June 14, 2009

HMMs for Event Recogintion in Video

1 Introduction

While machine vision has produced many techniques that can recognize objects in still images, it has not yet found a good way to recognize interactions between objects in video. The goal of this project, then, has been to do just that – to recognize simple events based on the transitions in the video. Such a system will be useful not only on its own, but it should be capable of supporting many other functions, such as offering a visual/spatial explanation of vocabulary that could be useful for natural language processing. By reconstructing the transitions in an event, the system could support answering common sense questions without being trained on these questions directly (i.e. if a ball flew over a person, did it touch the person? Our software would show that it most likely did not). In addition, event models could be combined to allow for imagining future events and estimating outcomes.

In order to achieve this goal, slightly modified HMMs were used to represent a set of input variables which included simple criteria evaluated from the raw video (i.e. contact between objects). For now, objects are color coded and include two people wearing different color shirts and a bright green ball. Given this input, HMMs were chosen, since events are by their nature sequential and involve transitioning through several states that should be identifiable from the input variables. In addition, the probabilistic nature of HMMs provides a way to compare event models against each other using relative probabilities and helps to account for noise in the videos better than a fixed model would be able to.

2 Technical Details

2.1 Input

As mentioned in the introduction, the input consists of a set of variables calculated from the video. Although some other variable schemes have been tried, the best results have been obtained from the following set of variables:

- focus object
- horizontal movement
- vertical movement
- contact between the focus and each other object
- distance between the focus and each other object
- angle between the focus and the closest other object

These variables are generated outside of the system described here, and are often filtered prior to use (smoothing over very high-frequency components that are caused by noise rather than actual events). This filtering generally improves performance. Additionally, events over a couple of frames are often averaged together to further reduce noise and computational complexity.

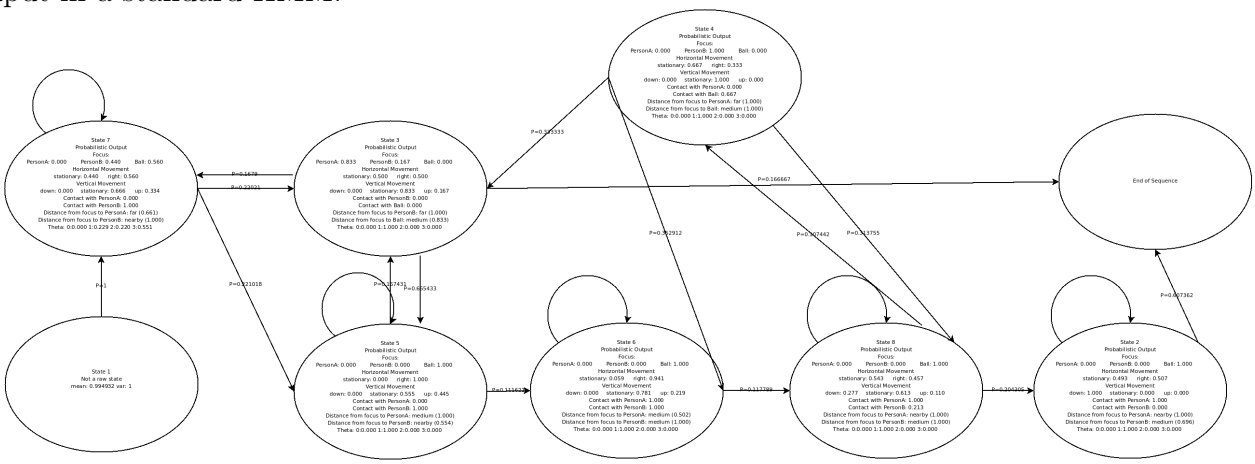
2.2 HMMs

The basic model used was a mostly standard Hidden Markov Model, with transitions between states and emissions of variables from states used to model an event. The main deviation from basic HMMs is that these try to describe multiple variables. This change is achieved by altering the emissions of member states, such that $P(event|state) = P(variable_a|state) * P(variable_b|state) \ldots$, or $P(event|state) = P(variable_a \cap variable_b \cap \ldots |state)$, under an (admittedly imperfect) assumption of independence between the variables. Thus, the overall probability of the HMM states $s_i \in S$ matching a sequence of events $e_i \in E$ is

$$P(E|S) = \prod_i P(e_i|s_i)P(s_i \rightarrow s_{i+1}) \tag{1}$$

$$= \prod_i P(s_i \rightarrow s_{i+1}) \prod_{\text{variables } a} P(e_{ia}|s_i) \tag{2}$$

The HMMs are trained using the forward-backward algorithm (a specific case of the EM algorithm). Emission training is done for each variable output individually as it would be for a single variable otuput in a standard HMM.



2.3 Combined Models

For each event, two HMMs are trained. Given two actors in most of the events, the models learn that one person (i.e. the person in the red shirt) is the doer, while the other person (blue shirt) is the receiver. Of course, the models should be able to recognize either person being the doer. This is achieved by manipulating the input data to produce two copies of each video where the variables for the two people are exchanged. These inputs are then aligned, so that one HMM will contain blue as the doer, while the second will contain red as the doer. Alignment was done by arbitrarily

choosing the first sequence, training an HMM for it, and then finding which sequences best match it (since these sequences should have the same doer and receiver). This unsupervised method works well for capturing the different ways an event can occur, but it lacks an English translation for what it has learned. One of the future variations is to provide event labels to define who the doer, receiver, etc are.

The next step to create a model capable of describing a video was to account for the noise of no event occurring – for any event, there may be people standing around before or after. A combined model should be able to describe the full video, while specifying which components of it contained the model's learned event. This noisy "no-event" model was created to assume uniform probabilities over variable emissions and state transitions – it should match events that do not correspond to the learned event better than the learned event model, but should not match corresponding events as well.

These three models (two with exchanged roles and the "no-event" model) were then combined into a single HMM that allows transitions between the constituent models only at their beginnings and ends. Given an event sequence, this HMM will find the sequence of state and model transitions to best match that sequence (using the viterbi algorithm). For instance, a video of the blue person giving the ball to the red person with people standing around before and after should be recognized as a set of states of "no event", followed by the model for blue as the doer, followed by more "no event" states. In this way, the combined model can identify both whether or not its event was recognized and when that event occurred.

2.4 Event Matching

Originally, all of the event models (for sixteen verbs) were combined into a single combined model. This scheme, however, did not allow for overlap between models, such as approaches contained in gives, throws contained in fly overs, etc. Since we want to identify and allow for overlap in these models, sixteen combined models are trained, one for each verb, and run in parallel. These models can then be compared by their relative probabilities and whether their learned event or "no event" component is active.

Given this data, the models can be compared to see which events are most likely in progress. At any given frame, only a portion of a model may have matched so far (such as the start of a give for video where two people are standing together with a ball), so an event is only considered to have occured if the event has proceeded to completion.

3 Variations

Given this basic framework, a number of variations have been tried to improve performance.

3.1 Gaussian Variable Models

For the basic models described above, continuous variables (such as the distance between objects as a percentage of image size) was discretized. This discretization, however, was arbitrary and susceptible to noise when the true distance was near a discretization boundary. To overcome this problem, gaussian variable models were introduced. Instead of finding the probability of each bin,

as in the discretized case, each state trained a gaussian distribution to model continuous variables. Both the mean and variance were learned (but there was a lower limit on the variance to prevent learning a discontinuous distribution). These models output a probability density function rather than a true probability, but since only relative probabilities are ever used, the effect was similar to that of the old models. Experimentation showed that these models improved performance when applied to the most arbitrary of the discretized variables, distance between objects, but was less effective or showed no improvements when applied to the other variables. Thus, the gaussian models are generally used only for that one variable.

3.2 Regularization

Given that we are working with very small data sets (three videos to train each verb), and a large number of parameters for each model (on the order of 50 per verb), there was ample opportunity for the models to be overtrained and miss near-matches. Suppose all three training videos for give showed no vertical movement while the give took place, which is possible since the give event itself is relatively brief. If there were a frame of vertical movement in test video, none of the states of give would match, and give would not be a possible event for that test video.

In machine learning, a common method for avoiding over-generalization given small data sets is regularization – providing a small probability for each outcome and preventing zero probabilities that might arise from a lack of data. This technique was applied for both emissions from states and transitions between states. Doing so made training more numerically tractable in the early versions of the program (numeric limitations have since been resolved), but was shown to dramatically decrease performance – upon closer inspection, few if none of the regularized state transitions were legitimate, but they allowed transitions that skipped vital parts of the models, even when the regularizing effect (the probability granted to these transitions) was very small. Results were vastly superior for the non-regularized models, which had half the false positive rate of the regularized models. Regularizing within states had less effect and might have slightly boosted performance in some cases, but it was also abandoned for the sake of consistency and clearer models.

3.3 Variable Deltas

One method Sajit has worked on has been finding the important events and frames of videos by detecting where variables change (while filtering out high frequency noise). Since the events his programs output were much less noisy than the raw events, using the output from his program as input to the HMMs for training and testing was a natural method to investigate. Frames without change were disregarded, and variables encoded change rather than static state (i.e. making or breaking contact rather than the existence of contact).

Cleaner filtered input, unfortunately, did not perform any better than the old raw input, and was actually slightly worse. Upon closer inspection, models were not learning the important pieces of events, because these important pieces often only appeared in a couple of frames where the important changes occurred. As a result, the models did not have clear states for these important parts, but instead mixed them with neighboring (noisy) frames. This experiment showed that having sustained important output is important to good learning of events.

3.4 Variable Filtering

I have tried a variety of techniques to select which variables are relevant for a given state/hmm. Logically, such filtering is necessary, since only some variables are relevant for any event. For instance, a give is described primarily by changes in contact (and possibly movement), while a fly over event is characterized by movement and change in angle between two objects. The distinction between relevant and irrelevant input will become more important as the number of inputs is increased; the current joint probability model is not scalable in the number of inputs. Thus far, all filtering methods have decreased accuracy, but they are documented below.

The first method I attempted was to model a couple of inputs as being relevant, still modeled by joint probabiltiy. The others were set to a uniform probability, under the assumption that such a uniform probability would be learned given enough training examples if these inputs are truly irrelevant. These variables could not be entirely ignored, because having different numbers of input variables could drastically effect the overall probability of a model, making comparisons between models impossible. Thus, the probability of an input i (consisting of component variables i_k) being emitted by state s is

$$P(i|s) = \prod_{k \in \mathcal{K}_{relevant}} P(i_k|s) \prod_{k \in \mathcal{K}_{irrelevant}} u(k) \tag{3}$$

where $u(i_k)$ is a uniform PDF over variable k, e.g. if variable k can take the values 0 and 1, then $u(i_k) = .5$.

The relevant inputs were selected by two methods: first, by setting any variables with probabilities near the uniform probability to be irrelevant – thereby making only the variables the model was "sure" about relevant, and secondly by choosing relevant variables by hand. Both methods gave higher error rates than the joint probability method used initially.

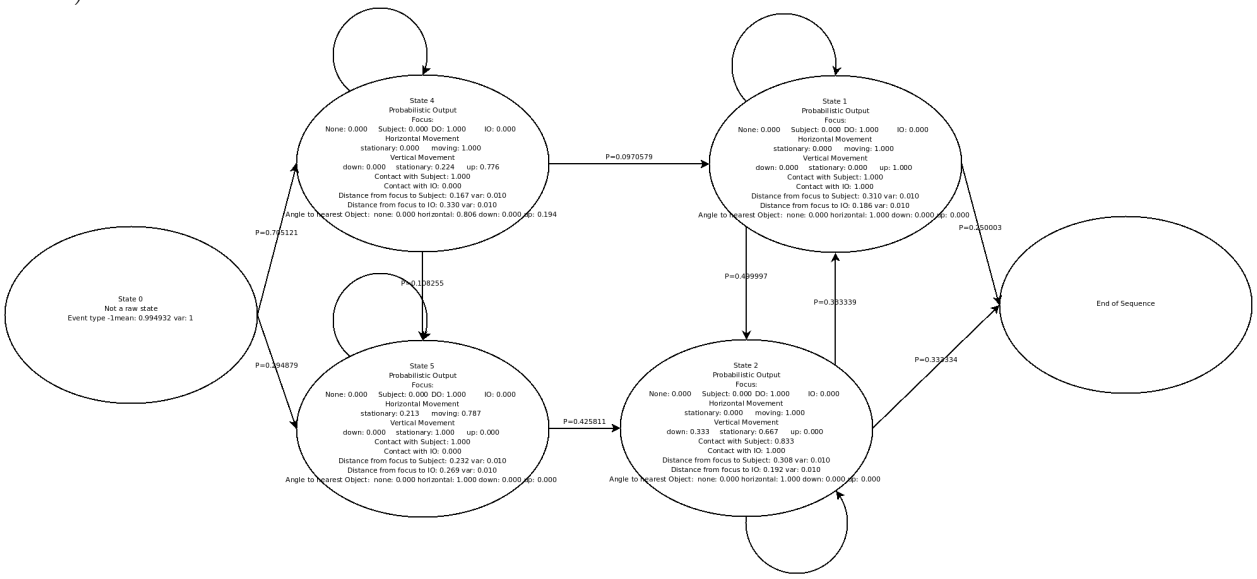
In reality, the irrelevant inputs are not entirely independent of the relevant inputs – for instance, movement follows a certain pattern in give, even if it does not define the event. Similarly, the presence or absence of certain actors is a very strong indicator of the event in the training and test sets, but not in the general case (i.e. jumps tend to have only the person jumping in the dataset video).

3.5 Roles

Early models were comprised of both a model and a "reversed" model, that is, one where "Blue Man" was the subject, and another where "Red Man" was the subject. These two models were independently trained with all of the training data. Such a method was, however, not extensible to more general events where these two people are not the subjects.

To extend the model to handle any assortment of subject, direct object, and indirect object, the notion of roles has been added. Each training video is labelled with a subject, direct object, and indirect object. Once frames have been evaluated and Events created for each video, these Events are then remapped so that the subject, direct object, and indirect object are consistent in the training data. (In the low-level representation, this means that whereas the Red Man previously was encoded as actor 0, the subject is now encoded as actor 0). As a result, only one model is created, and it contains a labelling of actors by subject, etc. Thus, if an event sequence matches a model, its actor 0 is the subject, its actor 1 the direct object, etc. In order to match a test event sequence to a model and correct roles, the sequence is compared for each possible permutation (six

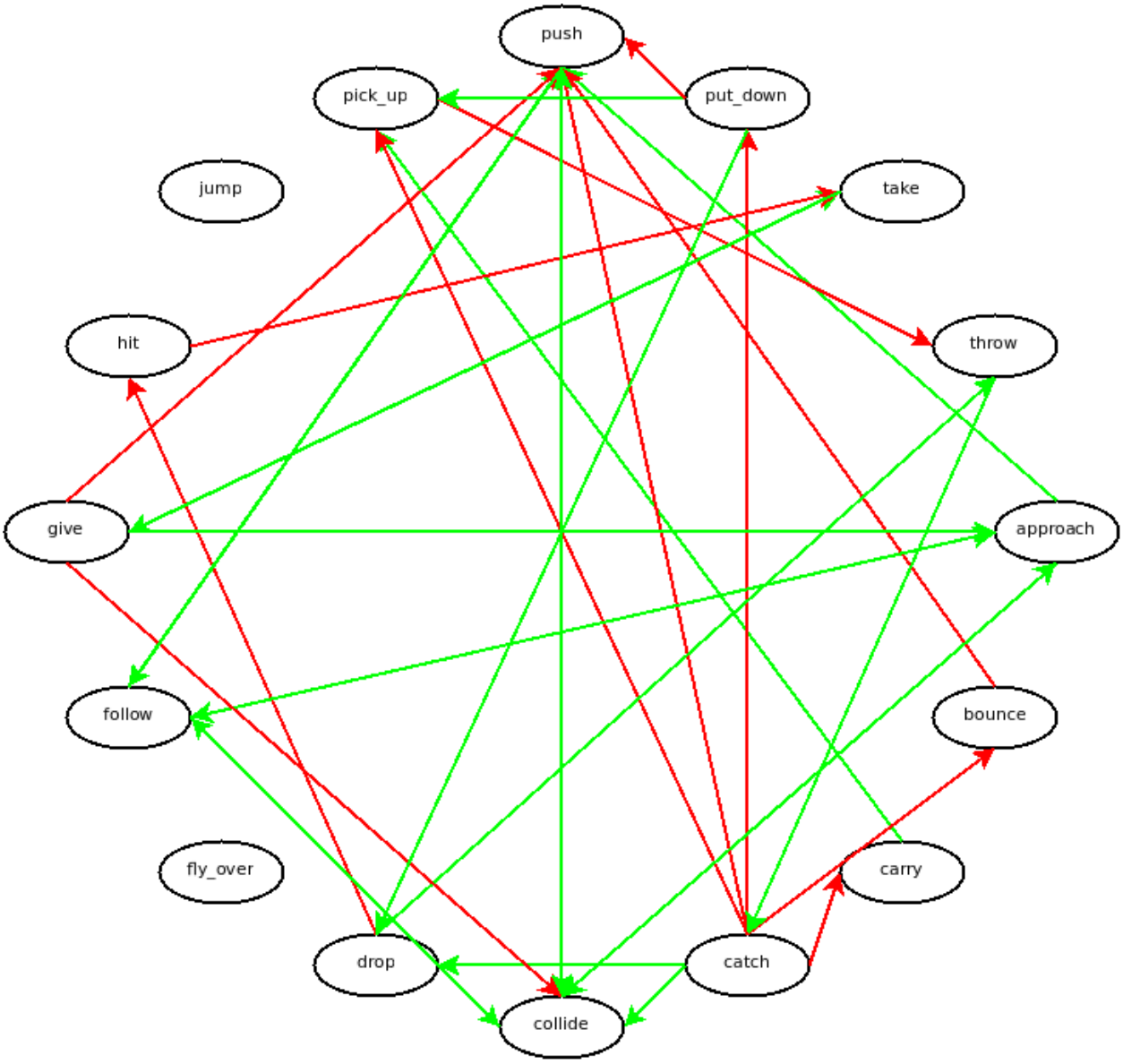
in this case). This method successfully identified the roles for the majority of cases where events fired; nearly all errors were reasonable for a system with no knowledge of actors (i.e. ball dropping a person).



4 Results

4.1 Early Results – before variations

The following chart of connections was generated to show the relations found between verbs. An arrow from verb a to verb b indicates that verb b was identified in videos of verb a, or a contains b. Of these arrows, those marked green are considered correct, while the red arrows are spurious matches.



4.2 Later Results – after variations

5 Usage