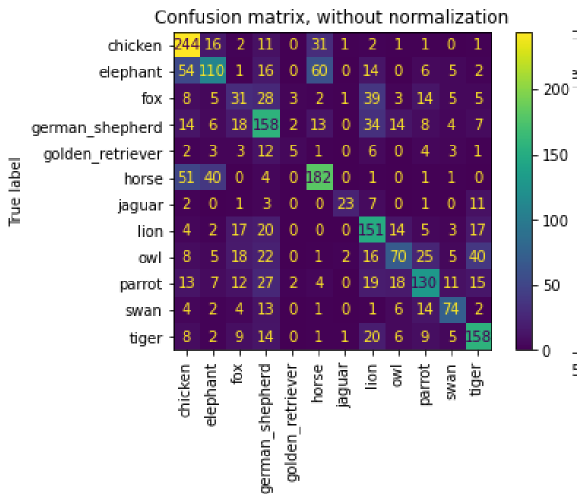
Figure 6: Learning curve with 3-fold test split to show some variance

We can clearly see that after 1500 samples the learning rate slowly starts to dampen out. This means that the ideal split for development and test data sits around 0.4.[3]

[3]The top-score used a development set of 0.8 and testing on 0.2 of the data.

3.3 Results

With the used pipeline we get a score of 1.34 and our confusion matrix on the data that the fit hasn't seen looks like this.



	chicken	elephant	fox	german_shepherd	golden_retriever	horse	jaguar	lion	owl	parrot	swan	tiger
chicken	244	16	2	11	0	31	1	2	1	1	0	1
elephant	54	110	1	16	0	60	0	14	0	6	5	2
fox	8	5	31	28	3	2	1	39	3	14	5	5
german_shepherd	14	6	18	158	2	13	0	34	14	8	4	7
golden_retriever	2	3	3	12	5	1	0	6	0	4	3	1
horse	51	40	0	4	0	182	0	1	0	1	1	0
jaguar	2	0	1	3	0	0	23	7	0	1	0	11
lion	4	2	17	20	0	0	0	151	14	5	3	17
owl	8	5	18	22	0	1	2	16	70	25	5	40
parrot	13	7	12	27	2	4	0	19	18	130	11	15
swan	4	2	4	13	0	1	0	1	6	14	74	2
tiger	8	2	9	14	0	1	1	20	6	9	5	158

Figure 7: Caption

As we can see it is clearly still a problem to recognize golden retrievers, this might be due to the lack of data becouse only 75 golden retrievers where given. A solution might be to increase this by augmentation.

4 Conclusion

Much has been learned about the general concepts of training , but there is still much to learn on the technicalities from the algorithms and the way they mathematically operate. The final report will focus more on data retrieval and mathematical properties and less on exploring the world of sklearn and the many functions it has.

References

[1] D. Sculley: Web-Scale K-Means Clustering
`https://www.eecs.tufts.edu/ dsculley/papers/fastkmeans.pdf`