CV HW5

0516006 賴奕善 0516218 軒轅照雯 0510134王泓仁

1. Introduction

Task I: Tiny images representation and nearest neighbor classifier

The "tiny image" feature, inspired by the work of the same name by Torralba, Fergus, and Freeman, is one of the simplest possible image representations. One simply resizes each image to a small, fixed resolution. It works slightly better if the tiny image is made to have zero mean and unit length. This is not a particularly good representation, because it discards all of the high frequency image content and is not especially invariant to spatial or brightness shifts. We are using tiny images simply as a baseline.

The next task for this part is implementation of nearest neighbor classifier. When tasked with classifying a test feature into a particular category, one simply finds the "nearest" training example (L2 distance is a sufficient metric) and assigns the test case the label of that nearest training example. The nearest neighbor classifier has many desirable features -- it requires no training, it can learn arbitrarily complex decision boundaries, and it trivially supports multiclass problems. It is quite vulnerable to training noise, though, which can be alleviated by voting based on the K nearest neighbors. Nearest neighbor classifiers also suffer as the feature dimensionality increases, because the classifier has no mechanism to learn which dimensions are irrelevant for the decision.

Task II: Bag of SIFT representation and nearest neighbor classifier

Bag of words models is a popular technique for image classification inspired by models used in natural language processing. The model ignores or downplays word arrangement (spatial information in the image) and classifies based on a histogram of the frequency of visual words. The visual word "vocabulary" is established by clustering a large corpus of local features.

In this part of the project, we move on to a more sophisticated image representation -- bags of quantized SIFT features. Before we can represent our training and testing images as bag of feature histograms, we first need to establish a vocabulary of visual words. We will form this vocabulary by sampling many local features from our training set and then clustering them with kmeans. The number of kmeans clusters is the size of our vocabulary and the size of our features.

Task III: Bag of SIFT representation and linear SVM classifier

In this part, we present how to train 1-vs-all linear SVMS to operate in the bag of SIFT feature space. Linear classifiers are one of the simplest possible learning models. The feature space is partitioned by a learned hyperplane and test cases are categorized based on which side of that hyperplane they fall on. Since linear classifiers are inherently binary, we have a 15-way classification problem. To decide which of 15 categories a test case belongs to, we train 15 binary, 1-vs-all SVMs. Then, all 15 classifiers will be evaluated on each test case and the classifier which is most confidently positive "wins".

2. Implementation Procedure

Task 1:

- Data Preprocess

```
data_path = 'hw5_data';
categories = {'Bedroom','Coast','Forest','Highway','Industrial','InsideCity','Kitchen' ...
              'LivingRoom','Mountain','Office','OpenCountry','Store','Street','Suburb','TallBuilding'};
num_categories = length(categories);
num_train_per_cat = 100;
num_test_per_cat = 10;
[train_img_paths, test_img_paths, train_labels, test_labels] = ...
img_paths(data_path, categories, num_train_per_cat, num_test_per_cat);

%get tiny images
tiny_test_img = tiny_img(test_img_paths);
tiny_train_img = tiny_img(train_img_paths);
```

Input data and save as matrix.

```
function [train_img_paths, test_img_paths, train_labels, test_labels] = ...
    img_paths(data_path, categories, num_train_per_cat, num_test_per_cat)

num_categories = length(categories);
train_img_paths = cell(num_categories * num_train_per_cat, 1);
test_img_paths  = cell(num_categories * num_test_per_cat, 1);
train_labels = cell(num_categories * num_train_per_cat, 1);
test_labels  = cell(num_categories * num_test_per_cat, 1);

for i=1:num_categories
   catdirname = lower(categories{i});
   images = dir( fullfile(data_path, 'train', catdirname, '*.jpg'));
   for j=1:num_train_per_cat
       train_img_paths{(i-1)*num_train_per_cat + j} = fullfile(data_path, 'train', catdirname, images(j).name);
       train_labels{(i-1)*num_train_per_cat + j} = categories{i};
   end

   images = dir( fullfile(data_path, 'test', catdirname, '*.jpg'));
   for j=1:num_test_per_cat
       test_img_paths{(i-1)*num_test_per_cat + j} = fullfile(data_path, 'test', catdirname, images(j).name);
       test_labels{(i-1)*num_test_per_cat + j} = categories{i};
   end
end
```

Resize images to 16*16.

```
function tiny_image = tiny_img(image_paths)

img_size = 16;
tiny_image = zeros(img_size*img_size, length(image_paths));

for i = 1:length(image_paths)
    I = imread(image_paths{i});

    %resize
    I = imresize(I, [img_size img_size]);
    tiny_image(:, i) = reshape(I.', [1 img_size*img_size]);

    %normalize
    tiny_image(:, i) = rescale(tiny_image(:, i));
end
```

- Use k nearest neighbors classification to make prediction and get accuracy:

```
%k-nearest neighbor
k=8; %37 0.56 0.58
predict_labels = k_nearest_neighbor(k, train_hists, test_hists, train_labels, categories);

%accuracy
match = cellfun(@strcmp, predict_labels, test_labels);
accuracy = sum(match)/(num_test_per_cat*num_categories);
```

Implementation of k nearest neighbors method:

Find k nearest neighbors of test data i with minimum distance, and assign the most frequent label of k nearest neighbors to test data as prediction.

```
function predict_labels = k_nearest_neighbor(k, train_data, test_data, train_labels, categories)

n = size(test_data,2);
predict_labels = cell(n, 1);

for i = 1:n
    %find the index of the k nearest neighbors of test data i
    dist = vl_alldist(train_data,test_data(:, i));
    [d, index] = mink(dist, k);

    %assign the label of test data i to the most frequent label of the k nearest neighbors
    vote = zeros(length(categories), 1);
    for l = 1:k
        cat_idx = find(strcmp(categories, train_labels{index(l, 1)}));
        vote(cat_idx, 1) = vote(cat_idx, 1)+1;
    end
    [v, I] = max(vote);
    predict_labels{i} = categories{I};
end
```

Task 2:

- After reading images, build vocabulary by train dataset.

```
%build vocabularies
vocab_size = 400;
num_samples = 10000;
vocab = build_vocab(train_img_paths, vocab_size, num_samples);
save('vocab.mat', 'vocab');
```

We generated features of train dataset by SIFT, and randomly choose thousands of those to do clustering by k-means. Finally, return the center of cluster as vocabulary.

```
function vocab = build_vocab(image_paths, vocab_size, num_samples)

%get all feature descriptors
des = [];
for i = 1:length(image_paths)
    I = imread(image_paths{i});
    I = rescale(I);
    I = single(I);
    [loc, d] = vl_dsift(I, 'fast', 'step', 8);
    des = [des d];
end

%randomly select feature descriptors to do k means clustering
%then return the cluster centers
s = randsample(size(des,2), num_samples);
sample = single(des(:, s));
[centers, ass] = vl_kmeans(sample, vocab_size);
vocab = centers;
```

- Change the count of sift features to histogram:

```
%use bag of sifts to represent images
train_hists = bags_of_sifts(train_img_paths);
test_hists = bags_of_sifts(test_img_paths);
```

The horizontal axis is the vocabulary and the vertical axis is the count of each vocabulary.

```
function hist = bags_of_sifts(image_paths)

load('vocab.mat');
vocab_size = size(vocab, 2);
n = length(image_paths);
hist = zeros(vocab_size, n);

%for each descriptor find the closest visual word
%build a histogram where the horizontal axis is the visual words and
%the vertical axis is the number of features assigned to each visual word
for i = 1:n
    I = imread(image_paths{i});
    I = rescale(I);
    I = single(I);
    [loc, d] = vl_dsift(I, 'fast', 'step', 8);
    d = single(d);
    dist = vl_alldist2(vocab, d);
    [m, index] = min(dist);
    hist(index, i) = hist(index, i)+1;
end

%normalize the histograms so that they are invariant to the size of imgs
hist = normalize(hist);
```

- Use k nearest neighbors to make prediction.

```
%k-nearest neighbor
k=8; %37 0.56 0.58
predict_labels = k_nearest_neighbor(k, train_hists, test_hists, train_labels, categories);

%accuracy
match = cellfun(@strcmp, predict_labels, test_labels);
accuracy = sum(match)/(num_test_per_cat*num_categories);
```

Task 3:

- After the histogram in task 2 built, classifing it with linear SVM.

Because we have 15 classes, we have to do 15 one vs all classification.

```
for i = 1:num_categories
    temp_label = double(strcmp(train_labels, categories{i}));
    temp_label(find(temp_label==0)) = -1;
    [w, b, info] = vl_svmtrain(train_hists, temp_label, lambda);
    score(i, :) = w'*test_hists + b;
end
```

In prediction, check 15 classifiers and choose the classifier with the highest score.

```
[m, idx] = max(score);
predict_labels = cell(num_test_imgs, 1);
for j = 1:num_test_imgs
    predict_labels{j} = categories{idx(j)};
end
```

3. Experimental Result

Task 1:

```
max_ac =

    0.2267
```

Task 2:

```
accuracy =

    0.5533
```

Task 3:

```
accuracy =

    0.6267
```

4. Discussion

How to find the best parameter?

In task 1, we tried k from 1 to 100 and found out that when k = 7, we got the best accuracy 0.2267.

In task 2, we tried different vocabulary sizes, such 50, 100, 200, 400, 1000 and found out 400 is the best one. We pick random sift descriptors when building vocabulary to decrease computation time, and also we found that the accuracy increased marginally. The reason might be too many features will make it less likely to have clearly separable cluster, thus throwing off the mean of each cluster.

In task 3, the regularization parameter is changeable. We tried from 0.00001 to 1 and get the best result at 0.0549.

5. Conclusion

In HW5, we show results of combining two of three algorithms step by step as the model and evaluate its model accuracy, trying to figure out which model offers a better result and how to enhance the performance to fulfill the requirement.

As we know that extracting image feature points and classification methods are the key to content-based image classification. **SIFT**(Scale-invariant feature transform) algorithm is used to extract feature points, all feature points extracted are clustered by **K-means clustering algorithm**, and then **BOW(bag of word)** of each image is constructed.

Finally, **SVM(Support Vector Machine)** is used to train a multi-class classifier to classify images. The SIFT algorithm has a strong tolerance for scaling, rotation, brightness changes, and noise. The k-means algorithm is simple in structure and fast in convergence. SVM can get better results in the small sample training set and has excellent generalization ability. Experimental results show that the accuracy of image classification in the above methods is better.

6. Work Assignment

report: 0510134 王泓仁、0516006 賴奕善
coding: 0516218 軒轅照雯