Zero-Shot Learning for Human Action Recognition

Nicole Rafidi
nrafidi@cs.cmu.edu

Yuzuko Nakamura
yuzi@cs.cmu.edu

Danny Zhu
dannyz@cs.cmu.edu

Abstract

We demonstrate the performance of zero-shot learning (ZSL) for the task of human action recognition. Instead of training a classifier to map from low-level video-extracted features directly to action labels, we instead train k regressors to map the videos to an intermediate semantic feature space where each action label is a point in this feature space. We classify by choosing the canonical action point that is closest to the query point. This allows us to classify actions for which we have never seen video examples, a generalization that improves the feasibility of human action recognition. Success of this method is evaluated as leave-two-out cross-validation (LTOCV) performance. We describe several further experiments.

1 Introduction

Zero-shot learning (ZSL) enables one to classify input into classes not yet seen in the training data. It does so by making use of a semantic knowledge base as an intermediate step between the features and classes [1]. The main objective of this project is to see whether zero-shot learning yields good performance in human action recognition. Human action recognition is a domain in which there are many possible class labels (actions), making it intractable to provide examples of each action in a training set [2]. This makes it a good candidate for improvement by zero-shot learning.

2 Related Work

Semantic knowledge bases both user-defined (e.g. [3] and [4]) and automatically learned (e.g. [5]) have previously been used as tools for learning and classifying actions. [3] recognizes classes of outdoor scenes by segmenting an image into tiles and classifying each into a visual concept such as water, foliage, and rocks, and then using the resulting grids to perform classification. In order to describe motion, [4] defines a set of operation triplets that each correspond to a way in which specific body parts can move. They then translate a video sequence into a subset of those triplets using domain-specific knowledge. Learning techniques using a bag of visual words approach to the problem such as [5] often use algorithms to learn the semantics of a lower-level, raw visual vocabulary in order to achieve more accurate classification results.

Similar to our approach, these works all make use of an intermediate step to process low-level visual features into something more meaningful before performing classification. However, these approaches are only applied to the domain of training on a set of actions and classifying video containing those actions. By contrast, our approach makes use of a user-defined semantic knowledge base to enable a classifier to correctly classify actions not present in training data.

3 Methods

3.1 Two-Stage Learning

To test the efficacy of ZSL on this problem, we established two steps of classification on the videos in the data sets. First, we found a low-dimensional set of core features that can be used to distinguish

the labels (running, walking, etc). We designed the feature set based on our knowledge of human actions. A potentially better way to have selected this set would have been using text corpus statistics or surveys.

A 12-dimensional vector represents each action. These 12 features are discrete values corresponding to some canonical aspect of an action (e.g. 'Is there horizontal displacement?'). See section 3.6.2 for the complete list of these features.

In the preprocessing stage, we extract low-level features from the videos using the interest points algorithm, a 3-dimensional generalization of Harris points. The points are selected as the local maxima of a particular function on the video which selects for large local variation in all directions [6]. We then compute an optical flow histogram (OFH) descriptor for each interest point. The x- and y-direction flow values in a neighborhood of a point are separately binned and the counts normalized. The two histograms are then concatenated to form the descriptor for each point. A single descriptor represents a whole video with a bag-of-features model. The OFH descriptors from all training videos are clustered by k-means. Any video then is represented by the normalized histogram of the clusters to which its OFH descriptors belong [7].

In the first stage of the classification process, 12 linear regressors are trained to map from the video features to the semantic features (one regressor per feature). In the second stage, each action is represented by a point in this 12-dimensional feature space. To classify a video, the corresponding point is found, and the closest action point is the label. See figure 1 for a summary.

A note on the measurement of distance: there are several ways to measure distance in multidimensional space. To be comprehensive, we used three measures of distance: Euclidean, Manhattan, and Cosine distance. Distance is the sum of these three distance measures to the query point.

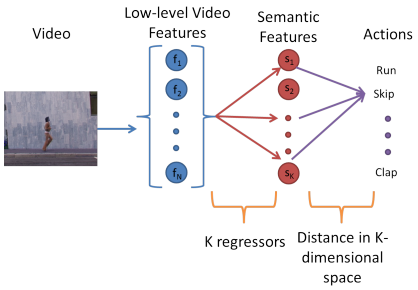

Figure 1: A summary of the two-stage learning process.

3.2 Leave-Two-Out Cross-Validation

To test the effectiveness of this two-step method over direct video-to-label mappings, we can perform leave-two-out cross-validation, and demonstrate the ZSL classifiers ability to distinguish between two previously unseen class labels based on their video data, a task that direct mappings are unable to perform.

First, the 12 regressors are trained on all but two of the actions. then, a video of a previously unseen action is presented. This produces a query point in semantic space. The distance between the query point and the actual held-out action points is measured, and the point is classified. See figure 2 for a summary.

3.3 Statistical Significance

Each action has a set of videos associated with it. With each of these videos used as a query point, we can average the success rate and assign a score for each action. The set of action scores can be compared to chance with a paired t-test to establish significance.

Chance performance at this time is estimated at 50% correct response. In the final version we will establish chance with a Monte Carlo simulation of the data set, by severing the connection between

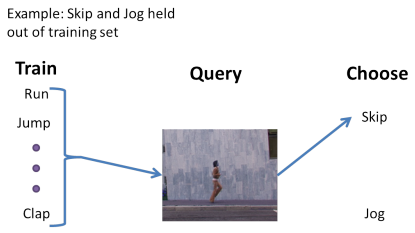

Figure 2: A diagram summarizing the leave-two-out cross-validation used to test zero-shot-learning.

the videos and their semantic feature labels and shuffling the data. Performance of our two-stage algorithm on this shuffled set approximates chance performance. Ideally this will be close to 50%, but it may not be due to imperfections/biases in the semantic feature model.

3.4 Feature Comparison and Selection

To see which semantic features yield the biggest advantage in decoding the action, we can test the LTOCV performance using only one semantic feature in the intermediate set. We can use the knowledge that some features perform better than others to improve overall LTOCV performance. We will feature select over the training fold of the cross-validation, and then use those selected values to evaluate the held out two actions. This will allow us to see if the features that do not do well alone somehow combine to yield good performance, or if they are noise.

3.5 Time Dependency

What is the minimum length of video time needed before we can decode actions with ZSL? Additionally, which features require more or less video time before they can be reliably decoded? In general, for most applications, the less video time needed for the decoder, the better. The semantic feature-level analysis of this phenomenon can help to further refine the semantic model. To answer these questions, we will run our previous experiments with progressively larger action segments from the original videos, and track performance.

3.6 Materials

3.6.1 Data Sets

Data was taken from the following two data sets: KTH, (http://www.nada.kth.se/cvap/actions/) and Weizmann (http://www.wisdom.weizmann.ac.il/~vision/SpaceTimeActions.html)

3.6.2 Semantic Features

The following are the semantic features that we selected, with values and explanations:

1. $Coord\_arms$: (Discrete 0-2) 0 is no coordination, 1 is coordination, 2 is anti-coordination.
2. $Coord\_legs$: (Discrete 0-2) 0 is none, 1 is coordination, 2 is anti-coordination.
3. $Torso$: (Binary) 1 if torso is moving relative to the rest of the body.
4. $Orient\_torso$: : (Discrete 0-2) 0 is that torso orientation is unrelated to the motion, 1 is that the torso is canonically parallel to motion, 2 is that the torso is canonically perpendicular to motion.
5. $Vert$: (Discrete 0-5) amount of vertical displacement of the center of mass.
6. $Horiz$: (Binary) 1 if there is whole body horizontal displacement.
7. $Hand$: (Binary) 1 if the hand is lower than the center of mass, 0 if it is higher.
8. $Speed$: (Discrete 0-5) speed of center of mass.
9. $Limbs$: (4 features, each Discrete 1-5) speed of each of the four limbs.

3.6.3 Coding Libraries

This project was coded using Python. Basic libraries used include, SciPy [8], NumPy [9], OpenCV [10], and The Python Image Library [11]. Ridge regression and cross-validation code taken from Scikit-learn [12].

4 Preliminary Results

Oh Em Gee you guysssss

5 Future Work

At the time of submission, we estimated chance-performance as 50% correct selections in LTOCV. However, this may be an under-estimation of chance performance. A more rigorous way to establish statistical significance of LTOCV results is described in section 3.3. Next we plan to explore which semantic features are the most informative for classifying the videos, by evaluating the performance of classification using each feature alone, as described in section 3.4. Lastly, we plan to do an analysis of how much of the video is necessary for ZSL to succeed. We will run the LTOCV analysis process using progressively larger time segments of the action content sections of the videos to find the minimum amount of time needed. Furthermore, we can look at how the individual features operate on these time scales as well. The full plan is described in section 3.5.

Appendix (Plan of Activities)

The following pieces of code from our initial proposal have all been implemented:

1. Something that extracts features from videos (Danny)
2. Something that maps features to action labels (Nicole)
3. Leave-two-out cross-validation code on the action set (Yuzi)

The following is what we plan to do before the final report (work allocation to be determined):

1. Implement Monte Carlo simulation
2. Test semantic features one at a time to determine which help with classifying
3. Segment video and test ZSL performance on those segments

References

[1] M. Palatucci, D. Pomerleau, G. Hinton, and T. Mitchell, "Zero-shot learning with semantic output codes," in *Neural Information Processing Systems (NIPS)*, December 2009.

[2] R. Poppe, "A survey on vision-based human action recognition," *Image and Vision Computing*, vol. 28, no. 6, pp. 976 – 990, 2010.

[3] J. Vogel and B. Schiele, "Semantic modeling of natural scenes for content-based image retrieval," *Int. J. Comput. Vision*, vol. 72, pp. 133–157, Apr. 2007.

[4] S. Park and J. Aggarwal, "Semantic-level understanding of human actions and interactions using event hierarchy," in *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW '04. Conference on*, p. 12, june 2004.

[5] Q. Zhao, Z. Lu, and H. H. S. Ip, "Action recognition based on learnt motion semantic vocabulary," in *Proceedings of the 11th Pacific Rim conference on Advances in multimedia information processing: Part I*, PCM'10, (Berlin, Heidelberg), pp. 193–202, Springer-Verlag, 2010.

[6] I. Laptev, "On space-time interest points," *Int. J. Comput. Vision*, vol. 64, pp. 107–123, Sept. 2005.

[7] I. Laptev and T. Lindeberg, "Local descriptors for spatio-temporal recognition," in *In First International Workshop on Spatial Coherence for Visual Motion Analysis*, 2004.

[8] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," 2001–.

[9] P. F. Dubois, K. Hinsen, and J. Hugunin, "Numerical python," *Computers in Physics*, vol. 10, May/June 1996.

[10] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[11] PythonWare, "Python Imaging Library (PIL).".

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python ," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.