Color Coherence Vectors for Image Classification

Jing Yu Koh (jingyu_koh@mymail.sutd.edu.sg)

October 9, 2018

Abstract

In this assignment for 51.504 Graduate Machine Learning, I attempt image classification on a very limited dataset of CIFAR-10 images using color coherence vectors.

Question 1.

The inputs are the pixels of the image. For each image i, the inputs are represented as $x^{(i)}$, and can be considered as a concatenated vector of pixel values:

$$x^{(i)} = \left(x_{1,1}^{(i)}, x_{1,2}^{(i)}, \cdots, x_{d,3}^{(i)}\right) \in \mathbb{R}^{3d}$$

where $x_{j,1}^{(i)}, x_{j,2}^{(i)}, x_{j,3}^{(i)}$ represent the RGB channels respectively of pixel j. In this case, $d = 32 \times 32 = 1024$, as our images are 32 by 32 pixels.

An alternative way to represent the images is using color coherence vectors [1]. In this case, the colorspace of the image is discretized into n distinct colors, and the coherence between pixel pairs are computed. This results in an input vector

$$x^{(i)} = \left(\alpha_1, \beta_1, \cdots, \alpha_n, \beta_n\right) \in \mathbb{R}^{2n}$$

where n represents the number of distinct colors to discretize to, and is a tunable parameter. **I will answer questions 2-4 with the first representation, and repeat it with the second representation in question 5.**

The labels or outputs of our classifier for image i can be represented as a vector $y^{(i)} \in \mathbb{R}^c$, where c is the number of classes. In this case, since we have 4 classes, $c = 4$. The index of y with the highest value (argmax) is taken to be the class prediction.

Question 2.

Using a vector of all image pixels, I trained it using SGD with batch size of 16 and logistic loss. I start with a learning rate of 1 and decay it over epochs.

Training is stopped when the average of the last 3 losses is below a certain threshold (in my case, I set $\epsilon = 0.01$), or after 1000 epochs. This ensures with a moderate level of certainty that the training has converged. **The result is that training accuracy reaches 0.95, while**

test accuracy is 0.525.

I found that stopping training when the average loss is below a certain threshold generally does not make training accuracy or testing accuracy worse, and allows for shorter training time. This may be because the model has already reached its minima, and additional training does not help (or may even overshoot and exit the minima).

Question 3.

The results of using hinge loss are generally similar to that of logistic loss. I train it using batch size of 16 and initial learning rate of 1. In all my experiments, I fix the random seed, so any random behavior is similar. **The result is that training accuracy reaches 0.975, while test accuracy is 0.6.**

I tried out various initial learning rate values (all of which decay as $\eta_k = 1/(1+k)$). However, the training did not converge faster with any other initial learning rates. This may be because the loss function is easier to optimize compared to other non-convex functions.

Hinge loss appears to be slightly better than using logistic loss. This may be because hinge loss puts pressure on the model to produce more confident scores, improving generalization.

Question 4.

I tried out various values of k. The results don't really indicate any patterns. This may be because the training set is too small to allow for any meaningful patterns.

k	Test Accuracy
1	0.425
2	0.3375
3	0.3375
10	0.425

Table 1: kNN Classifier Results

Question 5.

Another way to perform multi-class classification could be using multinomial logistic regression. In this case, we can represent the results using a vector of c dimension (where c represents the number of classes). The prediction is taken to be the argmax of c.

This performs similarly to the kNN classifier, producing a **train accuracy of 0.7875** and **test accuracy of 0.4125**.

Question 6.

Another method is using color coherence vectors [1]. I implemented this method in Python, and tried it for questions 2-4. The results are in general better than naively using a vector of image pixels. Experiments are conducted with the same hyperparameters as before.

1. For binary logistic regression of bird and cat with **logistic loss**, this method yields a train accuracy of **0.875** and test accuracy of **0.65**. This is significantly better than the previous result of **0.525**.

2. For binary logistic regression of bird and cat with **hinge loss**, this method yields a train accuracy of **0.85** and test accuracy of **0.7**. This is significantly better than the previous result of **0.6**.

3. For kNN with $k = 1$, this yields test accuracy of **0.3625**. This is worse than the original. The other values of k also give lower results for this. It could be possible that CCV loses some necessary information required for kNN classification.

4. For multi-class classification with multinomial logistic regression, this yields a **train accuracy of 0.625 and test accuracy of 0.4375**. This is better than the previous result of **0.4125**.

In general, this approach seems to be better than just considering the pixel outputs. This may be because color coherence vectors encode spatial information, which is necessary for image classification. The only method that performs worse is with

The lower training accuracy may be because the coherence vectors are more general and of lower dimension, and are not linearly separable. The generalizability could be what leads to a better testing performance.

Question 7.

I trained a multinomial logistic classifier using color coherence vectors on the augmented data. This achieved a **train accuracy of 0.55 and test accuracy of 0.425**. The reason for the decrease in accuracy is likely that the distribution of the new images is quite different from the original dataset, making it difficult to fit to the training data.

8. References

[1] G. Pass, R. Zabih, and J. Miller, "Comparing images using color coherence vectors," in *Proceedings of the fourth ACM international conference on Multimedia*, pp. 65–73, ACM, 1997.