Computer Vision Coursework 3

Qi Zhao
Student ID:30571626
qz1y18
qz1y18@soton.ac.uk

Xin Zhang
Student ID:29896665
xz5m18
xz5m18@soton.ac.uk

ABSTRACT

In this paper,it describes the details of our group's CV Coursework 3.There are two people in the group:Zhao Qi and Zhang Xin.In the coursework,we use 4 kind of image features and 5 kind of machine learning methods to build classifiers.And the Programming language we used is matlab.

1. INTRODUCTION

In the coursework,there are mainly three sections. In the first section, according to the coursework specification, we used the "tiny image " feature to process images and choose k-nearest-neighbour classifier to classifier pictures. In the next section,we used a bag-of-visual-words feature based on fixed size densely-sampled pixel patches and an ensemble of 15 linear SVM methods to classify pictures.In the third section,we used 4 kinds of features and 5 kind of classify methods to classify pictures.Through using 10-fold cross-validation ,we can get the average accuracy to compare every classifiers' performance. We will choose the best model to predict the test pictures.

2. COURSEWORK PROCESS

2.1 TASK 1

In task 1,I used "tiny image" feature.In order to transform the original images into tiny image,I build a function named **TinyImages()**.The function of it is simply resizing the original image to 16x16,and normalizing them. After transform training set and testing set into tiny images,I build a function named **knn()**,the function can predict the category for testing set by finding the training image with most similar features.Firsyly,the function compute the distance between inspected data and training data.Then it will sort out these distance value.The classify rule of the function is that the number of which category 's training data closed to the inspected data is the biggest number,which category the inspected data will belongs to.After using **knn()** classify testing set,I output it into file run1.txt.Some details of these functions can be found in **TinyImages.m** and **knn.m.**

2.2 TASK 2

In task 2,according to the coursework specification,I build three functions. There are **BuildVocabularyOfPatch()**, **GetBagsOfPatch()** and **svm_classify()**.

About **BuildVocabularyOfPatch(),** there are 4 input variates.They are the image path, the number of vocabulary,the size of patch and the step length.According to the specification of task2,I set the number of vocabulary to 500,the size of patch is 8 and the step length is 4.In **BuildVocabularyOfPatch(),** firstly it get the images through the image path.Then it will extract patch feature and convet them to 1D array through loop rows and cols. At last, it will use k-means method to cluster these patch features into 500 clusters. And the output is the 500 clusters.

About **GetBagsOfPatch()**,there are 4 input variates. They are the target image paths,the patch vocabulary,the size of patch and the step length.In **GetBagsOfPatch()**,it firstly construct patch features,which is the same as I did in **BuildVocabularyOfPatch()**,and then I assign each local feature to its nearest cluster center and build a histogram to indicate how many times each vocabulary of input was used.Then normalize the histogram.And the normalized histogram is the feature vector of the target images.

About **svm_classify()**.There are 6 input variates.They are training data set,training set labels,testing data set,the list of all kind of the labels and the lambda of svm model and the iterations of svm model.In the function,because there are 15 kind of labels,so I will build 15 linear svm classifier through using vl_svmtrain(which is in vlfeat package) to predict the chance of that data belongs to every category.Every svm model only can estimate in which level the predict image is belongs to the label that the model take charge.After an estimation, every model will return a score about the level. Then I compare these scores and find out which label's classifier gave the biggest value.And this label will be the data's prediction label.

In the task 2,firstly I use **BuildVocabularyOfPatch()** to build patch vocabulary,then I use **GetBagsOfPatch()** to build images' feature.After getting features,I use **svm_classify()** to predict the test images' labels.About the prediction step,in order to choose the lambda and iteration values of svm models,I use 10-fold cross-validation and calculate the average accuracy of classification.After choosing the best lambda and iteration values of **svm_classify**,I used it to predict the test images' labels.

2.3 TASK 3

In the task3,I used 4 kind of image features and 5 kind of classify methods. I compared their performance (accurancy) to find the best classifier to classify the test images. The features are tiny images, a bag-of-visual-words feature based on fixed size densely-sampled pixel patches, a bag-of-visual-words feature based on Dense SIFT and a bag-of-visual-words feature based on Pyramid Histogram of Words. The models are knn classifier, linear SVM, Naive bayes ,non-linear SVM and CNN. At last, I use 10-fold cross-validation and compare the accuracy of them to ensure which model is the best one to classify these images. Therefore, in this section, I build 8 functions. They are **BuildVocabularyOfDSIFT()**, **GetBagsOfDSIFT()**, **BuildVocabularyOfPHOW()**, **GetBagsOfPHOW()**,**NLsvm_classify()**, **NB_classify ()**, **CNN()** and **T10crossvalind()** . Then I will simplely introduce these functions.And the details of these functions can be found in code files.

About **BuildVocabularyOfDSIFT()** , **GetBagsOfDSIFT()** , **BuildVocabularyOfPHOW()** and **GetBagsOfPHOW()** , most of the processes are as same as that of **BuildVocabularyOfPatch()** and **GetBagsOfPatch()**.The main difference of them is that when they extract vocabulary from images, **BuildVocabularyOfPatch()** and **GetBagsOfPatch()** just extract picture patches, About **BuildVocabularyOfDSIFT()**, **GetBagsOfDSIFT()** ,we use function **vl_dsift()** to extract vocabulary,and **BuildVocabularyOfPHOW()**, **GetBagsOfPHOW()** use function **vl_phow()** to extract vocabulary.

The process of **NLsvm_classify()** and **NB_classify()** are almost as same as **svm_classify().**The main difference of them is that **NLsvm_classify()** trains 15 unlinear SVM classifiers (linear SVM with a Homogeneous Kernel Map), and **NB_classify()** trains 15 Naïve Bayes classifiers. The rest of the two functions are very similar to **svm_classify()**.The details can be found in code file NB_classify.m and NLsvm_classify.m .

In **CNN()**,I firstly build input layer, Next, I define the middle layers of the network. The middle layers are made up of repeated blocks of convolutional, ReLU (rectified linear units), and pooling layers. These 3 layers form the core building blocks of CNN.Then I set the the final layers which are typically composed of fully connected layers and a softmax loss layer.Next I combine the input, middle, and final layers.And then I initialize the first convolutional layer weights using normally distributed random numbers with standard deviation of 0.0001. In the next step, I set up the network training algorithm using the **trainingOptions** function. After this step , I use **trainNetwork()** function to train my data.That is all of the **CNN**().

In **T10crossvalind()**,there are 6 input variance,they are the training set,training set's labels,the classifier models , the list of all kind of the labels, and lambda and iterations of svm(if the classifier model is linear SVM or non-linear SVM).In this function, firstly, through using function **crossvalind** ,it generates an index which can divide data into 2 pieces which are training set and testing set.And the ratio of them are 9/1.Then it will train the model selected through training data and use the model we choosed in the input variance to predict the testing data's labels.Through the real labels of testing to check whether the predicted labels are correct.Then calculate the accuracy.Repeat the process 10 times.And output the average accuracy.

In the task 3.Firstly we build another two image features through using **BuildVocabularyOfDSIFT()** , **GetBagsOfDSIFT()** and **BuildVocabularyOfPHOW()** and **GetBagsOfPHOW()**.Then we try to use different features and different classifier models, using 10-fold cross-validation and compare the accuracy of them to estimate which combination is the best one. Then we use the best performance model to predict the data in testing set's labels.

3. RESULT AND ANALYSIS

The table below shows the result of task 3.

Model Combination	Average Accurancy
1.Bag of Dense SIFT and non-linear SVM	0.5613
2.Bag of Dense SIFT and Naïve Bayes	0.3780
3.Bag of PHOW and non-linear SVM	0.6353
4.Bag of PHOW and Naïve Bayes	0.4380
5.Tiny image and non-linear SVM	0.2140
6.Tiny image and Naïve Bayes	0.2453
7.Tiny image and CNN	0.1373

The table shows that Bag of PHOW and non-linear SVM have the best performance.

4. CONCLUSION

In this course work, we use 4 kind of image features and 5 kind of classify models.And based on the result ,there is a conclusion that Bag of PHOW and non-linear SVM have the best performance.I think the tiny images and CNN should have a better preference ,but I do not know why its average accuracy is so low.Maybe when I have a deeper understanding of CNN,I can explain this problem.

5. INDIVIDUAL CONTRIBUTATION

In this coursework, Xin Zhang mainly built the image features. And Qi Zhao mainly built the classifier functions. And the report was written by both Xin Zhang and Qi Zhao.

6. REFERENCES

[1] The VLFeat Authors. 2017. Matlab api. *ACM Trans. http://www.vlfeat.org/matlab/matlab.html*

[2] Piyush Rai.2011. Kernel Methods and Nonlinear Classification. https://www.cs.utah.edu/~piyush/teaching/15-9-print.pdf

[3] A. Vedaldi and A. Zisserman. *Efficient additive kernels via explicit feature maps*. In PAMI, 2011.

[4] A. Bosch, A. Zisserman, and X. Munoz. *Image classifcation using random forests and ferns.* In Proc. ICCV, 2007.

[5] Y. Singer and N. Srebro. *Pegasos: Primal estimated sub-gradient solver for SVM.* In Proc. ICML, 2007.