CSE SJTU

Sketch Recognizing System

SJTU Software Engineering Project

Siyang Zhang
2017/1/7

Contents

Introduction

Project Purpose

Humans have used sketching to depict our visual world since prehistoric times. Even today, sketching is possibly the only rendering technique readily available to all humans. Under this circumstance, it is nature to think about using computer to recognize this most ancient communication symbol. Since nowadays image processing and pattern recognizing has been highly developed, there are a huge numbers of tools and methods to make use of, which makes this work kind of easy. But still, there are challenges and problems to be solved. Sketches are quite different from photos or pictures. They only have simple lines, making it more difficult to extract the features of a certain category. Besides, sketches are not always clearly expressing, which means even people sometimes can misunderstand some sketches. There have already been several papers published discussing how to extract sketch features and train classifiers with state of art methods, including bag-of-words model[3] and fisher vector model[2]. In our project, we are going to implement one of the sketch recognizing algorithms, which is based on the fisher vector model, and build a whole system to extract features, train classifiers, optimize parameters and recognize new sketches.

Project Scope

Our sketch recognizing system includes the whole process to recognize a sketch: feature extracting, classifier training, parameter optimizing, accuracy testing and sketch recognizing. First our system must be based on a certain sketch data set, here in our project we use the TU-Berlin Sketch Dataset[3]. This dataset contains 250 sketch categories and each category contains 80 sketch images. With simple commands, you can extract features from a certain number of categories, build a train set and a test set, train a one-vs-rest classifier, and use the model acquired from the classifier to recognize new sketches. We implemented a simple sketch board, on which you can draw sketches and let the computer recognize what you were drawing. Our system also provides the function to optimize the parameters of feature extractor and classifier.

Document Overview

In the second section, we will discuss the requirement specification, including the functional requirement, non-functional requirement, and domain requirement. In the third section, we will discuss our software design, including the software model, software development tools, architectural design and algorithm design. In the fourth

section, we will discuss our testing method, including testing plan, test-design specification, test-case specification, and test procedure specification. Finally is the conclusion and references.

Requirement Specification

Functional Requirement

1. Collect feature from raw sketch images, build train set and test set for classifier training and accuracy testing
2. Train classifiers and obtain models for sketch recognizing
3. A sketch board on which new sketches can be drown down
4. Recognize new sketches drown on the sketch board
5. Optimize the parameters for feature extracting and classifier training

Non-Functional Requirement

1. The recognizing accuracy should be high enough, at least 0.7
2. The whole system should be cross-platform, which means the whole system should be able to work on all of the main stream systems.
3. The whole system should be robust and user friendly
4. All the computing processes should be able to run fast.

Domain Requirement

For sketch recognizing, the most unique thing is that you need to have a sketch board on which you can draw new sketches. This drawing interface should be robust and easy to use. Besides, the lines and image size should be exactly same as the dataset images, since they will affect the recognizing accuracy.

Software Design

Software Model

Since this system requires a lot of computation, which is quite time consuming, we choose to use rapid prototype model. With the help of RPM, we are able to first build a runnable demo to verify the correctness of our algorithm implementation. After using a small set of data to test the rapid prototype, we will implement it into the full version and test with the full dataset. This can save a lot of time for us, since extracting features for thousands of images and training hundreds of SVMs is quite time consuming, and by testing with small subsets of the full data set is more efficient.

Software Development Tools

To achieve the non-functional requirement of making this sketch recognizing system cross-platform, we use Matlab to develop our system. One of the reasons is that Matlab itself is cross-platform. It can be installed on Windows, Linux and MacOS. So our system is able to run on any platform with Matlab installed. Besides, Matlab is a very powerful tool for scientific computation. Matlab itself provides many powerful functions, and many of them have been optimized so that they can run fast. In addition to this, there are also many hird-party libraries supporting Matlab. In our system, we are using the VLFeat[1] Matlab toolbox, which is not only powerful but also easy to install and use.

Architecture Design

The whole sketch recognizing system contains four separated parts: feature extractor, classifier trainer, parameter optimizer and sketch recognizer. All the four separated parts can work individually. Data is input and output as Matlab data files. First the feature extractor will read raw images and extract the features of the images, output a train set data file and a test set data file. Then the classifier trainer will use the train set data file to train a one-vs-rest classifier, and save the model as a data file. Now you can either use the test set data file and the model data file to test the accuracy of the trained model, or use the sketch board interface to draw new sketches, then the system will return some candidate labels ordered by the possibility. When optimizing the parameters, the feature extractor and the classifier trainer will be called iteratively to test different parameters. You can compare the accuracy results saved in data files to choose the best parameters.

Algorithm Design

Up to now two kinds of models have been used for sketch recognizing, one is bag-of-word, and the other is fisher vector. According to the paper[2], the fisher vector model works better than bag-of-word model[3] on sketch recognizing, so we choose to implement the fisher vector model. Mention that the author of the paper didn't provide any detail about his implementation, no codes, no parameters, so we implement the whole processing algorithm on our own based on the author's description in his paper. Actually in his paper, he used only two paragraphs to descript the whole algorithm. We can't guarantee that our implementation is 100% the same as the author's implementation, but the result we obtained is not so far away from the result provided by the author, so the correctness of the implementation should be alright.

The basic idea of fisher vector is to use dense SIFT to extract descriptors from the raw images, then randomly choose some descriptors to compute GMM parameters, and use the derived parameters to encode the raw images with fisher vector. With the encoded images in place, we use them to train one-vs-rest SVM. Once the SVM is trained and the model is derived, we can use it to recognize new sketches.

In our implementation, we first resize the images in the data set from resolution 1111x1111 to resolution 480x480. Smaller images can reduce the computational complexity. After that, we use the dense SIFT tool provided by VLFeat to extract descriptors from the raw images, which is 128 dimensions. According to the paper, while the binSize takes 24, the best accuracy reaches. However, according to our implementation, the optimal binSize value is larger than 24. We use binSize 32 in our implementation. To further reduce computational complexity, we use the PCA function provided by Matlab to reduce the descriptors from 128 dimensions to 80 dimensions. Mention that this step will consume a lot of memory, so if you want to try the full dataset, make sure your computer has at least 32GB memory. Then we randomly choose 25600 descriptors from all the descriptors extracted from the images, and evaluate the parameters of a 256 cluster GMM with the GMM function provided by VLFeat. Next we use the parameters of GMM to encode all the images using the fisher vector function provided by VLFeat. After encoding the images, we can use the SVM function provided by VLFeat to train one-vs-all SVMs and save the parameters for testing and final sketch recognizing usage. According to our experiment, when we lambda takes 0.01 and epsilon takes 0.0003, the final accuracy reaches the highest. However, the parameters of SVM may change when you change the number of categories. So if you are using a smaller subset of the dataset, you should run the test program to look for your own optimal parameters.

Testing

Test Plan

Our testing mainly contains two parts: the first is to test with the test set, which is derived from the data set, and the second is to test with new sketches drown on the sketch board. Since we were using the rapid prototype model, we first build a rapid prototype. After that we use a small subset of the dataset to test the system. When we are sure that the whole system works well, and the accuracy is acceptable, we will complete the whole system, and use the whole dataset to train the final model, and test it by drawing new sketches on the sketch board.

Test Design

For the rapid prototype testing, we use the first 50 categories. We build training sets with the first 70 images in each category, and testing sets with the last 10 images. For the final testing, we use the whole dataset to train the model, and use the sketch board draw sketches of different objects, and observe the recognizing accuracy.

Test Case

There are basically two test cases in our testing work, one is to find the best parameters for dense SIFT, and the other is to find the best parameters for training SVM. In the first case, we make the parameters for SVM training constant, and change the binSize and stepSize parameters for dense SIFT in each iteration. In the second case, we make the parameters for dense SIFT constant, and change the lambda and epsilon parameters for the SVM in each iteration.

Test Procedure

To test the first case shown above, we first make the parameters of SVM training constant, and we do feature extracting, classifier training and accuracy testing iteratively, and each time we change one of the parameters for dense SIFT, binSize and stepSize. We save the final accuracy and after finishing each iteration, we use Matlab to draw a mesh graph, and we can see intuitively how the accuracy changes as the binSize and stepSize changes. The second case testing is similar, except for the feature extracting procedure only needs to be run for one time. The testing result is shown below. You can see how the final accuracy changes according to parameter changes.

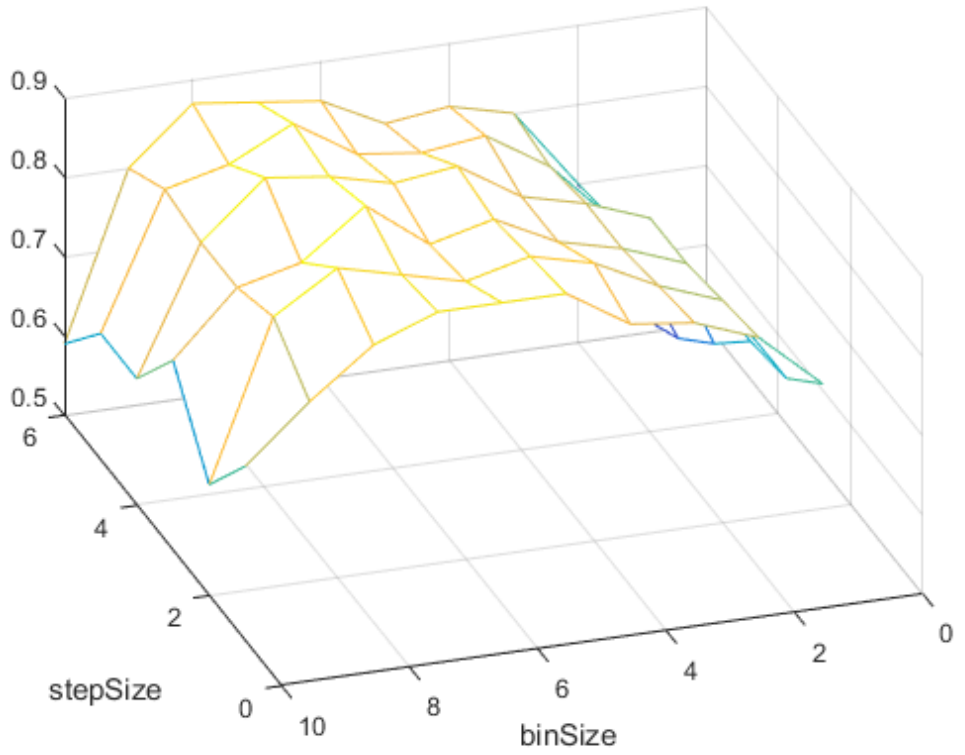

Figure 1 Dense SIFT Testing

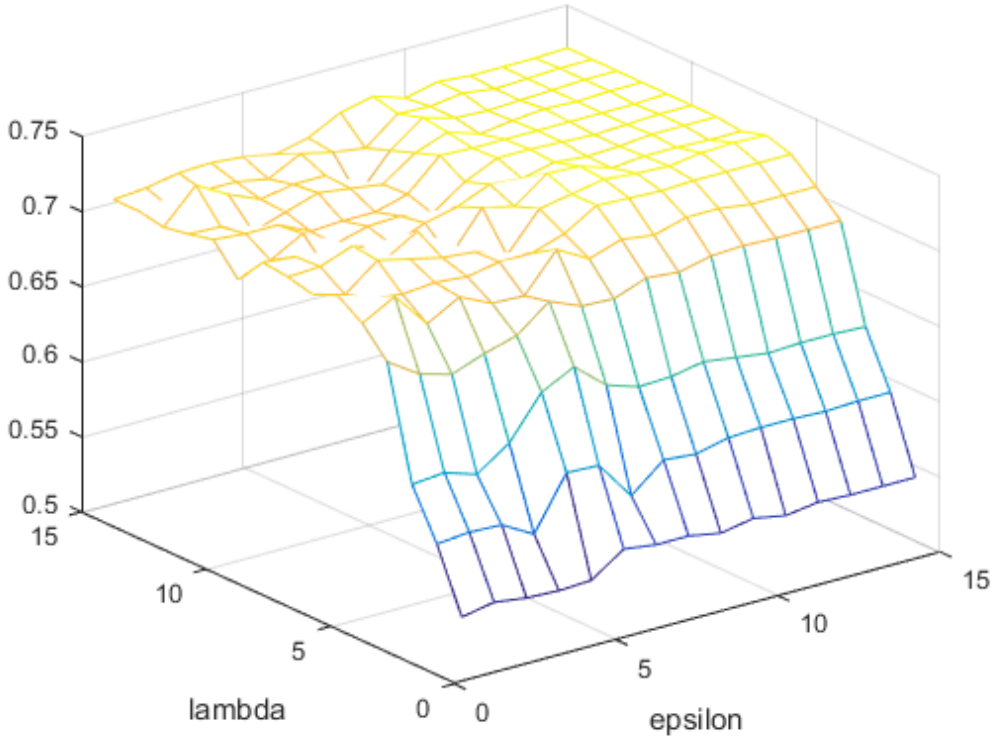

Figure 2 SVM Testing

Conclusion

In the sketch recognizing system we built, we implemented the state of art sketch recognizing algorithm, which is the fisher vector model, and we built a roust system including the whole process to train a sketch recognizing model, from feature extracting, classifier training to model testing. We achieved the accuracy about 0.7, which is high enough for general sketch recognizing. We built the whole system based on Matlab, making it cross platform. Basically we have achieved our goal.

References

[1] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library ofcomputer vision algorithms. http://www.vlfeat.org/, 2008.
[2] R. G. Schneider and T. Tuytelaars. Sketch classification and classification-driven analysis using fisher vectors.ACM Trans. Graph. (Proc. SIG-GRAPH) , 33(6):174:1–174:19, 2014.
[3] Mathias Eitz, James Hays, and Marc Alexa. How do humans sketch objects?ACM Trans. Graph. (Proc. SIGGRAPH), 31(4):44:1–44:10, 2012.