Image Representation And Classification

Yuxin Fan
57881161

December 16, 2018

Introduction

In this project, the goal is to do features of SIFT based image representation and Kmeans+SVM based image classification. My expected process is to extract image features of train images by SIFT, do Kmeans on features and then make histograms to representation each image class. Then, apply tfidf-weight method on it to improve performance and use the weighted result to train a SVM model. Finally, do similar thing on test images to representation them by image features and test the classification performance of the trained SVM model.

Bag of visual words
Bag of visual words (BOVW) is commonly used in image classification. Its concept is adapted from information retrieval and NLP's bag of words (BOW). In bag of words (BOW), we count the number of each word appears in a document, use the frequency of each word to know the keywords of the document, and make a frequency histogram from it. We treat a document as a bag of words (BOW). We have the same concept in bag of visual words (BOVW), but instead of words, we use image features as the "words". Image features are unique pattern that we can find in an image.
Main steps:
1.Extract local features
2.Learn "visual vocabulary"
3.Quantize local features using visual vocabulary
4.Represent images by frequencies of "visual words"

K-means
Minimize sum of squared Euclidean distances between features X_i and their nearest cluster centers M_k.

$$D(X, M) = \sum_{clusterk} \sum_{i\ in\ cluster\ k} (X_i - M_k)^2$$

Support Vector Machine
Support vector machines (SVMs) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

Implementation Details

Do SIFT and Kmeans on training images

Read images one by one and use vl_sift to compute each image's features and the number of features for each image. Store them in matrix and pass features to Matlab build-in function Kmeans to do clustering with a cluster number K=512.

```
imgdir = textread('imgdirs.txt', '%s');
[class_number,~] = size(imgdir);
train_number = 15;
[sift_d_mat, feature_num] = do_sifts(class_number,
   train_number, imgdir);
K = 512;
%do K-Means
[Idx,C]=kmeans(single(sift_d_mat'),K);
```

Build histogram for each training image class

Inder to represent an image, I use histogram to do it. After clustering, an image has lots of features and each feature match to one of the clusters. Count the number of features for each cluster for a image to build a histogram to represent the image.

```
counter_l = 1;
counter_r = 0;
imgs2cluster = zeros(class_number*train_number, K);
for i=1:class_number
for j=1:train_number
number = (i-1)*train_number+j;
counter_r = counter_l + feature_num(number);
[countM, ~] = hist(Idx(counter_l:counter_r-1), 1:K);
counter_l = counter_r;
imgs2cluster(number, :) = countM;
end
end
```

Train a SVM model

Simply lable the image histograms and pass them to libSVM to train a model.
Additional, I try combining images in same class and then pass a 256*K training matrix to SVM to see whether feasible to improve performance. It seems that there are little difference between them, but combine images to a class manually not do well. The mian point is still to get better features and clustering results.
Moreover, I used tfidf concept to get weighted image class features, but still performance not well.

```
train_lable_vector = repmat([1:class_number],train_number
   ,1);
train_lable_vector = reshape(train_lable_vector,[],1);
train = svmtrain(train_lable_vector, train_vector);
```

Build histograms for test images
When build histograms for training date, the matching between features and cluster point is known by the build-in kmeans method. But for test images, I have to find the match relationship by myself. Firstly, compute L2 distance and choose nearest clustering point is directly. Farther more, L2 distance performance not bad, but it is not tricky. I try using **Cosine similarity** to calculate cosine of a feature descriptor and all K cluster centers. Then find the cluster center with minimum angle to the feature. But, for some unknown reasons, it performance very bad. The test result in this case alway treat almost all test images to one class.

Spatial Pyramids
For the physical limit, I build the simplest spatial pyramid with only level 0 and level 1. But it performan very well beyond expectation. In the case, divedi a image into 4 parts, count the number of features of each cluster point for each image. Thus, the training data is N*5K with N*K total image features and N*4K level 1 space informations.

Experiments & Results

I run SIFT and Kmeans with following parameters, and resize each image to 64*64 due to limit of the physical and virtual memory of my personal computer.

```
vl_dsift(X, 'size', 4, 'step', 4);
Kmeans(Y, K);
imresize(img, [64,64]);
```

For BOVW model the result is,

Class number	Train/Test images each class	resize	SIFT	K	Ac
256	15/10	[64,64]	step=4, size=4	500	292/2560 = 11.41%
256	30/10	[64,64]	step=4, size=4	500	349/2560 = 13.63%
256	45/10	[64,64]	step=4, size=4	500	410/2560 = 16.02%
256	60/10	[64,64]	step=4, size=4	500	472/2560 = 18.43%

For spatial pyramids model, the result is,

Class number	Train/Test images each class	resize	SIFT	K	Ac
256	15/15	[64,64]	step=4, size=4	128	1615/3840 = 42.06%
256	15/15	[64,64]	step=4, size=4	128	1648/3840 = 42.92%
256	30/15	[64,64]	step=4, size=4	128	237/3840 = 6.17%
256	45/15	[64,64]	step=4, size=4	128	279/3840 = 7.27%
256	60/15	[64,64]	step=4, size=4	128	331/3840 = 8.62%

Discussions

1. **TF-IDF**, After I compute the histogram for each images, I applid TF-IDF on the histogram, want to make the main feature of each image more strong. My idea is to let the K cluster points be words, let 256 classes of images be the documents to fit TF-IDF concept. However, trained model trend to predict more than half test images to a specific class. I think the reason maybe that I resize the image to [64, 64], it is small. Some common features may be ignored when do SIFT. But still one or more image classes' SIFT result contains lots of these features. I apply TF-IDF on it so that the weight of these features are stronger. When training and testing, every image contains these common features more likely be regard as the specific class.

2. I also tried **normalize** all histogram distribution to see if it worked. But it seems that normalization can influence the process of SVM and decrease accuracy.

3. After apply spatial pyramids on the origin model, it have a great progress at the first execution. Spatial pyramids gives more space informations. Thus, it may performan better than the origin model. But such a improve beyond my expection. I increase train images and execute it again, the result becomes not good as before. I think this high accuracy is a coincidence.

4. As a whole, origin BOVW model is more stable than spatial pyramid model. I think it is because that when the image is rotated, spatial pyramid's space information will misslead the SVM model to make incorrect predictions.