# Discriminative Subsequence Mining for Action Classification

Sebastian Nowozin[1] Gökhan Bakır[2]
Koji Tsuda[1]

[1] : Max Planck Institute for Biological Cybernetics, Spemannstrasse 38, 72076 Tübingen, Germany
{sebastian.nowozin,koji.tsuda}@tuebingen.mpg.de

[2] : Google GmbH, Freigutstrasse 12, 8002 Zurich, Switzerland
ghb@google.com

## Abstract

*Recent approaches to action classification in videos have used sparse spatio-temporal words encoding local appearance around interesting movements. Most of these approaches use a histogram representation, discarding the temporal order among features. But this ordering information can contain important information about the action itself,* e.g. *consider the sport disciplines of* hurdle race *and* long jump*, where the global temporal order of motions (running, jumping) is important to discriminate between the two. In this work we propose to use a sequential representation which retains this temporal order. Further, we introduce Discriminative Subsequence Mining to find optimal discriminative subsequence patterns. In combination with the LPBoost classifier, this amounts to simultaneously learning a classification function and performing feature selection in the space of all possible feature sequences. The resulting classifier linearly combines a small number of interpretable decision functions, each checking for the presence of a single discriminative pattern. The classifier is benchmarked on the KTH action classification data set and outperforms the best known results in the literature.*

## 1. Introduction

Human activity recognition and classification systems can provide useful semantic information to solve higher-level tasks, for example to summarize or index videos based on their semantic content. Robust activity classification is also important for video-based surveillance systems, which should act intelligently, such as alerting an operator of a possibly dangerous situation.

Building a general activity recognition and classification system is a challenging task, because of variations in the environment, objects and actions. Variations in the *environment* can be caused by cluttered or moving background, camera motion, occlusion, weather- and illumination changes. Variations in the *objects* are due to differences in appearance, size or posture of the objects or due to self-motion which is not itself part of the activity. Variations in the *action* can make it difficult to recognize semantically equivalent actions as such, for example imagine the many ways to jump over an obstacle or different ways to throw a stick.

In current computer vision research, it is common to represent each data instance (i.e., video or image) as a histogram of visual words, see for example the recent VOC2006 object classification challenge [6]. However, due to the variations stated above, not all visual words are informative for classification. Thus, *feature selection* is important both for robustness against variations and interpretability of the classification rule. However, simply removing visual words based on some statistics (e.g., correlation to the class label) might be harmful, because, if combined with other features, a visual word can possibly become an important feature. In this light, finding the optimally discriminative *combination of features* is a combinatorial optimization problem, leading to an exponentially large feature space. The problem of high dimensionality of such feature space can partially be overcome using kernel methods, which allows one to learn a classification function implicitly. However, the cost is that the resulting classification function is not interpretable.

Recently, Nowozin et al. [13] proposed an image classification method termed *Itemset Boosting* that selects discriminative features from power sets of visual words. Thus, features are still selected from a meaningful exponentially large feature space, but the resulting classification function is sparse and interpretable. They find that a combination of

visual words captures objects in the image well (*e.g.*, motorbikes), and the classification accuracy was comparable to state-of-the-art SVM methods.

When we use the spatio-temporal words for action recognition, we can directly adopt the Itemset Boosting approach for this task. However, this discards the temporal order of spatio-temporal words. Instead, we propose to represent a video as a *sequence of sets* of discretized spatio-temporal words. Each set in the sequence is produced by collecting the features of a number of adjacent video frames, such that the overall sequence preserves approximately the global temporal order of features. To train a classifier using this representation we extend the PrefixSpan subsequence mining algorithm [14] in combination with LPBoost [3].

In the remainder of this section we briefly introduce related work. In Section 2 we first describe spatio-temporal features for action classification in videos and how they naturally form a sequential representation for videos. In Section 3 we propose the LPBoost classifier for sequential representations. This produces as subproblem a combinatorial optimization problem over the space of all sequences. To efficiently solve this problem, we propose in Section 4 the discriminative subsequence mining (DPrefixSpan) algorithm. Section 5 covers experiments and results based on the KTH action classification benchmark dataset. Finally, Sections 6 and 7 discuss these results and draw conclusions.

### 1.1. Related work

We now discuss two main groups of approaches popular in the literature, part-based representations and holistic representations.

**Part-based representations.** Part-based representations based on interest point detectors, combined with robust descriptor methods have been used very successfully for object classification tasks, see for example the approaches submitted to the VOC2006 challenge [6].

Recently, representations based on sparse local features have become popular also for human action classification. Laptev [9] proposed to assign each voxel in a spatio-temporal volume a saliency value and extract descriptors from the neighborhood of local saliency maxima. Schüldt et al. [17] used these features successfully for human action classification by discretizing them into codewords and producing histogram of the occurring words for each video. The histograms are treated as fixed-length vectors to train a classification function. Dollár et al. [4] argue in principle for the same approach but suggest to use a denser sampling of the spatio-temporal volume by only requiring each interest point to be a local maxima in the spatial directions instead of *both* spatial and temporal dimensions. They justify this change by increased classification performance on the same dataset. Niebles et al. [11] train an unsupervised probabilistic topic model on the same features as Dollár and obtain comparable classification performance. Another approach is due to Ke at al. [7], who use a forward feature selection procedure to train a classifier on volumetric features.

**Holistic representations.** In earlier works a more *holistic* representation has been advocated. Bobick et al. [2] proposed *motion history images* (MHI) as a meaningful way to encode short spans of motion. For each frame of the input video the MHI is the gray scale image which records the location of motion, where recent motion has high intensity values and older motion produces lower intensities. The MHI representation can be matched efficiently using global statistics, such as moment features. Weinland et al. [19] extended the idea to *motion history volumes* by means of multiple cameras. By using such controlled environment a high classification accuracy and desirable invariances can be achieved. However, for most practical cases, Weinland's environment with five cameras around the scene is too expensive or difficult to setup. Efros et al. [5] create stabilized spatio-temporal volumes for each object whose action is to be classified. For each volume a smoothed dense optical flow field is extracted and used as descriptor. Their method is particularly best for classifying the actions of distant objects where detailed information is unavailable. Yilmaz and Shah [21] again use a spatio-temporal volume, but only project the contour of each frame into the volume. Descriptors encoding direction, speed and local shape of the resulting surface are generated by measuring local differential geometrical properties. In [22], Zelnik-Manor and Irani describe features derived from space-time gradients at multiple temporal scales. To compare two sequences of these features, they use a sliding-window of fixed size and the distance between two such windows is calculated as $\chi^2$-distance or Mahalanobis distance. Their method works well to cluster a single long video sequence into similar actions, as well as recognizing actions in real-time.

There is a large body of work which first recover the posture of the human actor by means of tracking and fitting a detailed model of the human body. Action classification can then be performed by using the intrinsic model parameters as features, providing great robustness and invariance. Representatively, let us cite the work of Yacoob and Black [20], Ramanan and Forsyth [16], Agarwal and Triggs [1] and for an unsupervised method, Song et al. [18].

**Comparison.** *Part-based representations* treat the video as a set of independent features, where each feature is equally important, and by discarding the position information they are robust against changes in both space and time dimensions. A practical drawback of part-based representations is the variable size of the resulting representations,

which is often overcome by producing a histogram of fixed-length. Naturally, part-based representations do not require tracking and are often more resistant to clutter, as only few parts may be occluded.

*Holistic representations* derive a fixed-length description vector for each object whose action is to be classified. Approaches using these representations often require more preprocessing of the input data, such as object tracking, registration, shape fitting or optical flow field calculation. In case the environment conditions can be controlled these approaches perform very well.

Each of the above methods has its particular strength but is also limited in its application. In particular, the part-based methods discussed discard the temporal order of the parts, which contains useful information to disambiguate actions. For example, consider the disciplines of *hurdle race* and *long jump*, where the global temporal order of motions (running, jumping) is important to discriminate between the two. Therefore, in this work we use a part-based view but *preserve* information about the relative temporal order of spatio-temporal words by proposing a classifier for a sequential representation.

## 2. Sequence Representation of Videos

As a basis of our sequence representation, we use the spatio-temporal detector of Dollár which has shown good experimental performance in Niebles et al. [11] and Dollár et al. [4] for human action classification. In the Dollár detector, a response function $R = (I * g * h_{ev})^2 + (I * g * h_{od})^2$ is calculated at each spatio-temporal voxel $(x, y, t)$ in the video volume $I$. In the spatial directions, a 2D Gaussian kernel $g$ with bandwidth $\sigma$ is used, while temporally, a quadrature pair of 1D Gabor filters $h_{ev}(t; \tau, \omega) = -\cos(2\pi t\omega)e^{-t^2/\tau^2}$ and $h_{od}(t; \tau, \omega) = -\sin(2\pi t\omega)e^{-t^2/\tau^2}$ is used. The Gabor filters respond strongest on temporal intensity changes that vary at the frequency $\omega$, which has to be set in advance.

For each interesting point found, we have the spatio-temporal coordinates $(x, y, t)$ as well as the *descriptor*, the concatenated vector of voxel values in the neighborhood of the point. Typically, we have volumes of size $13 \cdot 13 \cdot 19$ voxels, so the descriptor is a 3211-dimensional vector. To reduce the dimensionality, PCA is used to keep the first 25 components of each descriptor. The reduced descriptors in $\mathbb{R}^{25}$ are clustered using $k$-means clustering to produce a codebook of prototypes. Using the code book, a video is represented as a set of words of the form $(x, y, t, w)$, where $(x, y)$ are the coordinates in the video frame $t$ and $w$ is the codebook index.

Finally, the words are sorted ascendingly by their time components $t$ and then grouped into *temporal bins* as shown in Figure 1, where the first frame a feature occurred is denoted *start*, the last frame is denoted *end*. Two parameters determine how the features are mapped into the temporal bins, i) the number of temporal bins $B$, and ii) the temporal overlap $\tau$, with $0 \leq \tau < 1$. The length of each temporal bin is simply the overall number of frames (end-start), divided by $B/(1 + \tau)$, such that a large value of $\tau$ denotes a larger overlap. The bins are distributed equidistant over the range of found features. Since the bins are overlapping, it is possible that a word is assigned to more than two bins. Although for the experiments we will keep $B$ fixed over all videos, our algorithm can deal with a variable number of bins.

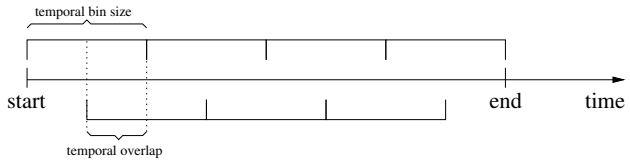


Figure 1. Temporal binning scheme: A number of overlapping temporal bins are distributed equidistantly over the video frames. Here $B = 7$, $\tau = 0.5$.

Now each video is encoded as a sequence of sets of integers. To be more precise, let us formalize the notion of *sequence* and define a *subsequence* relationship.

**Definition 1** (Sequence)**.** *A sequence $\boldsymbol{s} = (s_1, s_2, \ldots, s_\ell)$ is defined as an ordered list of* elements $s_i$*. Each element $s_i$ is a finite set of integers. Let $\mathcal{S}$ denote the space of all possible sequences.*

**Definition 2** (Subsequence)**.** *For any two sequences $\boldsymbol{s}_1, \boldsymbol{s}_2$ with $\ell_1, \ell_2$ elements, respectively, the relationship $\subseteq: \mathcal{S} \times \mathcal{S}$ is defined as follows.*

$$\boldsymbol{s}^1 \subseteq \boldsymbol{s}^2 \quad \Leftrightarrow \quad \exists\,(i_1, \ldots, i_{\ell^1}) \text{ with } i_p > i_q \text{ if } p > q, \text{ such that } \forall\, k = 1, \ldots, \ell^1 : \boldsymbol{s}^1_k \subseteq \boldsymbol{s}^2_{i_k},$$

*where the last $\subseteq$ relation is the normal subset relationship evaluated between two elements of the sequences.*

*Thus, a sequence $\boldsymbol{s}^1$ is a subsequence of $\boldsymbol{s}^2$ if there exists a strictly increasing element mapping such that each element of $\boldsymbol{s}^1$ is a subset of the respective mapped element of $\boldsymbol{s}^2$.*

Note that the above subsequence relationship allows arbitrary long gaps when matching the sequences.

## 3. Classifier

Action recognition is a multiclass classification problem in general, but first we focus on the binary classification problem. Let us denote the training data as $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^{\ell}$, where $\boldsymbol{x}_n$ is the sequence corresponding to a video and $y_n \in \{-1, 1\}$ is a class label. Denote by $\bar{\mathcal{S}} \subset \mathcal{S}$ the set of all *patterns*, i.e., all subsequences of $\{\boldsymbol{x}_n\}_{n=1}^{\ell}$. We use the LPBoost algorithm [3] to construct the classification function as a linear combination of weak hypothesis functions.

LPBoost is preferred over AdaBoost, because of its fast convergence and sparse coefficients [3]. Our hypothesis functions are defined as

$$h(\boldsymbol{x};\boldsymbol{s},\omega)=\begin{cases}\omega & \boldsymbol{s}\subseteq\boldsymbol{x}\\ -\omega & \text{otherwise.}\end{cases}, \tag{1}$$

where an extra variable $\omega\in\Omega,\Omega=\{-1,1\}$ is introduced so that the stumps are two-sided and can decide for either class decision. Our classification function has the form

$$f(\boldsymbol{x})=\sum_{(\boldsymbol{s},\omega)\in\bar{\mathcal{S}}\times\Omega}\alpha_{\boldsymbol{s},\omega}h(\boldsymbol{x};\boldsymbol{s},\omega). \tag{2}$$

where $\alpha_{\boldsymbol{s},\omega}$ is a weight for pattern $\boldsymbol{s}$ and parameters $\omega$ such that $\sum_{(\boldsymbol{s},\omega)\in\bar{\mathcal{S}}\times\Omega}\alpha_{\boldsymbol{s},\omega}=1$ and $\alpha_{\boldsymbol{s},\omega}\geq 0$. This is a linear discriminant function in an intractably large dimensional space. To obtain an evaluable classification function, we need to obtain a *sparse* weight vector $\boldsymbol{\alpha}$, where only few coefficients are nonzero. The training problem is formulated as the linear program,

$$\min_{\boldsymbol{\alpha},\boldsymbol{\xi},\rho}\quad -\rho+D\sum_{n=1}^{\ell}\xi_n \tag{3}$$

$$\text{sb.t.}\quad \sum_{(\boldsymbol{s},\omega)\in\bar{\mathcal{S}}\times\Omega} y_n\alpha_{\boldsymbol{s},\omega}h(\boldsymbol{x}_n;\boldsymbol{s},\omega)+\xi_n\geq\rho,\quad n=1,\ldots,\ell \tag{4}$$

$$\sum_{(\boldsymbol{s},\omega)\in\bar{\mathcal{S}}\times\Omega}\alpha_{\boldsymbol{s},\omega}=1,\quad \boldsymbol{\alpha}\geq 0,\quad \boldsymbol{\xi}\geq 0,$$

where $\rho$ is the soft-margin, separating negative from positive examples, $D=\frac{1}{\nu\ell}$, and $\nu\in(0,1)$ is a parameter controlling the cost of misclassification which has to be found using model selection techniques, such as cross-validation. Directly solving this optimization problem is intractable due to the large number of variables in $\boldsymbol{\alpha}$. Instead we solve the following *equivalent* dual problem instead.

$$\min_{\boldsymbol{\lambda},\gamma}\quad \gamma \tag{5}$$

$$\text{sb.t.}\quad \sum_{n=1}^{\ell}\lambda_n y_n h(\boldsymbol{x}_n;\boldsymbol{s},\omega)\leq\gamma,\ (\boldsymbol{s},\omega)\in\bar{\mathcal{S}}\times\Omega \tag{6}$$

$$\sum_{n=1}^{\ell}\lambda_n=1,\quad 0\leq\lambda_n\leq D,\ n=1,\ldots,\ell.$$

Afterwards the primal solution $\boldsymbol{\alpha}$ is obtained from the Lagrange multipliers [3].

The dual problem has a limited number of variables, but a huge number of constraints. Such a linear program can be solved efficiently by the *constraint generation* technique [3]: Starting with an empty hypothesis set, the hypothesis whose constraint (6) is violated the most is identified and added iteratively. Each time a hypothesis is added, the optimal solution is updated by solving the restricted dual problem. In each iteration, we have to solve the following problem to find an optimal hypothesis,

$$(\hat{\boldsymbol{s}},\hat{\omega})=\text{argmax}_{(\boldsymbol{s},\omega)\in\bar{\mathcal{S}}\times\Omega}g(\boldsymbol{s},\omega), \tag{7}$$

where the gain function is defined as

$$g(\boldsymbol{s},\omega)=\sum_{n=1}^{\ell}\lambda_n y_n h(\boldsymbol{x}_n;\boldsymbol{s},\omega). \tag{8}$$

The column generation algorithm terminates if there is no hypothesis violating the constraint (6).

The proposed classifier allows us to learn two-class decision functions. To solve a multiclass learning problem using the above method we use a 1-vs-1 class decomposition in the form of a *decision directed acyclic graph (DDAG)* [15], producing for $k$ classes $\frac{k(k-1)}{2}$ 1-vs-1 problems. While this is similar to the usual 1-vs-1 decomposition, the DDAG offers the additional advantage that we do not have to resolve ties during test time.

## 4. Optimal Pattern Search

In this section we describe an algorithm to solve the maximum-gain search problem (7). The problem of finding the maximum-gain sequence is difficult due to the size of the combinatorial space to be considered. By choosing a clever search strategy we can solve this problem optimally. Our proposed algorithm is a generalization of the PrefixSpan algorithm by Pei et al. [14], which is an algorithm to enumerate all *frequent* subsequences: Find all $\boldsymbol{s}\in\mathcal{S}$ such that

$$\sum_{n=1}^{\ell}I(\boldsymbol{s}\subseteq\boldsymbol{x}_n)\geq\tau,$$

where $\tau$ is a threshold called the minimum support parameter. It generates a search tree starting from an empty root node, as shown in Figure 2. In the search tree, each child node contains a sequence which is an extension of its parent node's sequence. By defining an ordering on the sequences, PrefixSpan never generates duplicate sequences in the search tree.

Naively, our problem can now be solved by constructing the search tree containing all patterns $\bar{\mathcal{S}}$ by PrefixSpan($\gamma=1$), and calculate the gain of all patterns. However, for efficient search, it is important to minimize the size of the explored search tree. To this aim, *tree pruning* is crucially important: Suppose the search tree is generated up to the pattern $\boldsymbol{s}$ and denote by $g^*$ the maximum gain among the ones observed so far. If it is guaranteed that $g(\boldsymbol{s}',\omega)$ is not larger than $g^*$ for any extension $\boldsymbol{s}'$ of $\boldsymbol{s}$ and any $\omega$, we can avoid the generation of downstream nodes without losing the optimal pattern. Here, we use the following pruning

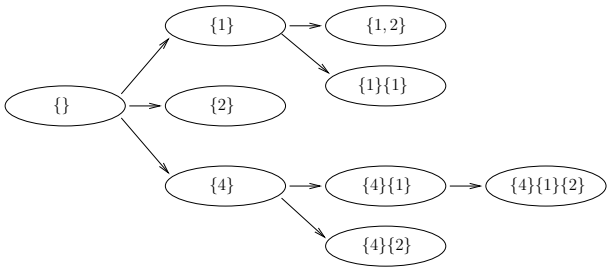


Figure 2. Illustration of the search tree used for subsequence mining.

condition first proposed in [10] for pattern search problems. See [8] for the proof.

**Theorem 1.** *Let us define the gain bound function as follows,*

$$\mu(\boldsymbol{s}) = \max\{ \begin{array}{l} 2\sum_{\{n|y_n=+1,\boldsymbol{s}\subseteq\boldsymbol{x}_n\}}\lambda_n - \sum_{n=1}^{\ell} y_n\lambda_n, \\ 2\sum_{\{n|y_n=-1,\boldsymbol{s}\subseteq\boldsymbol{x}_n\}}\lambda_n + \sum_{n=1}^{\ell} y_n\lambda_n \ \}. \end{array} \quad (9)$$

*If the following condition is satisfied,*

$$g^* > \mu(\boldsymbol{s}),$$

*the gain $g(\boldsymbol{s}', \omega)$ of any downstream sequence $\boldsymbol{s}' \supset \boldsymbol{s}$ does not exceed the current best $g^*$ for any $\omega \in \Omega$.*

Our Discriminative PrefixSpan mining algorithm (DPrefixSpan) is shown in Algorithm 1. Essentially, the difference from PrefixSpan is that, i) it finds the optimal pattern that maximizes the gain function instead of enumeration, and ii) the gain bound $\mu$ is used for tree pruning. The algorithm recursively generates a subsequence search tree (line 21). It keeps a variable $g^*$ which contains the highest gain value observed so far and is updated whenever a better subsequence is observed (lines 11 and 24). The tree pruning happens in line 14, where the recursion call is skipped if the pruning condition holds. Due to space restriction, we omit the description of the frequent item finding (Algorithm 1, line 3) and sequence projection (Algorithm 1, line 20) steps. These two operations are identical to the original PrefixSpan algorithm specifications in Pei et al. [14].

## 4.1. Iterative Deepening

The current maximum gain $g^*$ starts from $-\infty$ and monotonically increases as the tree grows. For effective pruning, it is desired that $g^*$ increases as rapidly as possible. DPrefixSpan generates the tree in depth-first order, which is usually not optimal in this respect. To discover high-gain patterns earlier, we use a variant of the *iterative deepening $A^*$* search algorithm [12], as shown in Algorithm 2. First, the algorithm calls DPREFIXSPAN with restricting the search depth to 1. Then, DPREFIXSPAN returns the optimal pattern of length 1. The depth is gradually increased until the overall optimal pattern is found (line 15). After each iteration, the best gain observed so far is used for pruning in the following iterations (line 6). If there are short high-

**Algorithm 1** Discriminative PrefixSpan mining algorithm (DPrefixSpan)

**Input:**
Prefix sequence: $\alpha \in \mathcal{S}$.
Set of sequences: $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\ell\}, \boldsymbol{x}_n \in \mathcal{S}$.
Labels: $Y = \{y_1, \ldots, y_\ell\}$, $y_i \in \{-1, 1\}$.
Weights: $\boldsymbol{\lambda} = \{\lambda_1, \ldots, \lambda_\ell\}$, $0 \leq \lambda_i \leq 1$.
Minimum required support: $\tau \geq 1$.
Minimum required gain: $\theta > 0$.
Maximum mining depth: level $> 0$.

**Output:**
Best observed gain: $g^*$.
Subsequence $\hat{\boldsymbol{s}} \in \mathcal{S}$ and additional parameter $\hat{\omega} \in \Omega$, maximizing $g(\boldsymbol{s}, \omega)$.
Extensions-possible flag: found_extensions.

**Algorithm:**

1: **function** DPREFIXSPAN($\alpha, X, Y, \boldsymbol{\lambda}, \tau, \theta$, level)
2: $g^* \leftarrow \theta$, $(\hat{\boldsymbol{s}}, \hat{\omega}) \leftarrow (<>, <>)$
3: Scan $X$ and find all frequent items $(b, \text{last\_insert}) \in I \times \{\bot, \top\}$, subject to
   1. $(b, \top)$: $b$ can be appended to the last element of $\alpha$, or
   2. $(b, \bot)$: $b$ can be appended as a new element to $\alpha$
4: found_extensions $\leftarrow \bot$
5: **for** each $(b, \text{last\_insert})$ found **do**
6: Produce $\alpha'$ from $\alpha$ and $(b, \text{last\_insert})$
7: $\omega \leftarrow \text{argmax}_{\omega\in\Omega} g(\alpha', \omega)$
8: **if** level $= 1$ **then**
9: **if** $g(\alpha', \omega) \geq g^*$ **then**
10: $(\hat{\boldsymbol{s}}, \hat{\omega}) \leftarrow (\alpha', \omega)$
11: best_gain $\leftarrow g(\hat{\boldsymbol{s}}, \hat{\omega})$
12: **end if**
13: **end if**
14: **if** $\mu(\alpha') \geq g^*$ **then**
15: ▷ Possible improvement by extending $\alpha'$
16: **if** level $\leq 1$ **then**
17: found_extensions $\leftarrow \top$
18: **continue** ▷ Maximum depth reached
19: **end if**
20: Produce $X|_{\alpha'}$ by $\alpha'$-projecting $X$
21: (sub_gain, $\boldsymbol{s}', \omega'$) $\leftarrow$ DPREFIXSPAN($\alpha'$, $X|_{\alpha'}, Y, \boldsymbol{\lambda}, \tau, g^*$, level $- 1$)
23: **if** $\boldsymbol{s}' \neq <>$ **then**
24: $(g^*, \hat{\boldsymbol{s}}, \hat{\omega}) \leftarrow$ (sub_gain, $\boldsymbol{s}', \omega'$)
25: **end if**
26: **end if**
27: **end for**
28: **end function**

gain subsequences, the algorithm finds them earlier and the pruning becomes more effective. In practice, this improved pruning is crucial to make the search problem tractable.

**Algorithm 2** Iterative deepening DPrefixSpan algorithm

**Input:**

Set of sequences: $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\ell\}, \boldsymbol{x}_n \in \mathcal{S}$.
Labels: $Y = \{y_1, \ldots, y_\ell\}, y_i \in \{-1, 1\}$.
Weights: $\boldsymbol{\lambda} = \{\lambda_1, \ldots, \lambda_\ell\}, 0 \leq \lambda_i \leq 1$.
Minimum support: $\tau \geq 1$.

**Output:**

Subsequence $\hat{\boldsymbol{s}} \in \mathcal{S}$ and additional parameters $\hat{\omega} \in \Omega$, maximizing $g(\boldsymbol{s}, \omega)$.

**Algorithm:**

**function** ID_DPREFIXSPAN($X, Y, \boldsymbol{\lambda}, \tau$)
  $g^* \leftarrow -\infty$
  level $\leftarrow 1$ ▷ Iterative deepening depth
  **loop**
    (current_gain, $\boldsymbol{s}_c, \omega_c$, found_extensions)
      $\leftarrow$ DPREFIXSPAN($<>, X, Y, \boldsymbol{\lambda}$, $\tau, g^*$, level)
    **if** current_gain $> g^*$ **then**
      $(g^*, \hat{\boldsymbol{s}}, \hat{\omega}) \leftarrow$ (current_gain, $\boldsymbol{s}_c, \omega_c$)
    **end if**
    **if** found_extensions = false **then**
      **break**
        ▷ No possibly optimal extensions beyond current level
    **end if**
    level $\leftarrow$ level $+ 1$ ▷ Increase mining depth
  **end loop**
**end function**

## 5. Experiments and Results

To evaluate our approach, we use the KTH human action classification data set of Schüldt et al. [17], available online[1]. It consists of 25 individuals, each performing six activities (boxing, hand-clapping, hand-waving, jogging, running and walking) under four different environment conditions. Together, with one broken video file removed, the data set totals 599 video clips. We used the training, validation and testing splits as proposed in [17], such that the sets contain 191, 192 and 216 samples, respectively.

The spatio-temporal features were extracted as described in section 2 using the toolbox[2] provided by Piotr Dollár with the default settings. Model selection is performed on the training and validation sets followed by a single training run on the combined training+validation set with the best parameters of the validation phase. The final reported

[1] http://www.nada.kth.se/cvap/actions/

[2] http://vision.ucsd.edu/~pdollar/research/research.html

Figure 3. KTH Action Classification dataset with six actions and a total of 599 video sequences. The actions are shown in alphabetical order: boxing, handclapping, handwaving, jogging, running, walking.

classification accuracy is the one evaluated on the test set. Codebooks of sizes 128, 192, 256, 384, 512, 768 and 1024 codewords are created from the training set descriptors. In all experiments, the same features and codebooks are used to produce sequences as well as the histograms.

**Subsequences.** For model selection, the number of bins is varied from $B = 1$ to $B = 15$; the temporal overlap $\tau = 0.5$ remains fixed. The LPBoost regularization parameter $\nu$ is set to 0.01, 0.05, 0.1 and 0.25. All combinations of codebook sizes, $B$ and $\nu$ have been tested.

**SVM baseline.** For the model selection of the baseline classifiers, the histograms are preprocessed in one of the following two ways, i) the 1-norm of the histogram is normalized, or ii) the histogram is "*binarized*", that is all non-zero entries of the histogram are set to one.

As SVM kernel we use the linear kernel, the RBF Gaussian kernel and the $\chi^2$-histogram-kernel[3]. For the RBF Gaussian and $\chi^2$-kernel the kernel width has been selected as the mean Euclidean and mean $\chi^2$ distance between all training samples, respectively. This is a common heuristic choice known to work well in practice. As multiclass decomposition both 1-vs-rest and 1-vs-1 decompositions have been tested.

In total, for the SVM baseline all combinations of the codebook sizes, histogram preprocessing methods, multiclass decompositions and kernel choices are part of the model selection procedure. Thus the model selection for the SVM baseline is much more exhaustive than in previous works [4, 17].

### 5.1. Results

The classification results of our Subsequence Boosting approach, the results of the baseline SVM classifiers and the

[3] $K(h, h') = \exp\left(-\frac{1}{A}\left[\frac{1}{2}\sum_{\{n:h_n+h'_n>0\}}\frac{(h_n-h'_n)^2}{h_n+h'_n}\right]\right)$.

results from the literature are shown in Table 1. During the model selection process a codebook with 768 codewords turned out to be consistently the best for all tested classifier types. Each of our 1-vs-1 class Subsequence Boosting classifiers selected around 20-70 active patterns, where the tendency is fewer and shorter patterns for classes that are easy to distinguish (*e.g.* boxing versus running), and more and longer patterns for difficult to separate classes.

Figure 4 visualizes the sequence of a single decision stump of a trained classifier. In Figure 5 we further illustrate how the subsequences typically match into unseen test sequences. The confusion matrix for our Subsequence Boosting classifier is shown in Figure 6.

Our features and preprocessing seem to be of high quality, given that the baseline SVM method produces better results than reported in the literature. In part, this is also due to more thorough model selection, as noted above.

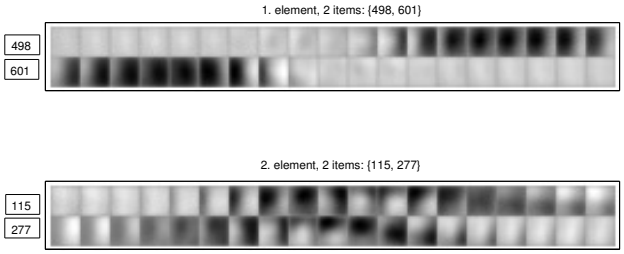


Figure 4. A discriminative pattern. Here, the pattern sequence is $\{498, 601\}\{115, 277\}$ and was selected in the jogging-vs-walking classifier. Each row in the figure shows a codebook vector as 19 frames of size 13x13 over time. The pattern was assigned a negative $\omega$-weight, such that the presence of this pattern will influence the decision towards the walking class.

| Method | KTH accuracy |
|---|---|
| Niebles et al. [11], LOO, pLSA | 81.50 |
| Dollár et al. [4], LOO, SVM RBF | 80.66 |
| Schuldt et al. [17], splits, SVM match | 71.71 |
| Ke et al. [7], splits, forward feat.-sel. | 62.94 |
| baseline SVM linear bin, 1-vs-1 | 83.33 |
| baseline SVM RBF bin, 1-vs-1 | 85.19 |
| baseline SVM $\chi^2$ bin, 1-vs-1 | 87.04 |
| Subsequence Boosting, $B = 12$, splits | 84.72 |

Table 1. Results for the KTH human action classification data set. For all the baseline SVM and Boosting results the model selection has been performed on the validation set, followed by a single training run on the joined training+validation set. The multiclass accuracy shown is the one measured on the final test set. For the baseline SVM results, the best classifier on the validation set was found with a codebook size of 768 and a regularization parameter of $C = 10$ for all kernels. The subsequence boosting result is obtained with a codebook size of 768, $B = 12$ and $\nu = 0.05$.

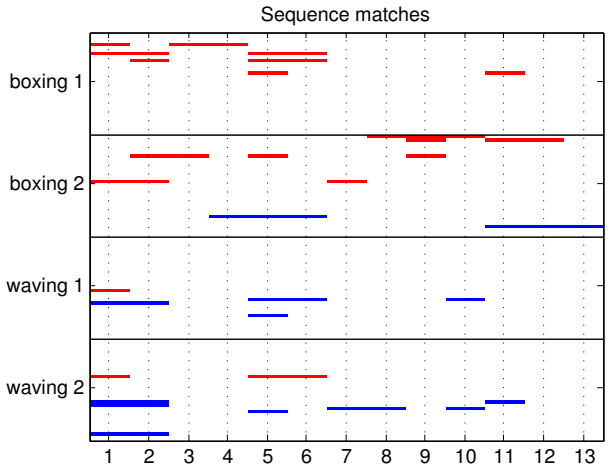


Figure 5. Visualization of how the most influential patterns match in four unseen test sequences in the boxing-vs-handwaving classifier, for the case of a 768-word codebook and 13 temporal bins. Each of the four rows shows a distinct test videos, where the first two correspond to boxing, the latter two to handwaving. We visualize the 32 pattern sequences of the decision stumps with the highest coefficient value $\alpha$. Sequences voting for boxing ($\omega = 1$) are shown at the top of each row in red (●) and sequences voting for handwaving ($\omega = -1$) are shown at the bottom of each row in blue (●). All four test sequences are classified correctly.

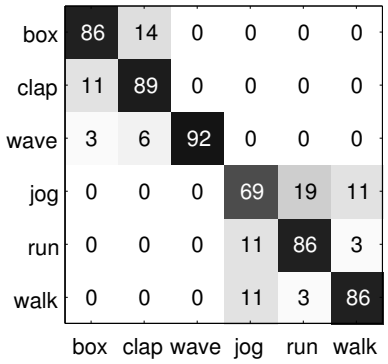


Figure 6. Confusion matrix of the Subsequence Boosting classifier on the KTH test set. The classifier was produced with a 768 element codebook, $B = 12$ and $\nu = 0.05$. Confusions happen between the boxing, hand-clapping and hand-waving classes, as well as between the jogging, running and walking classes.

## 6. Discussion

We achieved state-of-the-art classification results using our proposed algorithm and report the best results in the literature so far. Our algorithm has favorable properties, such as increased interpretability of the resulting classification function, explicit feature selection, global optimal convergence and fast testing times, but in the end we did not show a clear and significant improvement of the classification accuracy over a histogram approach with a SVM classifier and nonlinear kernel.

This is quite surprising and it is not obvious why this is the case. Possibly, the KTH data set favors a histogram based classifier because each action is quite homogeneous and does not involve global changes or complex behaviour. Also, as in the reported literature results, in our classifier the confusions happen in two clusters, namely i) boxing, handclapping and handwaving, and ii) jogging, running and walking. Each of these actions might be easily confused on both a local temporal scale as well as a coarse temporal scale, and we might very well do not gain much by including the temporal order of features.

## 7. Conclusion

We proposed a novel classifier for sequence representations, suitable for action classification in videos. A goal of our work is to make efficient pattern selection algorithms from the data mining community accessible to the computer vision community.

In the future we will apply our approach to classifiy higher order action patterns. Unfortunately, the lack of an openly available action classification data set for such high level actions is currently a problem.

The data set as well as our implementation of the PREFIXSPAN and DPREFIXSPAN algorithms are made available under the GNU General Public License at `http://www.kyb.mpg.de/bs/people/nowozin/pboost/`.

**Acknowledgments.** We thank the reviewers and Christoph Lampert and Matthias Hein for suggesting improvements to the paper. We would also like to thank Piotr Dollár for making his Matlab toolbox for the extraction of spatio-temporal interest points available to the public. This work is funded in part by the EU CLASS project, IST 027978.

## References

[1] A. Agarwal and B. Triggs. Learning to track 3D human motion from silhouettes. In C. E. Brodley, editor, *ICML*. ACM, 2004.

[2] A. F. Bobick and J. W. Davis. The recognition of human movement using temporal templates. *IEEE Trans. Pattern Anal. Mach. Intell*, 23(3):257–267, 2001.

[3] A. Demiriz, K. P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Journal of Machine Learning*, 46:225–254, 2002.

[4] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *International Workshop on Performance Evaluation of Tracking and Surveillance*, pages 65–72, 2005.

[5] A. A. Efros, A. C. Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *ICCV*, pages 726–733. IEEE Computer Society, 2003.

[6] M. Everingham, A. Zisserman, C. Williams, and L. V. Gool. The pascal visual object classes challenge 2006 (VOC2006) results. Technical report, 2006.

[7] Y. Ke, R. Sukthankar, and M. Hebert. Efficient visual event detection using volumetric features. In *ICCV*, pages 166–173. IEEE Computer Society, 2005.

[8] T. Kudo, E. Maeda, and Y. Matsumoto. An application of boosting to graph classification. In *NIPS*, 2004.

[9] I. Laptev. On space-time interest points. *International Journal of Computer Vision*, 64(2-3):107–123, 2005.

[10] S. Morishita. Computing optimal hypotheses efficiently for boosting. In S. Arikawa and A. Shinohara, editors, *Progress in Discovery Science*, volume 2281 of *Lecture Notes in Computer Science*, pages 471–481. Springer, 2002.

[11] J. C. Niebles, H. Wang, and L. Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. In *British Machine Vision Conference*, page III:1249, 2006.

[12] N. J. Nilsson. *Artificial Intelligence: A New Synthesis*. Morgan Kaufmann Publishers, San Francisco, 1998. ISBN 1558604677.

[13] S. Nowozin, K. Tsuda, T. Uno, T. Kudo, and G. Bakır. Weighted substructure mining for image analysis. In *CVPR '07: Proceedings of the 2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

[14] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Trans. Knowl. Data Eng*, 16(11):1424–1440, 2004.

[15] J. C. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAGs for multiclass classification. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *NIPS*, pages 547–553. The MIT Press, 1999.

[16] D. Ramanan and D. A. Forsyth. Automatic annotation of everyday movements. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *NIPS*. MIT Press, 2003.

[17] C. Schüldt, I. Laptev, and B. Caputo. Recognizing human actions: A local SVM approach. In *ICPR (3)*, pages 32–36, 2004.

[18] Y. Song, L. Goncalves, and P. Perona. Unsupervised learning of human motion. *IEEE Trans. Pattern Anal. Mach. Intell*, 25(7):814–827, 2003.

[19] D. Weinland, R. Ronfard, and E. Boyer. Free viewpoint action recognition using motion history volumes. *Computer Vision and Image Understanding*, 104(2-3):249–257, 2006.

[20] Y. Yacoob and M. J. Black. Parameterized modeling and recognition of activities. *Computer Vision and Image Understanding*, 73(2):232–247, 1999.

[21] A. Yilmaz and M. Shah. Actions sketch: A novel action representation. In *CVPR*, pages 984–989. IEEE Computer Society, 2005.

[22] L. Zelnik-Manor and M. Irani. Event-based analysis of video. In *CVPR*, pages 123–130. IEEE Computer Society, 2001.