Behavior Segmentation: Training, Inference, Features, and Loss Functions

Steve Branson
sbranson@cs.ucsd.edu

Kristin Branson
bransonk@janelia.hhmi.org

August 24, 2010

1 Introduction

We present a behavior detection and segmentation learning algorithm using a structural SVM. Other methods based on per-frame behavior classification tend to erroneously produce many short bouts of behavior. This can be undesirable, as the true objective in many practical applications is related to counting the number of bouts of each behavior. It is common for such methods to apply a post-processing smoothing phase (e.g., using a Hidden Markov Model) to help remedy this problem. On the other hand, since the behavior segmentation problem is one-dimensional (as opposed to the 2D image segmentation problem), algorithms that solve more sophisticated formulizations of the segmentation problem optimally are possible and computationally efficient.

The structural SVM approach learns a function to directly predict a behavior segmentation, and can optimize more appropriate loss functions, such as the number of bouts misclassified. Additionally, it can optimize over more appropriate higher order statistics of the predicted segmentation, such as the mean, standard deviation, min, and max feature response of each bout, and comparisons of feature responses to neighboring bouts. Since the learning problem is convex, performance is predictable and computationally efficient, even when using feature spaces with tens of thousands of features. Since it is relatively easy to adapt the method to different types of loss functions and incorporate complex high-dimensional feature spaces with very few parameters or hand-tuning necessary, the method should be general to a wide variety of different animals or types of behaviors.

2 Training

Let x_i be an observed video sequence of T_i frames, $x_i = \{x_i^1, x_i^2, ..., x_i^{T_i}\}$. Here, each x_i^t is associated with a vector of frame-level features $\vec{\phi}(x_i^t)$. $\vec{\phi}(x_i^t)$ could be a collection of parameters returned by a tracker, a collection of appearance parameters of the t-th frame, or some combination of the two.

Let y_i be the corresponding groundtruth behavior segmentation. A behavior segmentation is a collection of bouts of behavior $y_i = \{y_i^1, y_i^2, ..., y_i^{m_i}\}$, where each $y_i^j = (s_i^j, e_i^j, b_i^j)$ is a bout that begins at time s_i^j and ends at time e_i^j. $b_i^j \in B$ is the label of the j-th bout, where B is a set of mutually exclusive behavior classes (behaviors that cannot occur at the same time).

Our goal is to learn a computationally efficient behavior segmentation function $g : x \to y$ given a training set of n labelled sequences $D = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$. We train the algorithm using a structural SVM, which can be understood as an extension of techniques used for binary classfication to problems with more complicated structures and loss functions. It could also be understood as a restriction of graphical models to certain representations that admit efficient convex learning algorithms.

The main ingredients of the learning algorithm are:

- **A customizable loss function** $\mathcal{L}(y, \hat{y})$, which is a penalty incurred for predicting segmentation $\hat{y}$ when the true segmentation is y

Algorithm 1 Structural SVM Learning

```
INPUT: $D = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$
1: Initialize: $\vec{w}^* \leftarrow \vec{0}, \;\; \forall i \; A_i \leftarrow \emptyset$
2: repeat
3:     for i=1 to n do
4:         Add the most violated constraint to the active set: $A_i \leftarrow A_i \cup \max_{\hat{y}} [f(x_i, \hat{y}) + \mathcal{L}(y, \hat{y})]$
5:     end for
6:     Minimize weights over active sets: $\vec{w}^* \leftarrow \arg\min_{\vec{w}} \|\vec{w}\|^2 + C \sum_{i=1}^{n} \left( \max_{\hat{y} \in A_i} [f(x_i, \hat{y}) + \mathcal{L}(y, \hat{y})] - f(x_i, y_i) \right)$
7: until Until convergence
OUTPUT: $\vec{w}^*$
```

- **A customizable feature space** $\vec{\psi}_{seg}(x, y)$ that converts frame-level features $\vec{\phi}(x_i^t)$ into a more semantically meaningful representation that is a function of the predicted segmentation y
- **A score function** $f : x, y \to \mathcal{R}$ that is high when y is a good segmentation for x and low when it is a bad segmentation. For computational purposes, f is parameterized by a set of linear weights $\vec{w}$

$$f(x, y) = \vec{w} \cdot \vec{\psi}_{seg}(x, y) \tag{1}$$

- **An inference algorithm** that is capable of efficiently finding the highest scoring segmentation out of the space of all possible segmentations

$$g(x) = \arg\max_{y} f(x, y) \tag{2}$$

- **A learning algorithm** that finds the optimal set of parameters $\vec{w}$ over the training set D by solving

$$\vec{w}^* = \arg\min_{\vec{w}} \|\vec{w}\|^2 + C \sum_{i=1}^{n} \left(\max_{\hat{y}} [f(x_i, \hat{y}) + \mathcal{L}(y, \hat{y})] - f(x_i, y_i) \right) \tag{3}$$

The objective in Equation 3 is minimized if for every training example x_i, the score of the true segmentation y_i is higher than the score of any other segmentation $\hat{y}$ by at least $\mathcal{L}(y, \hat{y})$. This is called margin-scaling, and has some theoretical justifications based on large-margin theory. For the 0/1 loss $\mathcal{L}(y, \hat{y}) = 1[y \neq \hat{y}]$, the objective is identical to a multiclass SVM.

Equation 3 is convex in $\vec{w}$ because it is the maximum of a set of affine functions. This holds for any arbitrary choice of $\mathcal{L}(y, \hat{y})$ and $\vec{\psi}_{seg}(x, y)$ (as long as they are computable). The main caveat is that it involves taking the max over an exponentially large space of segmentations. In practice though, most of these segmentations are similar or redundant. The optimization algorithm that the SVM^{struct} package is a delayed constraint generation algorithm that iteratively samples new segmentations by searching for the most violated constraint

$$\bar{y} = \arg\max_{\hat{y}} [f(x_i, \hat{y}) + \mathcal{L}(y, \hat{y})] \tag{4}$$

The basic method is similar to the pseudo-code shown in Algorithm 1. Due to the inherent structure in related segmentations, the number of segmentations that need to be sampled in practice is small. In our experiments, we typically gain another bit of precision toward reaching the global optimum in a constant number of iterations (in our case 50), which is consistent with optimization theory. This factor of 50 depends on the amount of structural similarity existing in the labels y.

3 Loss Function

We use the bout-level loss function

$$\mathcal{L}(y, \hat{y}) = \sum_{(s,e,b) \in y} \frac{\ell^b_{fn}}{e - s} \left(\bigcap_{\hat{y}, \hat{b} \neq b} (s, e) \right) + \sum_{(\hat{s},\hat{e},\hat{b}) \in \hat{y}} \frac{\ell^b_{fp}}{\hat{e} - \hat{s}} \left(\bigcap_{y, b \neq \hat{b}} (\hat{s}, \hat{e}) \right) \tag{5}$$

where $\bigcap_{(\hat{s},\hat{e},\hat{b})\in\hat{y},\hat{b}\neq b}(s,e)$ is the number of frames in $\hat{y}$ intersecting with (s,e) with different behavior class $\hat{b}\neq b$, ℓ^b_{fn} is the cost for missing a bout of class b, and behavior $\ell^{\hat{b}}_{fp}$ is the cost for incorrectly detecting a bout of class $\hat{b}$. This loss function softly penalizes predictions where the start or end of the bout is slightly incorrect. On the other hand, it is very different than a per-frame cost, due to the addition of a false positive cost and because it penalizes bouts of different duration equally. As a result, $\mathcal{L}(y,\hat{y})$ heavily favors smooth segmentations that can explain the observed data using a small number of bouts.

A generalized version of $\mathcal{L}(y,\hat{y})$ that supports arbitrary confusion costs can be written as

$$\mathcal{L}'(y,\hat{y}) = \sum_{(s,e,b)\in y}\sum_{b'\in B}\frac{\ell^b_{\hat{b}}}{e-s}\left(\bigcap_{\hat{y},\hat{b}=b'}(s,e)\right) + \sum_{(\hat{s},\hat{e},\hat{b})\in\hat{y}}\sum_{b'\in B}\frac{\ell^{\hat{b}}_{b}}{\hat{e}-\hat{s}}\left(\bigcap_{y,b=b'}(\hat{s},\hat{e})\right) \tag{6}$$

where $\ell^b_{\hat{b}}$ is the cost of predicting a bout of class $\hat{b}$ when the true class is b.

4 Score Function

We can decompose the segmentation score function $f(x,y)$ in terms of bout scores and transition costs, summing over each bout in the segmentation:

$$f(x,y) = \sum_{(s^j,e^j,b^j)\in y}\left[f_b(x^j,s^j,e^j) + h(b^j,b^{j+1})\right], \qquad \text{Segmentation Score} \tag{7}$$

$$f_b(x,s,e) = \vec{w}_b\cdot\vec{\psi}_{bout}(x,s,e,b)], \qquad \text{Bout Score} \tag{8}$$

$$h(b^j,b^{j+1}) = \vec{\lambda}_{b^j}\cdot\vec{\psi}_{trans}(b^j,b^{j+1}), \qquad \text{Bout Transition Score} \tag{9}$$

Here, $f_b(x,s,e)$ is a bout scoring function that is high when a bout of behavior of class b is likely to occur between time frames s and e. $\vec{\psi}_{bout}(x,s,e,b)$ is a bout-level feature space, which is customizable (see next section). $h(b^j,b^{j+1})$ is a (usually negative) score associated with transitioning from behavior b^j to behavior b^{j+1}. Setting $\vec{\psi}_{trans}(b^j,b^{j+1})$ to a length $|B|$ vector of zeros, where the b^{j+1}th entry is -1 allows $\vec{\lambda}_b$ to be interpreted as a vector of behavior transition costs.

Algorithm 1 will jointly learn all bout score parameters $\vec{w}_b$ and transition costs $\vec{\lambda}_b$ if we write the expressions for $\vec{w}$ and $\vec{\psi}_{seg}(x,y)$ as

$$\vec{w} = [\vec{w}_1,\vec{\lambda}_1,\vec{w}_2,\vec{\lambda}_2,...,\vec{w}_{|B|},\vec{\lambda}_{|B|}] \tag{10}$$

$$\vec{\psi}_{seg}(x,y) = [u_1,v_1,u_2,v_2,...,u_{|B|},v_{|B|}] \tag{11}$$

$$u_k = \sum_{\substack{(s^j,e^j,b^j)\in y\\ b^j=k}}\vec{\psi}_{bout}(x,s^j,e^j,b^j), \qquad v_k = \sum_{\substack{(s^j,e^j,b^j)\in y\\ b^j=k}}\vec{\psi}_{trans}(b^j,b^{j+1}) \tag{12}$$

5 Feature Space

The feature space $\vec{\psi}_{seg}(x,y)$ is a function of both the raw input x and candidate segmentation y. It is an abstracted version of the feature space, where statistical patterns existing in x may be more easily discovered or represented using knowledge of the start, end, and behavior class of each bout. We define a variety of different types of basic feature expansion operations that are used to synthesize bout-level features $\vec{\psi}_{bout}(x,s,e,b)$ from frame-level features $\vec{\phi}(x^t_i)$. It is our hope that a wide variety of different types of behaviors can be represented and learned using combinations of these different feature expansion operations. All bout-level features are computable in constant time (independent of the duration of the bout) using precomputed data structures like the integral images.

5.1 Bout Statistic Features

These are features that compute some simple statistic over the bout duration. Each particular statistic occurs over a single frame feature $\phi_k(x^t)$ at a time, and adds one new bout-level feature to $\vec{\psi}_{seg}(x,y)$. Sum, average, and standard deviation features are computed using precomputed integral images over the raw feature and and square feature responses. Min and max features are updatable in constant time.

Sum: $\text{sum}_k(s,e) = \sum\limits_{s \le t < e} \phi_k(x^t)$	Sum Variance: $\text{var}_k(s,e) = \sum\limits_{s \le t < e} \left(\phi_k(x^t) - \mu_k(s,e)\right)^2$	Minimum: $\min_k(s,e) = \min\limits_{s \le t < e} \phi_k(x^t)$
Average: $\mu_k(s,e) = \frac{1}{e-s} \sum\limits_{s \le t < e} \phi_k(x^t)$	Standard Deviation: $\sigma_k(s,e) = \sqrt{\frac{1}{e-s}\text{var}_k(s,e)}$	Maximum: $\max_k(s,e) = \max\limits_{s \le t < e} \phi_k(x^t)$

5.2 Threshold and Histogram Features

We allow threshold features, which convert any of the above bout statistics to features of value 0 or 1:

$$\begin{aligned} \text{Max Threshold :} \quad & \text{gt}_k(s,e,Q,\text{op}) = [\text{g}_k^1, \text{g}_k^2, ..., \text{g}_k^Q], \qquad \text{g}_k^l = 1[q_k^l > \text{op}_k(s,e)] \\ \text{Min Threshold :} \quad & \text{lt}_k(s,e,Q,\text{op}) = [\text{l}_k^1, \text{l}_k^2, ..., \text{l}_k^Q], \qquad \text{l}_k^l = 1[q_k^l < \text{op}_k(s,e)] \end{aligned}$$

where op_k is one of the above bout statistics (e.g., sum_k, μ_k, σ_k, etc.) and each q_k^l is an arbitrary constant. The threshold constants are computed by evaluating median statistics on the training set. In other words, we construct a set of Q thresholds $q_k^1, q_k^2, ...q_k^Q$, such that an equal fraction of the training set lies in each interval q_k^l to q_k^{l+1}.

We also allow raw histogram features, which assign a frame to histogram bin l if $q_k^l \le \phi^k(x^t) < q_k^{l+1}$, and computes a count of the number of frames lying in each histogram bin:

$$\begin{aligned} \text{Histogram :} \quad & \text{hist}_k(s,e,Q) = [h_k^1, h_k^2, ..., h_k^Q], \qquad h_k^l = \sum_{s \le t < e} 1[q_k^l \le \phi_k(x^t) < q_k^{l+1}] \\ \text{Normalized Histogram :} \quad & \text{n\_hist}_k(s,e,Q) = \frac{1}{e-s}[h_k^1, h_k^2, ..., h_k^Q] \end{aligned}$$

5.3 Temporal Region Features

A temporal feature is a feature occurring over some temporal region that may be different than the entire bout (s,e). It is usually convenient to express the region in a coordinate system normalized with respect to the start and end of the bout, where the point s maps to 0 and e maps to 1.

Temporal partition features divide the bout into R evenly spaced temporal regions and add new features to $\vec{\psi}_{bout}(x,s,e,b)$ as the concatenation of each region. Temporal pyramids concatenate multiple partitions of size $R = 2^0, 2^1, ...2^{L-1}$.

$$\begin{aligned} \text{Temporal Region :} \quad & \text{op}'_k(s,e,s',e') = \text{op}_k\Big(s + s'(e-s),\ s + e'(e-s)\Big) \\ \text{Temporal Partition :} \quad & \text{part}_k(s,e,R,\text{op}) = [r_k^1, r_k^2, ...r_k^R], \qquad r_k^l = \text{op}'_k\left(s, e, \frac{l-1}{R}, \frac{l}{R}\right) \\ \text{Temporal Pyramid :} \quad & \text{pyr}_k(s,e,L,\text{op}) = [p_k^1, p_k^2, ...p_k^L], \qquad p_k^l = \text{part}_k(s,e,2^{l-1}) \end{aligned}$$

Temporal region, partition, and pyramid features can be constructed ontop of bout statistic, threshold, or histogram features.

5.4 Temporal Difference Features

We use Haar-like features, which are the difference between two temporal region features. In particular, we use features designed to accentuate the difference between frames in the beginning of the bout and the frames preceeding the bout, and the difference between frames in the end of the bout and frames following

Algorithm 2 Behavior Segmentation

```
INPUT: $x$, $\vec{w}$
1: Initialize: S[0][1...$B$] $\leftarrow$ 0, M[0][1...$B$] $\leftarrow \emptyset$
2: for $e = 1$ to $T$ do
3:     for $b \in B$ do
4:         Compute the optimal score $S[e][b]$ and solution $M[e][b]$ for beginning a bout of class $b$ at time $e$:
               $\mathrm{S}[e][b], \mathrm{M}[e][b] \leftarrow \min_{0 \le s < e, b' \in B} \left( \mathrm{S}[s][b'] + \vec{w}_b \cdot \vec{\psi}_{bout}(x, s, e, b) - \lambda_{b'}(b) \right)$
5:     end for
6: end for
7: Extract optimal solution: $(s^m, b^m) \leftarrow \max_b M[T][b]$,    $(s^{i-1}, b^{i-1}) \leftarrow M[s^i][b^i]$,    $i = m, m-1, ...2$
OUTPUT: $y = \{(s^1, e^1, b^1), (s^2, e^2, b^2), ..., (s^m, e^m, b^m)\}$
```

the bout:

$$\begin{aligned}
\text{Start Difference :} \quad & \mathrm{start}_k(s,e,d,\mathrm{op}) = \mathrm{op'}_k(s,e,0,d) - \mathrm{op'}_k(s,e,-d,0) \\
\text{Absolute Start Difference :} \quad & \mathrm{a\_start}_k(s,e,d,\mathrm{op}) = |\mathrm{start}_k(s,e,d,\mathrm{op})| \\
\text{End Difference :} \quad & \mathrm{end}_k(s,e,d,\mathrm{op}) = \mathrm{op'}_k(s,e,1-d,1) - \mathrm{op'}_k(s,e,1,1+d) \\
\text{Absolute End Difference :} \quad & \mathrm{a\_end}_k(s,e,d,\mathrm{op}) = |\mathrm{end}_k(s,e,d,\mathrm{op})|
\end{aligned}$$

5.5 Harmonic Features

Harmonic features accentuate periodic, repeated motions such as waving or walking:

$$\begin{aligned}
\text{Harmonic :} \quad & \mathrm{harmonic}_k(s,e,h,\mathrm{op}) = \sum_{l=1}^{H} -1^l \times \mathrm{op'}_k\left(s, e, \frac{l-1}{h}, \frac{l}{h}\right) \\
\text{Absolute Harmonic :} \quad & \mathrm{a\_harmonic}_k(s,e,h,\mathrm{op}) = |\mathrm{harmonic}_k(s,e,h,\mathrm{op})|
\end{aligned}$$

5.6 Global Difference Features

Global difference features compare the feature response inside the bout to the global average response (the average response throughout the entire tracked sequence):

$$\begin{aligned}
\text{Global Sum Difference :} \quad & \mathrm{g\_sum}_k(s,e,\mathrm{op}) = \mathrm{op}_k(s,e) - \frac{\mathrm{op}_k(0,T)}{T}(e-s) \\
\text{Global Absolute Sum Difference :} \quad & \mathrm{g\_a\_sum}_k(s,e,\mathrm{op}) = |\mathrm{g\_sum}_k(s,e,\mathrm{op})| \\
\text{Global Average Difference :} \quad & \mathrm{g\_ave}_k(s,e,\mathrm{op}) = \frac{\mathrm{op}_k(s,e)}{e-s} - \frac{\mathrm{op}_k(0,T)}{T} \\
\text{Global Absolute Average Difference :} \quad & \mathrm{g\_a\_ave}_k(s,e,\mathrm{op}) = |\mathrm{g\_ave}_k(s,e,\mathrm{op})|
\end{aligned}$$

6 Inference

We can find the optimal solution to $g(x) = \mathrm{argmax}_y f(x,y)$ where $f(x,y)$ is defined in Equation 8 using dynamic programming in time $\mathrm{O}(T^2(B^2 + D))$, where T is the number of frames in x, B is the number of behavior classes, and D is the dimensionality of the bout-level feature space $\vec{\psi}_{seg}(x,y)$. The algorithm is shown in Algorithm 2.

The algorithm to find the most violated constraint (Eq. 4) is identical, except that Line 4 of Algorithm

2 incorporates the appropriate component of the loss term

$$\mathrm{S}[e][b], \mathrm{M}[e][b] \leftarrow \min_{s<e, b' \in B} \Big(\mathrm{S}[s][b'] + \vec{w}_b \cdot \vec{\psi}_{bout}(x, s, e, b) - \lambda_{b'}(b) + \mathcal{L}_{comp}(y, (s, e, b')) \Big) \tag{13}$$

$$\mathcal{L}_{comp}(y, (\bar{s}, \bar{e}, \bar{b})) = \sum_{(s,e,b)\in y} \left(\frac{\ell^b_{\bar{b}}}{e-s} + \frac{\ell^{\bar{b}}_{b}}{\bar{e}-\bar{s}} \right) \Big((s, e) \cap (\bar{s}, \bar{e}) \Big) \tag{14}$$

We can improve the runtime of Algorithm 2 to $\mathrm{O}\big(T \frac{\log T}{\log(1+\gamma)}(B^2 + D)\big)$ by reducing the search space in line 4 to time durations in a geometrically increasing series $(1+\gamma)^i$, and γ is an arbitarily small approximation factor.

7 Active Labeling

We allow an active labeling interface, where the user can speed up the annotation process using an initial segmenter trained on only a small training set. This is done by running the segmentation algorithm, correcting the mistakes by providing a partial labelling y' (an assignment to only a fraction of the frames), re-running the segmentation algorithm such that any returned segmentation must agree with y'

$$g(x) = \arg \max_{y, y' \subseteq y} f(x, y) \tag{15}$$

and so on. This can be implemented with a slight change to Line 4 of Algorithm 2

$$\mathrm{S}[e][b], \mathrm{M}[e][b] \leftarrow \min_{s<e, b' \in B} \Big(\mathrm{S}[s][b'] + \vec{w}_b \cdot \vec{\psi}_{bout}(x, s, e, b) - \lambda_{b'}(b) + \delta(y', s, e, b) \Big) \tag{16}$$

$$\delta(y', s, e, b) = \begin{cases} -\infty & \text{if } \bigcap\limits_{y', b' \neq b} (s, e) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

Additionally, as new training examples are added, we can reduce the re-training time significantly by caching the active sets A_i from prior runs of the algorithm.