11-775 Large Scale Multimedia Analysis
Spring 2019
HW 2

Sheetal Shalini
Andrew ID : sshalini

Collaboration Statement

1. Did you receive any help whatsoever from anyone in solving this assignment? Whiteboard discussion with Prashant Gupta, Shefali Garg, Karan Saxena, Ashwin Srinivasan

2. Did you give any help whatsoever to anyone in solving this assignment? Whiteboard discussion with Prashant Gupta, Shefali Garg, Karan Saxena, Ashwin Srinivasan

3. Did you find or come across code that implements any part of this assignment ? No

Problem Statement

The task is to perform multimedia event detection (MED) with video features like SURF and CNN features.

Dataset

The dataset contains 2935 videos, with 3 positive events (P001: assembling shelter; P002: batting in run; P003: making cake) and 1 negative event class (NULL). There are 836 training instances and 400 validation instances. The validation set is used to tune hyper-parameters, conduct ablation studies and report the average precision scores. There are 1699 test videos, for which the results have been predicted by the designed model and their accuracies have been compared.

Preprocessing

I downsampled all the 2935 videos to a constant keyframe size of 160x120.

SURF features

For every keyframe of every video, I extracted the SURF features, appended them to get a single SURF feature matrix for a video and dumped them. From each video, I then selected 20% of the frames randomly. Each frame is represented by a SURF vector, which is of a fixed dimension. I stacked all these vectors together into a single CSV file. Mini Batch KMeans clustering has been performed on these frames in order to put the frames into 450 clusters based on their SURF vectors. Each video is then represented as a list of length 450, where each element represents the normalized count of the no.of frames of that video that belong to the respective index cluster. **This vector of length 450 is the final feature vector of a video, passed on for training**. Multiple classifiers have been trained on the feature vectors and their results are reported below.

CNN features

For every keyframe of every video, I extracted the CNN features from the last layer of a pretrained Alexnet model. Since I had downsampled all the keyframes to a constant size of 160x120, each of their CNN features also had the same dimension, so I stacked them depth-wise to get a single matrix representation for a video of size (no.of keyframes, 512), and then dumped them. I then extracted the feature matrix of every video, averaged them through dimension 1 to get a single vector of **length 512, which is the final feature vector of a video, passed on for training**. Multiple classifiers have been trained on the feature vectors and their results are reported below.

MED Pipeline

SURF

The following steps have been performed in the respective files mentioned:

1. **downsample_videos.py** : Downsampled each video such that their keyframes have a dimension of 160x120 each, at a downsampling frame length of 60, and downsampling frame rate of 15.

2. **surf_feat_extraction.py** : The SURF features for every video were extracted and dumped.

3. **select_frames.py** : From each video, a certain ratio of the frames (20%) are selected at random to constitute the SURF feature vectors.

4. **train_kmeans.py** : Mini Batch KMeans is trained to cluster the frames into 450 clusters according to their SURF vectors.

5. **create_kmeans.py** : The SURF feature vectors for each video are created, wherein each element represents the normalized count of the no.of frames from that video that appear in the respective index cluster.

6. **train_svm.py** : The classifiers are trained in this file.

7. **test_svm.py** : The predictions of the classifier models on the test set are generated.

8. **generate_submission.py** : This file compares the values in the 3 files generated by test_svm.py and generates the final submission csv.

CNN

The following steps have been performed in the respective files mentioned:

1. **downsample_videos.py** : Downsampled each video such that their keyframes have a dimension of 160x120 each, at a downsampling frame length of 60, and downsampling frame rate of 15.

2. **cnn_feat_extraction.py** : The CNN features for every video were extracted using the last layer embeddings of Alexnet and dumped. Since every keyframe has the same dimensions, the CNN features of each of those keyframes is stacked depth-wise to get a single matrix representation for a video, of size (no.of keyframes, 512).

3. **create_cnnfeat.py** : The CNN feature matrix for every video is averaged along the first dimension to get a single feature vector of length 512 for a video.

4. **train_svm.py** : The classifiers are trained in this file.

5. **test_svm.py** : The predictions of the classifier models on the test set are generated.

6. **generate_submission.py** : This file compares the values in the 3 files generated by test_svm.py and generates the final submission csv.

CNN + SURF concatenated

The following steps have been performed in the respective files mentioned:

1. **concat_features.py** : This file takes the individual **CNN (512 length feature vector)** and **SURF features (450 length feature vector)** as inputs, and concatenates them to produce a **962 length feature vector**, which is then used further for training. This produces the best results!

Experiments and Results

The following different classifiers have been experimented with their default configurations, and their Average Precision (AP) on the validation set with both the types of extracted features (SURF and CNN), and SURF + CNN features concatenated have been reported in the tables below.

- CPU time taken to create feature vectors : 50mins

Classifier	AP-P001	AP-P002	AP-P003
SVM	0.0222216	0.0296936	0.0389336
KNN	0.206111	0.286198	0.136345
AdaBoost	0.062493	0.114763	0.12232
Gradient Boosting	**0.202796**	**0.636896**	**0.127289**
Gaussian Naive Bayes	0.152923	0.188491	0.0966387
Decision Tree	0.0748053	0.0598584	0.0930458
Random Forest	0.249534	0.305826	0.237666
MLP	0.205769	0.475322	0.0984115
MLP with chi2	0.147514	0.571111	0.0937089

Table 1: Performance of classifiers with SURF features

Classifier	AP-P001	AP-P002	AP-P003
SVM	0.0241388	0.0252032	0.0387091
KNN	0.212122	0.344509	0.108655
AdaBoost	0.0654759	0.509695	0.0855762
Gradient Boosting	0.131563	0.293731	0.164961
Gaussian Naive Bayes	0.0685017	0.178023	0.0764744
Decision Tree	0.0568605	0.114553	0.0746514
Random Forest	0.161445	0.524474	0.194106
MLP	**0.273117**	**0.505111**	**0.18169**
MLP with chi2	0.164915	0.498304	0.168783

Table 2: Performance of classifiers with CNN features

- CPU time taken to train classifiers : Almost instantaneous
- AWS credits not used

Conclusion

The best **MAP values of 0.316038, 0.546828, 0.147019** and a **test set accuracy of 0.86503** were obtained by training an **MLP classifier with chi2 kernel on the CNN + SURF feature vectors concatenated (feature length : 962 [512 CNN + 450 SURF])**. I trained it like a Multi-class classifier, and this showed better results than training on 3 separate binary classifiers because in the latter, the values predicted by the 3 separate classifiers could lie in different ranges and scales. A multi-class classifier overcomes this short-

Classifier	AP-P001	AP-P002	AP-P003
SVM	0.0227336	0.0254036	0.0373699
KNN	0.225083	0.259455	0.135939
AdaBoost	0.0833112	0.163355	0.175835
Gradient Boosting	0.19913	0.334684	0.243176
Gaussian Naive Bayes	0.0856536	0.230846	0.0907019
Decision Tree	0.0741667	0.070007	0.075269
Random Forest	0.190305	0.568233	0.175566
MLP	**0.250795**	**0.595712**	**0.121905**
MLP with chi2	**0.316038**	**0.546828**	**0.147019**

Table 3: Performance of classifiers with CNN + SURF features concatenated

coming by predicting the scores of the 3 classes on the same scale. A simple MLP without chi2 kernel also surpasses the baseline MAP results. And surprisingly, Gradient boosting on SURF features also passed baseline results. This is because ensemble methods reduce the variance of the network, and prevent overfitting.

Future Work

Learning video representations by using the concept of VLAD (Vector of Locally Aggregated descriptors).