CM50265 Machine Learning 2

Coursework 1

1. Task 1 – Boosting Image

1.1 Task 1 Introduction

For Task 1, we were provided with three classes of images, cars, bikes and people in real world settings, and we had to implement a working boosting based classifier and validate it using cross-validation methods by observing the results gained from our model's classification predictions.

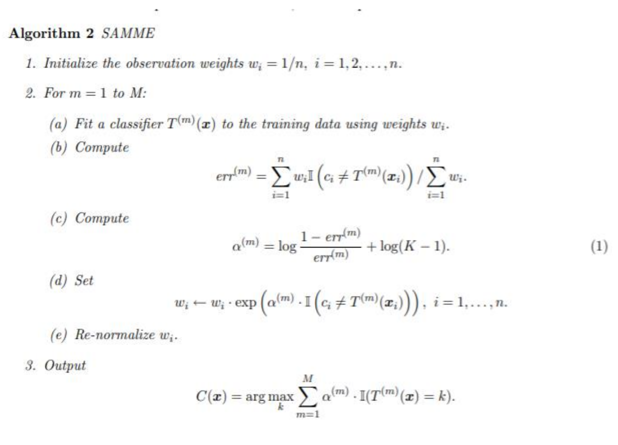
1.2 Processing Dataset

For processing of the image dataset, I read each image and extract their HOG description using functions from the library OpenCV. Assuming that the images were of size 128x128 and their HoGDescriptor having length of 34020.

I had explored some pre-processing steps such as converting the images to Greyscale before extracting their HoG Description, but I had observed no improvement. Using colored images, I got an accuracy of 0.53 whereas using Greyscale images I got 0.48.

1.3 Boosting Classifier

My choice of a boosting classifier that I decided to implement is the SAMME algorithm shown also in *Figure 1* which is based on Adaboost (Ji Zhu et al. 2006, p. 5)



```
Algorithm 2 SAMME
  1. Initialize the observation weights $w_i = 1/n,\ i = 1, 2, \ldots, n$.
  2. For $m = 1$ to $M$:
     (a) Fit a classifier $T^{(m)}(\boldsymbol{x})$ to the training data using weights $w_i$.
     (b) Compute
           $err^{(m)} = \sum_{i=1}^{n} w_i \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right) / \sum_{i=1}^{n} w_i.$
     (c) Compute
           $\alpha^{(m)} = \log \frac{1 - err^{(m)}}{err^{(m)}} + \log(K - 1).$        (1)
     (d) Set
           $w_i \leftarrow w_i \cdot \exp\left(\alpha^{(m)} \cdot \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right)\right),\ i = 1, \ldots, n.$
     (e) Re-normalize $w_i$.
  3. Output
           $C(\boldsymbol{x}) = \arg\max_k \sum_{m=1}^{M} \alpha^{(m)} \cdot \mathbb{I}(T^{(m)}(\boldsymbol{x}) = k).$
```

Figure 1 - SAMME Algorithm

The main difference between SAMME and Adaboost is the change in a model's weight estimation when there are more than 2 classes used. When there are only 2 classes the algorithm performs exactly like Adaboost.

Implementing the SAMME Pseudocode I had to choose a value for M which are the number of 'weak' classifiers which would be combined to represent the final output of the boosted classifier. This hyperparameter M can make a significant impact on the final performance of the Boosting classifier thus I took measures in optimizing it.

By iterating through these classifiers, we compute each classifier's weight using the error rate observed and then update the sample weights w_i.

We can then use each prediction made be all the classifiers, use their pre-computed weight and calculate a final sum which we can then compare for each class to conclude on a prediction.

1.4 Optimizing hyperparameters & Cross-Validation

To accurately evaluate the performance of my implemented classifier I choose to implement 5-fold cross validation. Splitting the complete dataset into 5 sections and each iteration using 1 different of these sections as a test set and the others as a training set. Doing it 5 times I computed the mean accuracy observed as the true evaluation of the performance of the model.

To then optimize the hyperparameter M, I choose to test multiple values ranging from 5 to 155 with increments of 5 giving me the chance to observe how the performance changes with increasing values of M and having multiple choices for deciding on an optimal one.

Testing using these values we can see from *Figure 2* that the performance varies as M increases but the one with the highest accuracy of 0.54 is when M is **155**, thus this would be my final choice for this hyperparameter.

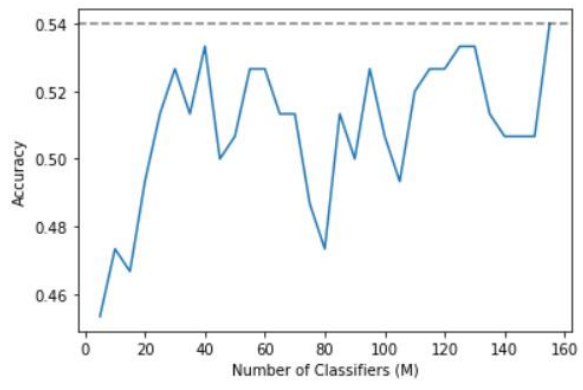


Figure 2 - Cross validation accuracy on different number of classifiers

2. Task 2 - SVM Image

2.1 Task 2 Introduction

For this task I have used the previously processed image dataset with the built-in library functionalities from scikit-learn to find a good performing Support Vector Machine model applied on different kernels.

2.2 Support Vector Machine SVM

Using scikit-learn libraries we can extract the SVM of the train dataset which we can use to make predictions on the test image dataset.

2.3 Kernels

The core aspect of this task is the implementation of different kernels and the evaluation and comparison of their respective performance differences. Kernels play a huge role in SVMs as they can transform the input data into the required form. Identifying a good performing kernel with the appropriate values of their corresponding hyperparameters can have a vast beneficiary impact in the final performance of the SVM model.

I have used the 4 built in scikit-learn kernels, which are the linear kernel, polynomial kernel, Radial Basis Function (RBF) kernel, and the sigmoid kernel and I have also chosen to implement 3 more. These are, the Laplacian Kernel, the Log Kernel, and the Cauchy Kernel (César Souza 2010). I have chosen the Laplacian Kernel because it has great similarity with the RBF kernel except it is less sensitive for changes in the sigma parameter. The Log kernel is described as particularly good for images and the Cauchy kernel because it is mostly used in high dimensional spaces which in our case have.

These 3 kernel functions are represented by *Figures 3,4 and 5* and were then implemented in Task 2 to be tested using cross-validation.

$$k(x, y) = \exp\left(-\frac{||x-y||}{\sigma}\right)$$

Figure 3 – Laplacian Kernel

$$k(x, y) = -log(||x-y||^d + 1)$$

Figure 4 – Log Kernel

$$k(x, y) = \frac{1}{1 + \frac{||x-y||^2}{\sigma^2}}$$

Figure 5 - Cauchy Kernel

2.4 Cross Validation

Further effort could have been made to optimize the hyperparameters that these kernels use but since we only want to compare their performance with their default values I will not go into depth for these exhaustive optimizations.

Comparing the performance for these 7 kernels I was able to identify the best performing one and choose it in my final implementation of the SVM model.

For this task I have chosen to use a 10-fold cross validation method to help me appropriately evaluate the performance differences of the kernels. And because the image dataset is relatively small with only 150 images, I repeated the process 10 times with a newly shuffled dataset to better represented the performance of each kernel.

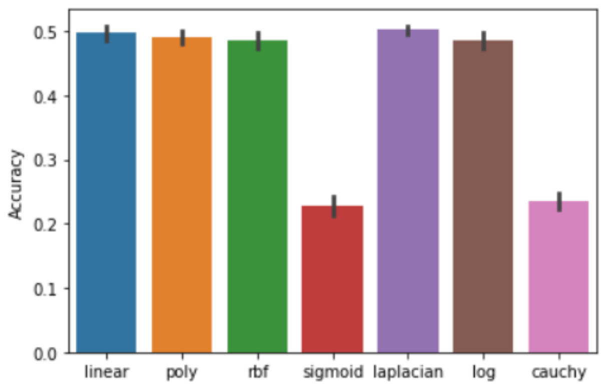


Figure 6 - Kernel Performance on 10-fold cross validation with 10 iterations

From *Figure 6* above we can observe that the best overall performing kernel is the **Laplacian Kernel** with a mean accuracy of **50.2**, thus it is the optimal one for the task and is my final choice for the kernel that I will use for the SVM on the image dataset.

3. Task 3 – Boosting Text

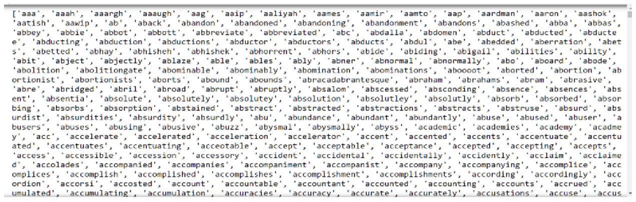
3.1 Task 3 Introduction

For Task 3 we are supplied with a dataset with movie reviews and sentiments and are expected to obtain sentiment analysis using a boosting classifier.

3.2 Processing Dataset

The basis of performing sentiment analysis on text data is by vectorizing the dictionary created from the test set which will in turn be used on the test set and find the best matching result.

To process the movie review dataset, I initially shuffle it to remove any unnecessary bias during training and then using the library NLTK I got a vectorized representation for all the reviews with the overall vocabulary of the dataset. A small sample of the words in the total vocabulary is as shown in *Figure 7* and has a size of 38198 words.



```
['aaa', 'aaah', 'aaargh', 'aaaugh', 'aag', 'aaip', 'aaliyah', 'aames', 'aamir', 'aamto', 'aap', 'aardman', 'aaron', 'aashok',
'aatish', 'aawip', 'ab', 'aback', 'abandon', 'abandoned', 'abandoning', 'abandonment', 'abandons', 'abashed', 'abba', 'abbas',
'abbey', 'abbie', 'abbot', 'abbott', 'abbreviate', 'abbreviated', 'abc', 'abdalla', 'abdomen', 'abduct', 'abducted', 'abducte
e', 'abducting', 'abduction', 'abductions', 'abductor', 'abductors', 'abducts', 'abdul', 'abe', 'abedded', 'aberration', 'abet
s', 'abetted', 'abhay', 'abhisheh', 'abhishek', 'abhorrent', 'abhors', 'abide', 'abiding', 'abigail', 'abilities', 'ability',
'abit', 'abject', 'abjectly', 'ablaze', 'able', 'ables', 'ably', 'abner', 'abnormal', 'abnormally', 'abo', 'aboard', 'abode',
'abolition', 'abolitiongate', 'abominable', 'abominably', 'abomination', 'abominations', 'aboooot', 'aborted', 'abortion', 'ab
ortionist', 'abortionists', 'aborts', 'abound', 'abounds', 'abracadabrantesque', 'abraham', 'abrahams', 'abram', 'abrasive',
'abre', 'abridged', 'abril', 'abroad', 'abrupt', 'abruptly', 'absalom', 'abscessed', 'absconding', 'absence', 'absences', 'abs
ent', 'absentia', 'absolute', 'absolutely', 'absolutey', 'absolution', 'absolutley', 'absolutly', 'absorb', 'absorbed', 'absor
bing', 'absorbs', 'absorption', 'abstained', 'abstract', 'abstracted', 'abstractions', 'abstracts', 'abstruse', 'absurd', 'abs
urdist', 'absurdities', 'absurdity', 'absurdly', 'abu', 'abundance', 'abundant', 'abundantly', 'abuse', 'abused', 'abuser', 'a
busers', 'abuses', 'abusing', 'abusive', 'abuzz', 'abysmal', 'abysmally', 'abyss', 'academic', 'academies', 'academy', 'acadme
y', 'acc', 'accelerate', 'accelerated', 'acceleration', 'accelerator', 'accent', 'accented', 'accents', 'accentuate', 'accentu
ated', 'accentuates', 'accentuating', 'acceotable', 'accept', 'acceptable', 'acceptance', 'accepted', 'accepting', 'accepts',
'access', 'accessible', 'accession', 'accessory', 'accident', 'accidental', 'accidentally', 'accidently', 'acclaim', 'acclaime
d', 'accolades', 'accompanied', 'accompanies', 'accompaniment', 'accompanist', 'accompany', 'accompanying', 'accomplice', 'acc
omplices', 'accomplish', 'accomplished', 'accomplishes', 'accomplishment', 'accomplishments', 'according', 'accordingly', 'acc
ordion', 'accorsi', 'accosted', 'account', 'accountable', 'accountant', 'accounted', 'accounting', 'accounts', 'accrued', 'acc
umulated', 'accumulating', 'accumulation', 'accuracies', 'accuracy', 'accurate', 'accurately', 'accusations', 'accuse', 'accus
```

Figure 7 - Movie Dataset processing dictionary

Some additional processing steps that I took to further help with the performance of the model is with the removal of English stop words which have been thoroughly proven that their inclusion serves no help during sentiment analysis. And additionally, I have added a regular expression tokenizer function to only include words that only contain normal English letters, as numbers cannot have any real correlation between the sentiments.

3.3 Boosting Classifier

Using the same Boosting Classifier as defined in Task 1, I now had opportunity to further optimize the parameter M which are the number of 'weak' classifiers that were be used to make up the complete classifier which will help with the performance of this specific dataset and task.

3.4 Cross Validation

The cross-validation method that I have implemented for this Task is the holdout cross validation where I split the dataset into a test set and a training set.

In order to achieve the best possible representation for the performance of the classifier when a specific value for M is used, I tested each value 10 times each time using a differently shuffled split of the dataset and getting its mean accuracy. Understandably, this will have some overlapping of data in both sets but it should be enough to help me choose a close to optimal value for M.

Using a large range of M values from 2 to 120 with increments of 2 I got the results shown by *Figure 8*. Evaluating these results, we can clearly see that as more 'weak' classifiers are used the performance of the model increases but at a decreasing rate.

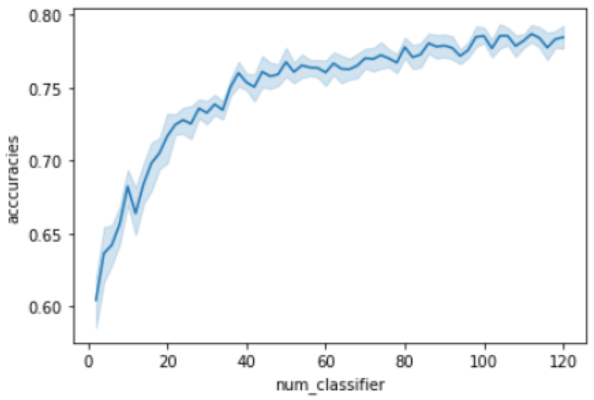


Figure 8 - Cross Validation accuracy for varying number of classifiers

So, in order to be sure that I have covered bigger ranges of values for M, I decided to also test values such as 150, 175, 200, 250, 300, 350, and 400. I then added these to my accuracies dataframe and the results are shown in *Figure 9* below.

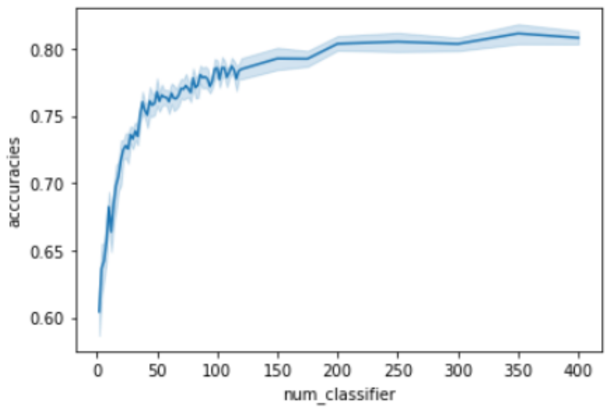


Figure 9 - Complete range of varying number of classifiers validated.

With these results I chose as my optimal value for **M to be 350** as it had the highest mean accuracy of 0.811. I could have tested even higher values because the trend shows that higher classes have positive impact in the accuracy of the model but cross validating such values is very time consuming and overall futile.

4. Task 4 – SVM Text

4.1 Task 4 Introduction

Task 4 uses the previously processed movie dataset on the implemented SVM model and tested on several kernels to identify the optimal one for this specific task.

4.2 Dataset

As previously described, I have processed the movie dataset containing reviews and sentiments such as to extract the vocabulary and the vectors for each movie in the train set that can be used to make predictions on the test set.

I have also used all processing methods as before such as removing English stop words that aid no use for our prediction model and applying a regular expression tokenizer to only include words that only contain letters from the English alphabet.

4.3 SVM Classifier

Again, using the same SVM classifier methods as implemented on Task 2.

4.4 Kernels

For this section I had again chosen to use the same kernels as I had on task 2. The 4 built in Kernels provided by scikit-learn and my 3 implemented kernels, Laplacian, Log and Cauchy. These additional kernel functions are shown by previously shown *Figures 3, 4 and 5*.

4.5 Cross Validation

For my final cross validation approach in testing and identifying the best performing kernel I chose to use the holdout cross validation where we split the dataset into a test set and a training set to a ratio of 20 – 80.

To achieve the best representation of the cross-validation kernel performance I tested each kernel 15 times, each time shuffling and splitting the dataset and storing the accuracy results in a Dataframe.

As shown by *Figure 10* we can clearly identify the best performing kernels, with linear, RBF, Laplacian and log, all being in the range from 0.83 to 0.851.

But, out of these kernel choices, the one that I decided to keep as my optimal one was once again the **Laplacian Kernel** which out of all, scored the highest mean accuracy of 0.85.

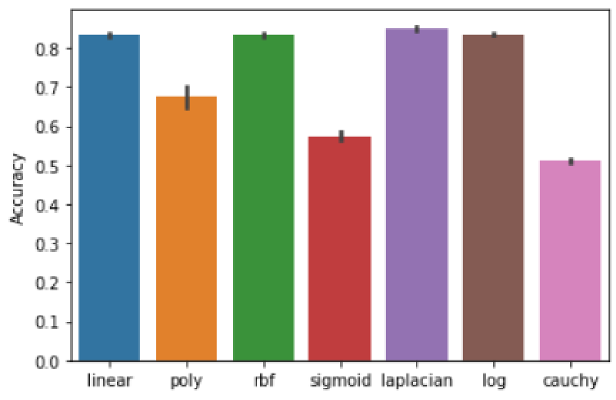


Figure 10 - Kernel Performance on movie dataset on the SVM model

References

Ji Zhu, Saharon Rosset, Hui Zou, and Trevor Hastie *Multi-class AdaBoost* (2006) [online] Available at: https://web.stanford.edu/~hastie/Papers/samme.pdf [Accessed 11 March 2021]

Crsouza.com. 2021. Kernel Functions for Machine Learning Applications – César Souza. [online] Available at: http://crsouza.com/2010/03/17/kernel-functions-for-machine-learning-applications/ [Accessed 13 March 2021].