An implementation of a Support Vector Machine for the task of image recognition

Manol Tonchev

December 2019

1 Introduction

Image recognition is a research field that seems to be growing in popularity over the years. It has wide commercial applications from self-driving cars to facial recognition.

I have a data set containing 10 thousand training images and 1 thousand testing images, each image is described by a 32x32x3 array that holds the 3 color channels for each 32-pixel long side. The training and testing sets are accompanied with corresponding label sets that assign a category to each image. There are 10 categories each one with a thousand training images and a hundred testing images, each one with an exact category. The data set is a sample form a widely used set called CIFAR-10[1].

The goal is to classify the images from the test set as accurately as possible in the 10 categories. I decided to implement a Support Vector Machine. The main task that I need to complete in this paper and in my implementation is to compare the performance of different kernels within the model after some fine-tuning.

2 Method

2.1 Feature Extraction

After loading the images we need to create feature vectors that can be fed into the SVM. We need to plug them into a function that gives us a *histogram of oriented gradients.* This is a feature descriptor widely used in image recognition. The function returns an array with length depending on the size of the image(324-long in our case). This array has float values in each index that essentially position the image on an axis. The array describes an n-dimension vector, where n is the length of the array. Extracting the features is achieved with the use of a for loop that goes over every image in the training and testing sets and, after computing their feature vectors, stores them into an appropriate array (10000x324 and 1000x324). I then normalize the data and move on to classification.

2.2 Classifiers

2.2.1 Support Vector Machine

I decided to implement a Support Vector Machine. It a supervised learning algorithm that is trained using data X_i, Y_i, where X_i describes a feature vector and Y_i is the corresponding class label. Then the model is trained with the task of developing a good function $f(X)$ that gives the correct Y on data that it hasn't been trained on. It should perform well on the processed data, described by feature vectors. The function creates a vector that separates 2 categories from the data, we can use that by separating one versus all and getting looping over to get the right categorisation when dealing with many categories.

3 Results

After implementing a Support Vector Machine I experimented with different kernels. All of the kernels discussed had their C and, where applicable, gamma values experimented with. First off I found that the sigmoid kernel performed quite poorly, achieving accuracy of no more than 50%. The linear kernel performed decently reaching an accuracy of 53% using a C of 0.55. However, the polynomial kernel showed a significant improvement going up to 58.5% accuracy. The C value was set at 0.5 and the gamma was scaled, but it was done without normalizing the data. If the data was normalized it dropped to 51.3%.

This kernel was outperformed by the Radial-basis function kernel. It achieved an accuracy of 60.7% with a C of 4.5 and gamma being scaled. This kernel is the one that performed the best on this testing set. Out of the options within the sklearn API it seems to perform the best. On figure 1 we can observe the confusion matrix for this kernel. The vertical axis is the ground truth labels and the horizontal axis is the predicted categories. We can clearly see that the model under-performs on categories 3. Category 2 is a set of birds. This might be due to the high variance of bird types

Ground Truth	**Predicted Values**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	70	1	6	3	6	2	1	4	7	0
2	4	65	0	1	4	1	3	2	12	8
3	7	1	51	8	5	13	8	3	2	2
4	4	3	15	42	8	13	8	3	2	2
5	3	2	12	12	62	4	3	0	2	0
6	2	0	7	7	7	61	6	7	2	1
7	1	5	10	7	6	4	64	1	1	1
8	2	1	4	11	11	8	2	59	0	2
9	13	11	4	0	1	0	2	0	66	3
10	6	7	1	2	3	2	0	4	8	67

Table 1: Figure 1: SVM, RBF kernel confusion matrix.

or just an irregular complexity in the data set. I noticed other kernels also under-performed on it. Other categories that in general did not perform well are dogs(6) and cats(4). This might be explained by the similarity between the two animals. However, this issue was alleviated by the RBF kernel, as we can see on the matrix.

Ground Truth	**Predicted Values**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	60	4	4	2	4	1	3	4	17	1
2	10	56	2	2	5	2	3	3	8	9
3	8	3	33	6	8	16	13	5	6	2
4	2	4	15	26	12	10	20	6	2	3
5	3	8	12	6	49	6	10	3	2	1
6	1	2	16	13	4	46	8	9	1	0
7	3	8	8	5	8	12	54	2	0	0
8	2	2	6	10	13	6	4	50	2	5
9	24	11	3	1	2	1	2	1	49	6
10	2	12	2	2	3	3	1	4	11	60

Table 2: Figure 2: SVM, Sigmoid kernel confusion matrix.

On figure 2 we can see the confusion matrix of the SVM using a sigmoid kernel with a C value of 1 and a gamma value set to scaling. It had an accuracy of 48.3%. We can see that scores are worse across the board when compared to to the RBF kernel. The sigmoid kernel struggled in particular with the bird and cat categories again. There seems to be a linear improvement from sigmoid to RBF.

4 Conclusion

I found that the RBF kernel performs the best when compared to other kernels. However, there wasn't much space for fine-tuning the models, since the testing set is not that large and a change in accuracy of 0.1% meant a single picture being categorised correctly, meaning that mnor improvements were hard to observe. A potential improvement of my solution would automate fine-tuning and be able to show some more detailed and systematic findings.

References

[1] Benjamin Recht et al. "Do CIFAR-10 classifiers generalize to CIFAR-10?" In: *arXiv preprint arXiv:1806.00451* (2018).