Big Data Analysis Midterm Lab Notebook

Rui Wang (ruiw1)

**1. Problem Statement**

In this project, the problem is how to classify images of Plankton, that is, given an image of Plankton, label it and assign it to a category. There are totally 121 classes in Plankton training data set used to train a model for classification.

The goal is to achieve classification as high accurate as possible.

**2. General Design**

1. Data

All the experiment is held on the subset of original Kaggle data set. Because the whole data set is too large for my laptop, it is usually not possible to finish tasks, like extracting features, training models and predicting, within a reasonable time. The experimental data I choose from training data contains 10,000 images in 28 classes. I do cross validation on this subset and 80% of them are my training set, 20% left are testing set. This set is reasonable large for my laptop.

1. Models

What kind of method should I use? After investigating articles and papers in image classification area, I decide to use SVM in this experiment. I use SVM because of image features. I found typical image features are all vectors with float type that are perfect features for SVM.

1. Experiment Process

The basic process is, extract feature from images, then do cross validation on training set, after that train a SVM model on training part and finally test on testing part.

1. Third-party Library

I used Libsvm in my experiment. Libsvm is an open source implementation of SVM. I also use opencv to process and extract features.

1. **Experiments**

**Experiment #1**

**Features:**

Use all pixels to form a feature vector.

**Accuracy:**

Such feature extraction idea produces so large feature space, which makes training process so long. I cannot test this idea on my laptop in reasonable time.

**Error analysis**

All pixels to represent image is a good beginning, but this is too computational expensive. As we all know, if pixels is sampled in a proper way, most of the information that image has would be kept properly. What’s worse, use all pixels actually introduce huge amount of noise and skew the experiments.

**Solution**

So in next step, I consider exploring a way to represent image with less pixels. This way should both reduce the dimension of pixel feature vector and not lose much information of image.

**Experiment #2**

**Feature:**

In order to reduce the computation complex, all the pixels is not a choice anymore. Instead, I reduce every 50 pixels to 3 values, the minimum, maximum and average values among these 50 pixels. This idea help reduce training time.

**Accuracy**

I gain **40.001%** accuracy on 2142 test images. The total number of support vectors is 8289.

**Error analysis:**

As I have mentioned pixels is not a good feature. Pixels are not stable when image rotates, scales, translates. It is also not stable when there is illumination, occlusion and noise. It is possible to improve current method by sampling pixels after rotating, scaling or translating image, but generally speaking pixels themselves are not good representation for images. So samples of pixels, or variant version of combinations of pixels would not help much. More powerful features are needed to stably represent an image no matter image has Rotation, scaling, translation, illumination, occlusion or noise.

**Solution:**

I’d like use SIFT and HOG feature in next several steps.

**Experiment #3**

**Feature:**

Since pure pixels feature is not a good candidate, after studying, I find there is an image feature called Scale-invariant feature transform, SIFT for short. SIFT feature is local feature of image, and it is pretty stable. SIFT can robustly identify objects even among clutter and under partial occlusion, because the SIFT feature descriptor is invariant to uniform, [orientation](https://en.wikipedia.org/wiki/Orientation_(geometry)), and partially invariant to [affine distortion](https://en.wikipedia.org/wiki/Affine_transformation) and illumination changes.

**Training Step:**

1. Extract SIFT features from training set.
2. Apply K-Means to cluster SIFT features.
3. Generate feature vectors for tanning set. Each vector is a histogram which represent distribution of SIFT features of each image.
4. Apply SVM on feature vectors, and get a well-trained model.
5. Test model

**Accuracy**

|  |  |
| --- | --- |
| Number of clusters of SIFT feature | Accuracy (%) |
| 100 | 54.03 |
| 200 | 56.99 |
| 500 | 54.90 |

(2060 images in 28 classes.)

**Error analysis:**

New feature does improve the accuracy of my model. I get 16% improvement, which is exciting. Can I do better? One way I can do is keeping tuning the number of clusters. Another is way is trying other feature. This feature is powerful, and maybe there is still other powerful feature exist. What’s more, if there other powerful features exist, I could combine these features to see if I can predict accuracy.

**Experiment #4**

**Feature:**

There is a feature called Histogram of Oriented Gradient, HOG for short. HOG is used for detect object in image, and it’s widely used in applications such as human face detection, which rely on detect the edges and shapes. This feature looks like a good one for my task since I can consider Planktons as objects in image.

**Training Step:**

In order to get better output, I need to tune several parameters to get HOG feature. I found out that several parameters all affect the HOG vector I get from image. So I use the length of vector to distinguish the different parameters setting. Also, before I start to extract HOG feature, I always resize image firstly.

**Accuracy**

|  |  |
| --- | --- |
| Length of HOG feature | Accuracy (%) |
| 54 | 31.44 |
| 216 | 19.34 |
| 864 | 18.11 |

(2131 images in 28 classes.)

**Error Analysis**

Experiment result shows HOG feature is even worse than bag of word pixel feature. I don’t really know the reason here. But one interesting find is, the short length of HOG feature vector I have, the higher accurate I get. Short length means I don’t so detailed description of image, which means I set pretty large sliding windows in which histograms are computed. I guess this reason large granularity is good because too detailed idea bring noise to model.

**Solution:**

After this experiment, I am thinking about if I can combine HOG feature and SIFT feature, to train a more powerful model

**Experiment #5**

**Feature**

This time I try to combine SIFT feature and HOG feature. Because for each feature, as vectors, they have fixed dimension, this makes it is possible to concatenate directly.

**Training Step:**

1. Compute SIFT feature, which is 128-dimension vector.
2. Cluster SIFT key points and compute SIFT histogram for each image as before.
3. Compute HOG feature, which is a vector whose dimensions depends on parameter setting.
4. Concatenate SIFT histogram and HOG feature, and generate a feature vector.
5. Use concatenated vector to train SVM model.

**Accuracy**

|  |  |  |
| --- | --- | --- |
| # SIFT cluster | Length of HOG feature | Accuracy (%) |
| 20 | 54 | 30.58 |
| 40 | 54 | 30.24 |
| 100 | 54 | 28.54 |

(2060 images in 28 classes)

**Error Analysis**

It turns out combining two features is not necessary helping improve accuracy. I don’t know the real reason here. One way to keep improving is trying bagging strategy, which is a idea that let all the models vote, and each model is trained differently. I won’t try bagging in this project but leave space for me to explore in the feature.

1. **What’s Next?**

Maybe I need to explore more state-of-art model, like deep neural networks, to classify plankton images. Deep learning is a powerful model for image classification. In deep neural networks, at least I don’t need to put much effort on feature engineering, but focus on design the structure of neural networks.