Weakly Supervised Action Labeling in Videos Under Ordering Constraints

Piotr Bojanowski, Rémi Lajugie, Francis Bach, Ivan Laptev, Jean Ponce, Cordelia Schmid, Josef Sivic

Weakly Supervised Action Labeling in Videos Under Ordering Constraints

Piotr Bojanowski[1⋆] Rémi Lajugie[1⋆⋆] Francis Bach[1⋆⋆] Ivan Laptev[1⋆]
Jean Ponce[2⋆] Cordelia Schmid[1⋆⋆⋆] Josef Sivic[1⋆]

[1]INRIA [2]École Normale Supérieure

Abstract. We are given a set of video clips, each one annotated with an *ordered* list of actions, such as "walk" then "sit" then "answer phone" extracted from, for example, the associated text script. We seek to temporally localize the individual actions in each clip as well as to learn a discriminative classifier for each action. We formulate the problem as a weakly supervised temporal assignment with ordering constraints. Each video clip is divided into small time intervals and each time interval of each video clip is assigned one action label, while respecting the order in which the action labels appear in the given annotations. We show that the action label assignment can be determined together with learning a classifier for each action in a discriminative manner. We evaluate the proposed model on a new and challenging dataset of 937 video clips with a total of 787720 frames containing sequences of 16 different actions from 69 Hollywood movies.

1 Introduction

Significant progress towards action recognition in realistic video settings has been achieved in the past few years [22, 24, 26, 30, 35]. However action recognition is often cast as a classification or detection problem using fully annotated data, where the temporal boundaries of individual actions, e.g. in the form of pre-segmented video clips, are given during training. The goal of this paper is to exploit the supervisory power of the temporal ordering of actions in a video stream, as illustrated in figure 1.

Gathering fully annotated videos with accurately time-stamped action labels is quite time consuming in practice. This limits the utility of fully supervised machine learning techniques on large-scale data. Using data redundancy, weakly and semi-supervised methods are a promising alternative in this case. On the other hand, it is easy to gather videos with some level of textual annotation but poor temporal localization, from movie scripts for example. This type of weak supervisory signal has been used before in classification [22] and temporal localization [7] tasks. However, the crucial information on the ordering of actions

⋆ WILLOW project-team, DI/ENS, ENS/INRIA/CNRS UMR 8548, Paris, France.
⋆⋆ SIERRA project-team, DI/ENS, ENS/INRIA/CNRS UMR 8548, Paris, France.
⋆⋆⋆ LEAR team, INRIA Grenoble Rhône-Alpes, France.

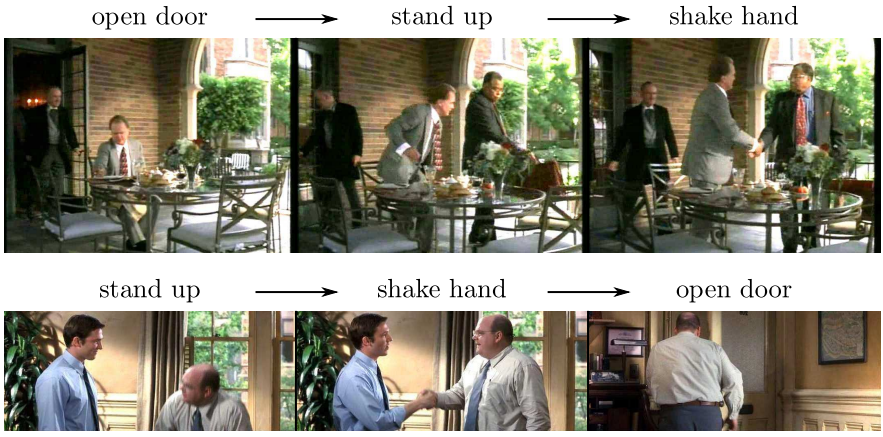

Fig. 1. Examples of video clips with associated actions sequence annotations such as provided in our dataset. Both examples contain the same set of actions but occurring in a different order. In this work we use the type and order of events as a supervisory signal to learn a classifier of each action and temporally localize each action in the video.

has, to the best of our knowledge, been ignored so far in the weakly supervised setting. Following recent work on discriminative clustering [3, 37], image [18] and video [5] cosegmentation, we propose to exploit this information in a discriminative framework where both the action model and the optimal assignments under temporal constraints are learned together.

1.1 Related Work

The temporal ordering of actions, e.g. in the form of Markov models or action grammars, have been used to constrain action prediction in videos [13, 15, 21, 23, 29, 34]. These kinds of spatial and temporal constraints have been also used in the context of group activity recognition [2, 20]. Similar to us, these papers exploit the temporal structure of videos, but focus on inferring action sequences from noisy but pre-defined action detectors, often in constrained surveillance and laboratory settings with a limited number of actions and static cameras. In contrast, in this work we explore the temporal structure of actions for learning action classifiers in a weakly supervised set-up and show results on challenging videos from feature length movies.

Related is also work on recognition of **composite activities** [28], where atomic action models ("cut", "open") are learned given full supervision on a cooking video dataset. Composite activity models ("prepare pizza") are learned on top of the atomic actions, using the prediction scores for the atomic actions as features. Annotations are, however, used without taking into account the ordering of actions.

Temporal models for recognition **of individual actions** have been explored in e.g. [22, 26, 33]. Implicit models in the form of temporal pyramids have been used with bag-of-features representations [22]. Others have used more explicit temporal models in the form of, e.g. latent action parts [26] or hidden Markov models [33]. Contrary to these methods, we do not use an a priori model of the temporal structure of individual actions, but instead exploit the given ordering constraints between actions to learn better individual actions models.

Weak supervision for learning actions has been explored in [5, 7, 22]. These methods use uncertain temporal annotations of actions provided by movie scripts. Contrary to these works our method learns multiple actions simultaneously and incorporates temporal ordering constraints on action labels obtained, e.g. from the movie scripts.

Dynamic time warping algorithms (DTW) can be used to match temporal sequences, and are extensively used in speech recognition, e.g. [9, 27]. In computer vision, the temporal order of events has been exploited in [25], where a DTW-like algorithm is used at test time to improve the performance of non-maximum suppression on the output of pre-trained action detectors.

Discriminative clustering is an unsupervised method that partitions data by minimizing a discriminative objective, optimizing over both classifiers and labels [3, 37]. Convex formulations of discriminative clustering have been explored in [3, 10]. In computer vision these methods have been successfully applied to co-segmentation [19]. The approach presented in this paper is inspired by this framework, but adds to it the use of ordering constraints.

In this work, we make use of the **Frank-Wolfe algorithm** (a.k.a conditional gradient) to minimize our cost function. The Frank-Wolfe algorithm [8, 17] is a classical convex optimization procedure that permits optimizing a continuously differentiable convex function over a convex compact domain only by optimizing linear functions over the domain. In particular, it does not require any projection steps. It has recently received increased attention in the context of large-scale optimization [11, 17].

1.2 Problem Statement and Contributions

The temporal assignment problem addressed in the rest of this paper and illustrated by Fig. 1 can be stated as follows: We are given a set of N video clips (or clips for short in what follows). A clip is defined as a contiguous video segment consisting of F frames, and may correspond, for example, to a scene (as defined in a movie script) or a collection of subsequent shots. Each clip is divided into T small *time intervals* (chunks of videos consisting of $F/T = 10$ frames in our case), and annotated by an ordered list of K elements taken from some action set $\mathcal{A}$ of size $A = |\mathcal{A}|$ (that may consist of labels such as "open door", "stand up", "answer phone", etc., as in Fig. 1 for example). Note that clips are not of the same length but for the sake of simplicity, we assume they are. We address the problem of assigning to each time interval of each clip one action in $\mathcal{A}$, respecting the order in which the actions appear in the original annotation list (Fig. 2).

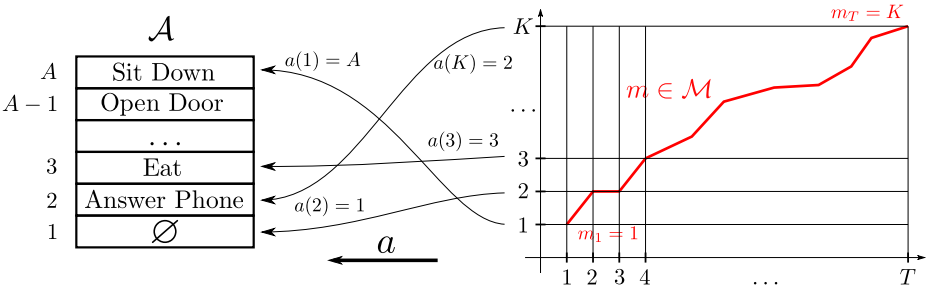

Fig. 2. Right: The goal is to find assignment of video intervals 1 to T (x-axis) to the ordered list of action annotations $a(1)$ to $a(K)$ indexed by integer k from 1 to K (y-axis). Left: The ordered annotation index k is mapped, through mapping a to action labels from the set $\mathcal{A}$, in that example $a(3)$ ="Eat". To preserve the given ordering of annotations we only consider assignments $\mathcal{M}$ that are non-decreasing. One such assignment m is shown in red.

Contributions. We make the following contributions: **(i)** we propose a discriminative clustering model (section 2) that handles weak supervision in the form of temporal ordering constraints and recovers a classifier for each action together with the temporal localization of each action in each video clip; **(ii)** we design a convex relaxation of the proposed model and show it can be efficiently solved using the conditional gradient (Frank-Wolfe) algorithm (section 3); and finally **(iii)** we demonstrate improved performance of our model on a new action dataset for the tasks of temporal localization (section 6) and action classification (section 7). All the data and code are publicly available at `http://www.di.ens.fr/willow/research/ordering`.

2 Discriminative Clustering with Ordering Constraints

In this section we describe the proposed discriminative clustering model that incorporates label ordering constraints. The input is a set of video clips, each annotated with an ordered list of action labels specifying the sequence of actions present in the clip. The output is the temporal assignment of actions to individual time intervals in each clip respecting the ordering constraint provided by the annotations together with a learnt classifier for each action, common for all clips. In the following, we first formulate the temporal assignment of actions to individual frames as discriminative clustering (section 2.1), then introduce a parametrization of temporal assignments using indicator variables (section 2.2), and finally we describe the choice of a loss function for the discriminative clustering that leads to a convex cost (section 2.3).

2.1 Problem Formulation

Let us now formalize the temporal assignment problem. We denote by $x_n(t)$ in $\mathbb{R}^d$ some local descriptor of video clip number n during time interval number t. For

every k in $\{1,\dots,K\}$, we also define $a_n(k)$ as the element of $\mathcal{A}$ corresponding to annotation number k (Fig. 2). Note that the set of actions $\mathcal{A}$ itself is not ordered: even if we represent $\mathcal{A}$ by a table for convenience, the elements of this table are action labels and have no natural order. The annotations, on the other hand, are ordered, for example according to where they occur in a movie script, and are represented by some integer between 1 and K. Thus a_n maps (ordered) annotation indices onto (unordered) actions, and depends of course on the video clip under annotation. Parts of any video clip may belong to the background. To account for this fact, a dummy label $\varnothing$ is inserted in the annotation list between every consecutive pair of actual labels.

Let us denote by $\mathcal{M}$ the set of *admissible assignments* on $\{1,\dots,T\}$, that is, the set of sequences $m = (m_1,\dots,m_T)$ with elements in $\{1,\dots,K\}$ such that $m_1 = 1$, $m_T = K$, and $m_{t+1} = m_t$ or $m_{t+1} = m_t + 1$ for all t in $\{1,\dots,T-1\}$ Such an assignment is illustrated in Fig. 2.

Let us also denote by $\mathcal{F}$ the space of classifiers of interest, by $\Omega : \mathcal{F} \to \mathbb{R}$ some regularizer on this space and by $\ell : \mathcal{A} \times \mathbb{R}^A \to \mathbb{R}_+$ an appropriate loss function. For a given clip n and a fixed classifier f, the problem of assigning the clip intervals to the annotation sequence can be written as the minimization of the cost function:

$$E(m, f, n) = \frac{1}{T} \sum_{t=1}^{T} \ell\left(a_n(m_t), f(x_n(t))\right) \tag{1}$$

with respect to assignment m in $\mathcal{M}$. The regularizer Ω prevents overfitting and we therefore define a scalar parameter λ to control this effect. Jointly learning the classifiers and solving the assignment problem corresponds to the following optimization problem:

$$\min_{f \in \mathcal{F}} \left[\sum_{n=1}^{N} \min_{m \in \mathcal{M}} E(m, f, n) \right] + \lambda \Omega(f). \tag{2}$$

2.2 Parameterization Using an Assignment Matrix

As will be shown in the following sections, it is convenient to reformulate our problem in terms of indicator variables. The corresponding multi-class loss is $\ell : \{0,1\}^A \times \mathbb{R}^A \to \mathbb{R}_+$, and the classifiers are functions $f : \mathbb{R}^d \to \mathbb{R}^A$. For a clip n, let us define the *assignment matrix* $Z^n \in \mathbb{R}^{T \times A}$ which is composed of entries z^n_{ta} such that $z^n_{ta} = 1$ if the interval t of clip n is assigned to class a.

Let Z^n_t denote the row vector of dimension A corresponding to the t-th row of Z^n . The cost function $E(m, f, n)$, defined in Eq. (1) can be rewritten as $\frac{1}{T}\sum_{t=1}^{T} \ell\left(Z^n_t, f(x_n(t))\right)$.

Note: To avoid cumbersome double summations, we suppose from now that we work with a single clip. This allows us to drop the superscript notation, we replace Z^n by Z and skip the sum over clips. We also replace the descriptor notation $x_n(t)$ by x_t and the row extraction notation Z^n_t by Z_t. This is without

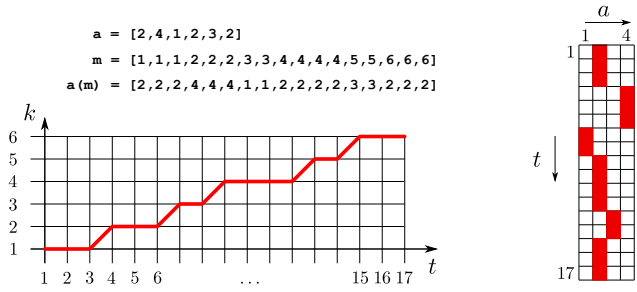

Fig. 3. Illustration of the correspondence between temporal assignments (left) and associated valid assignment matrices that map action labels a to time intervals t (right). **Left**: a valid assignment non-decreasing $m_t = k$. **Right**: the corresponding assignment matrix Z. One can build the assignment matrix Z given the assignment m and the annotation sequence a by putting a 1 at index $(t, a(m_t))$ in Z for every t. One obtains m given Z by iteratively constructing a sequence of integers of length T such that $m_{t+1} = m_t$ if the t-th and $(t+1)$-th row of Z are identical, and $m_{t+1} = m_t + 1$ otherwise.

loss of generality, and our method as described in the sequel handles multiple clips with some simple bookkeeping.

Because of temporal constraints, we want the assignment matrices Z to correspond to valid assignments m. This amounts to imposing some constraints on Z. Let us therefore define $\mathcal{Z}$, the set of all valid assignment matrices as:

$$\mathcal{Z} = \left\{ Z \in \{0,1\}^{T \times A} \mid \exists\, m \in \mathcal{M},\ \text{s.t.},\ \forall\, t,\ Z_{ta} = 1 \iff a(m_t) = a \right\}. \quad (3)$$

There is a bijection between the sets $\mathcal{Z}$ and $\mathcal{M}$. For each m in $\mathcal{M}$ there exists a unique corresponding Z in $\mathcal{Z}$ and *vice versa*. Figure 3 gives an intuitive illustration of this bijection.

The set $\mathcal{Z}$ is a subset of the set of stochastic matrices (positive matrices whose rows sum up to 1), formed by the matrices whose columns consist of exactly K blocks of contiguous ones occurring in a predefined order ($K = 6$ in Fig. 3). There are as many elements in $\mathcal{Z}$ as ways of choosing $(K-1)$ transitions among $(T-1)$ possibilities, thus $|\mathcal{Z}| = \binom{T-1}{K-1}$, which can be extremely large in our setting (in our setting $T \approx 100$ and $K \approx 10$). Furthermore, it is very difficult to describe explicitly the algebraic constraints on stochastic matrices that define $\mathcal{Z}$. This point will prove important in Sec. 3 when we propose an optimization algorithm for learning our model. Using these notations, Eq. (2) is equivalent to:

$$\min_{f \in \mathcal{F}, Z \in \mathcal{Z}} \frac{1}{T} \sum_{t=1}^{T} \ell\left(Z_t, f(x_t)\right) + \lambda \Omega(f). \quad (4)$$

2.3 Quadratic Cost Functions

We now choose specific functions ℓ and f that will lead to a quadratic cost function. This choice leads, to a convex relaxation of our problem. We use multi-

class linear classifiers of the form $f(x) = x^T W + b$, where $W \in \mathbb{R}^{d \times A}$ and $b \in \mathbb{R}^{1 \times A}$. We choose the square loss function, regularized with the Frobenius norm of W, because in that case the optimal parameters W and b can be computed in closed form through matrix inversion. Let X be the matrix in $\mathbb{R}^{T \times d}$ formed by the concatenation of all $1 \times d$ matrices x_t. For this choice of loss and regularizer, our objective function can be rewritten using the matrices defined above as:

$$\frac{1}{T}\sum_{t=1}^{T} \ell(Z_t, f(x_t)) + \lambda\Omega(f) = \frac{1}{T}\|Z - XW - b\|_F^2 + \frac{\lambda}{2}\|W\|_F^2. \tag{5}$$

This is exactly a ridge regression cost. Minimizing this cost with respect to W and b for fixed Z can be done in closed form [3, 12]. Setting the partial derivatives with respect to W and b to zero and plugging the solution back yields the following equivalent problem:

$$\min_{Z \in \mathcal{Z}} \operatorname{Tr}\left(ZZ^T B\right), \text{ where } B = \frac{1}{T}\Pi_T(I_T - X\left(X^T\Pi_T X + T\lambda I_d\right)^{-1} X^T)\Pi_T, \tag{6}$$

and the matrix Π_p is the $p \times p$ centering matrix $I_p - \frac{1}{p}\mathbf{1}_p\mathbf{1}_p^T$. This corresponds to implicitly learning the classifier while finding the optimal Z by solving a quadratic optimisation problem in Z. The implicit classifier parameters W and b are shared among all video clips and can be recovered in closed-form as:

$$W = (X^T\Pi_d X + \lambda I)^{-1} X^T \Pi_T Z D^{1/2}, \qquad b = \frac{1}{T}\mathbf{1}^T(Z - Xw)D^{1/2}. \tag{7}$$

3 Convex Relaxation and the Frank-Wolfe Algorithm

In Sec. 2, we have seen that our model can be interpreted as the minimization of a convex quadratic function (B is positive semidefinite) over a very large but discrete domain. As is usual for this type of hard combinatorial optimization problem, we replace the discrete set $\mathcal{Z}$ by its convex hull $\overline{\mathcal{Z}}$. This allows us to find a continuous solution of the relaxed problem using an appropriate and efficient algorithm for convex optimization.

3.1 The Frank-Wolfe Algorithm

We want to carry out the minimization of a convex function over a complex polytope $\overline{\mathcal{Z}}$, defined as the convex hull of a large but finite set of integer points defined by the constraints associated with admissible assignments. When it is possible to optimize a linear function over a constraint set of this kind, but other usual operations (like projections) are not tractable, a good way to optimize a convex objective function is to use the iterative Frank-Wolfe algorithm (a.k.a. conditional gradient method) [4, 8]. We show in Sec. 3.2 that we can minimize linear functions over $\overline{\mathcal{Z}}$, so this is an appropriate choice in our case.

The idea behind the Frank-Wolfe algorithm is rather simple. An affine approximation of the objective function is minimized yielding a point Z^* on the

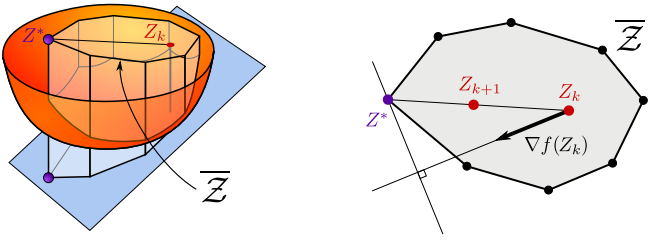

Fig. 4. Illustration of a Frank-Wolfe step (see [17] for more details). **Left**: the domain $\overline{\mathcal{Z}}$ interest, objective function (red), and its linearization at current point (blue). **Right**: top view of $\overline{\mathcal{Z}}$. Note that, in the algorithm, we actually minimize a linear function at each step. Adding a constant to it does not affect the solution of the minimization problem. That is why, we depicted an hyperplane that seems shifted from the origin.

edge of $\overline{\mathcal{Z}}$. Then a convex combination of Z^* and the current point Z_k is computed. This is repeated until convergence (see Alg. 1). The interpolation parameter γ can be chosen either by using the universal step size $\frac{2}{p+1}$, where p is the iteration counter (see [17] and references therein) or, in the case of quadratic functions, by solving a univariate quadratic equation. In our implementation, we use the latter. A good feature of the Frank-Wolfe algorithm is that it provides a duality gap (referred to as the linearization duality gap [17]) that can be used as a certificate of sub-optimality and stopping criterion. The procedure is described in the special case of our relaxed problem in Algorithm 1. Figure 4 illustrates one step of the optimization.

3.2 Minimizing Linear Functions over $\overline{\mathcal{Z}}$ by Dynamic Programming

It is possible to minimize linear functions over the binary set $\mathcal{Z}$. Simple arguments (see for instance Prop B.21 of [4]) show that the solution over $\mathcal{Z}$ is also a solution over $\overline{\mathcal{Z}}$. We will therefore focus on the minimization problem on $\mathcal{Z}$ and keep in mind that it also gives a solution over $\overline{\mathcal{Z}}$ as required by the Frank-Wolfe algorithm. Minimizing a linear function on $\mathcal{Z}$ amounts to solving the problem: $\min_{Z\in\mathcal{Z}} \operatorname{Tr}\left(C^T Z\right) = \sum_{t=1}^{T}\sum_{a=1}^{A} Z_{ta}C_{ta}$, where C is a cost matrix in $\mathbb{R}^{T\times A}$. Using the equivalence between the assignment matrix (Z) and

$k \leftarrow 0$
while $\operatorname{Tr}(\nabla_{\mathrm{f}}(\mathrm{Z_k})(\mathrm{Z_k} - \mathrm{Z}^*)) \geq \epsilon$ **do**
  Compute the current gradient in Z, $\nabla_f(Z_k) = Z_k^T B$.
  Choose Z^* in $\arg\min_{Z\in\overline{\mathcal{Z}}} \operatorname{Tr}(Z\nabla_f(Z_k))$ using dynamic programming.
  Compute the optimal Frank-Wolfe step size γ.
  $Z_{k+1} = Z_k + \gamma(Z^* - Z_k)$
  $k \leftarrow k + 1$.
end

Algorithm 1: The Frank-Wolfe optimization procedure.

the plain assignment (m) representations (Fig. 3), this is equivalent to solving $\min_{m\in\mathcal{M}} \sum_{t=1}^{T}\sum_{a=1}^{A} \mathbb{1}_{a(m_t)=a} C_{ta}$. To better deal with the temporal structure of the assignment, let us denote by $D \in \mathbb{R}^{T\times K}$ the matrix with entries $D_{tk} = C_{ta(k)}$. The minimization problem then becomes $\min_{m\in\mathcal{M}} \sum_{t=1}^{T} D_{tm_t}$, which can be solved using dynamic time warping. Indeed, let us define for all $t \in \{1,\dots,T\}$ and $k \in \{1,\dots,K\}$: $P_t^*(k) = \min_{m\in\mathcal{M}} \sum_{s=1}^{t} D_{sm_s}$. We can think of $P_t^*(k)$ as the cost of the optimal path from $(1,1)$ to (t,k) in the graph defined by admissible assignments, and we have the following dynamic programming recursion: $P_t^*(k) = D_{tk} + \min(P_{t-1}^*(k-1), P_{t-1}^*(k))$.

The optimal value $P_T^*(K)$ can be computed in $O(TK)$ using dynamic programming, by precomputing the matrix D, incrementally computing the corresponding $P_t^*(k)$ values, and maintaining at each node (t,k) back pointers to the appropriate neighbors.

3.3 Rounding

At convergence, the Frank-Wolfe algorithm finds the (non-integer) global optimum $\hat{Z}$ of Eq. (6) over $\overline{\mathcal{Z}}$. Given $\hat{Z}$, we want to find an appropriate nearby point Z in $\mathcal{Z}$. The simplest geometric rounding scheme consists in finding the closest point of $\mathcal{Z}$ according to the Frobenius distance : $\min_{Z\in\mathcal{Z}} \|\hat{Z} - Z\|_F^2$. Expanding the norm yields: $\|\hat{Z} - Z\|_F^2 = \mathrm{Tr}(\hat{Z}^T\hat{Z}) + \mathrm{Tr}(Z^TZ) - 2\mathrm{Tr}(\hat{Z}^TZ)$.

Since $\hat{Z}$ is fixed, its norm is a constant. Moreover, since Z is an element of $\mathcal{Z}$, its squared norm is constant and equal to T. The rounding problem is therefore equivalent to: $\min_{Z\in\mathcal{Z}} -2\mathrm{Tr}(\hat{Z}^TZ)$, that is to the minimization of a linear function over $\overline{\mathcal{Z}}$. This can be done, as in Sec. 3.2, using dynamic programming.

4 Practical Concerns

In this section, we detail some refinements of our model. First, we show how to tackle a semi-supervised setting where some time-stamped annotations are available. Second, we discuss how to avoid the trivial solutions, a common issue in discriminative clustering methods [3, 10, 18].

4.1 Semi-supervised Setting

Let us suppose that we are given some fully annotated clips (in the sense that they are labeled with time-stamped annotations), corresponding to a total of L time intervals. For every interval l we have a descriptor X_l in $\mathbb{R}^d$ and a class label a_l in $\mathcal{A}$. We can incorporate this data by modifying the optimization problem as follows:

$$\min_{f\in\mathcal{F}} \left[\min_{m\in\mathcal{M}} E(m,f,n)\right] + \frac{1}{L}\sum_{l=1}^{L} \ell(a_l, f(X_l)) + \lambda\Omega(f). \tag{8}$$

The first term is the weakly supervised assignment cost and the second one is the loss on annotated data. This supervised model does not change the optimization procedure, which remains valid.

4.2 Minimum Size Constraints

There are two inherent problems with discriminative clustering. First, the constant assignment matrix is typically a trivial optimum because, as explained in [10], the optimization domain is symmetruc over the permutations of labels. Thanks to our temporal constraints, the set $\mathcal{Z}$ is not symmetric and thus we are not subject to this effect. The second difficulty is linked to the use of the centering matrix Π_T in Eq. (6). Indeed, we notice that the constant vector of length T is an eigen vector of Π_T and thus the column-wise constant matrices are trivial solutions to our problem. Due to the temporal ordering constraints these solutions are not admissible, but in practice we have noticed that the algorithm returned an assignment with the label $\varnothing$ being dominant. We cope with these issues by adding a class penalizing linear term and by weighting the different classes. We refer the reader to [6] for more details.

5 Dataset and Features

Dataset. Our input data consists of challenging video clips annotated with sequences of actions. One possible source for such data is movies with their associated scripts [5, 7, 22, 32]. The annotations provided by this kind of data are noisy and do not provide ground-truth time-stamps for evaluation. To address this issue, we have constructed a new action dataset, containing clips annotated by sequences of actions. We have taken the 69 movies from which the clips of the Hollywood2 dataset were extracted [22], and manually added full time-stamped annotation for 16 classes (12 of these classes are already present in Hollywood2). To build clips that form our input data, we search in the annotations for action chains containing at least two actions. To do so, we pad the temporal action annotations by 250 frames and search for overlapping intervals. A chain of such overlapping annotations forms one video clip with associated action sequence in our dataset. In the end we obtain 937 clips, with the number of actions ranging from 2 to 11. We subdivide each clip into temporal intervals of length 10 frames. Clips contain on average 84 intervals, the shortest containing 11, the longest 289.

Feature representation. We have to define a feature vector for every interval of a clip. We build a bag-of-words vector x_t per interval t. Recall that intervals are of length 10 frames. To aggregate enough features, we decided to pool features from the 30-frame-long window centered on the interval. We compute video descriptors following [36]. We generate a vocabulary of size 2000 for HOF features. We restricted ourselves to one channel to improve the running time, while being aware that by doing so we sacrifice some performance. In our informal experiments, we also tried the MBH channels yielding very close performance. We use the Hellinger kernel to obtain the explicit feature map by square-rooting the l_1 normalized histograms. Every data point is associated with a vector x_t in $\mathbb{R}^{2000}$.

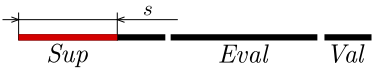

Fig. 5. Splitting of the data described in Sec. 6.

6 Action Labeling Experiments

Experimental Setup. To carry out the action labeling experiments, we split 90% of the dataset into three parts (Fig. 5) that we denote *Sup* (for supervised), *Eval* (for evaluation) and *Val* (for validation). *Sup* is the part of data that has time-stamped annotations, and it is used only in the semi-supervised setting described in Sec. 4.1. *Val* is the set of examples on which we automatically adjust the hyper-parameters for our method (λ, κ, D). In practice we fix the *Val* set to contain 5% of the dataset. This set is provided with fully time-stamped annotations, but these are not used during the cost optimization. None of the reported results are computed on this set. We evaluate the quality of the assignment on the *Eval* set. Note that we carry out the Frank-Wolfe optimization on the union of all three sets. The annotations from the *Sup* set are used to constrain Z in the semi-supervised setup while those from the *Val* set are only used for choosing our hyper parameters. The supervisory information used over the rest of the data are the ordered annotations without time stamps. Please also keep in mind that there are no "training" and "testing" phases *per se* in this primary assignment task. All our experiments are conducted over five random splits of the data. This allows us to present results with error bars.

Performance Measure. Several measures may be used to evaluate the performance of discriminative clustering algorithms. Some authors propose to use the output classifier to perform a classification task [7, 37] or use the output partition of the data as a solution of the segmentation task [18]. Yet another way to evaluate is to use a loss between partitions [14] as in [3]. Note that because of temporal constraints, for every clip we have a set of corresponding (prediction, ground-truth) pairs. We have thus chosen to measure the assignment quality for every ground-truth action interval I^* and prediction I as $|I \cap I^*|/|I|$. This measure is similar to the standard Jaccard measure used for comparing ensembles [16]. Therefore, with a slight abuse of notation, we refer to this measure as the Jaccard measure. This performance measure is well suited for our problem since it respects the following properties: **(1)** it is high if the action predicted is included in the ground-truth annotation, **(2)** it is low if the prediction is bigger than the annotation, **(3)** it is lowest if the prediction is out of the annotation, **(4)** it does not take into account the prediction of the background class. The score is averaged across all ground-truth intervals. The perfect score of 1 is achieved when all actions are aligned to the correct annotations, but accurate temporal segmentation is not required as long as the predicted labels are within the ground truth interval.

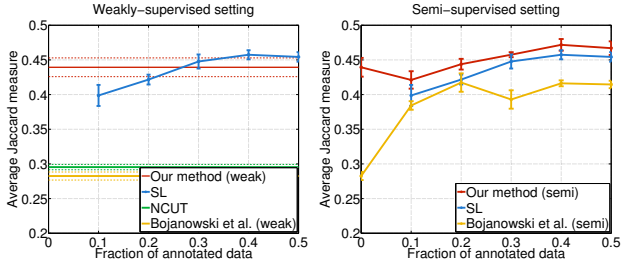

Fig. 6. Alignment evaluation for all considered models. **Left**: weakly-supervised methods. This graph is shown for various fractions of fully supervised data only to compare to the SL baseline. "Weak" methods do not make use of this supervision. **Right**: semi-supervised methods. See project webpage [1] for qualitative results.

Baselines. We compare our method to the three following baselines. All these are trained using the same features as the ones used for our method. For all baselines, we round the obtained solution Z using the scheme described in Sec. 3.3.

Normalized Cuts (NCUT). We compare our method to normalized cuts (or spectral clustering) [31]. Let us define B as the symmetric Laplacian of the matrix E: $B = I - D^{-\frac{1}{2}} E D^{-\frac{1}{2}}$, where $D = \mathrm{Diag}\,(E\mathbf{1})$. E measures both the proximity and appearance similarity of intervals. For all (i, j) in $\{1, \ldots, T\}^2$, we compute: $E_{ij} = e^{-\alpha|i-j| - \beta d_{\chi^2}(X_i, X_j)}\ \mathbb{1}_{|i-j|<d_{\min}}$, where d_{χ^2} is the Chi-squared distance. More precisely, we minimize over all cuts Z the cost $g(Z) = \mathrm{Tr}\left(ZZ^T B\right)$. g is convex (B is positive semidefinite) and we can use the Frank-Wolfe optimization scheme developed for our model. Intuitively, this baseline is searching for a partition of the video such that time intervals falling into the same segments have close-by features according to the Chi-squared distance.

Bojanowski et al. [5]. We also consider our own implementation of the weakly-supervised approach proposed in [5]. We replace our ordering constraints by the corresponding "at least one" constraints. When an action is mentioned in the sequence, we require it appears at least once in the clip. This corresponds to a set of linear constraints on Z. We adapt this technique in order to work on our dataset. Indeed, the available implementation requires storing a square matrix of the size of the problem. Instead, we choose to minimize the convex objective of [5] using the Frank-Wolfe algorithm which is more scalable.

Supervised Square Loss (SL). For completeness, we also compare our method to a fully supervised approach. We train a classifier using the square loss over the annotated *Sup* set and score all time intervals in *Eval*. We use the square loss since it is used in our method and all other baselines.

Weakly Supervised Setup. In this setup, all baselines except (SL) have only access to weak supervision in the form of ordering constraints. Figure 6 (left) illustrates the quality of the predicted asignmentss and compares our method to baselines. Our method performs better than all other weakly-supervised methods. Both the Bojanowski et al. and NCUT baselines have low scores in the

weakly-supervised setting. This shows the advantage of exploiting temporal constraints as weak supervisory signal. The fully supervised baseline (blue) eventually recovers a better alignment than our method as the fraction of fully annotated data increases. This occurs (when the red line crosses the blue line) at the 25% mark, as the supervised data makes up for the lack of ordering constraints. Fully time-stamped annotated data are expensive to produce whereas movies scripts are often easy to get. It appears thus that manually annotated videos are not always necessary since good performance is reached simply by using weak supervision. Figure 7 shows the results for all weakly-supervised methods for all classes. We notice that we outperform the baselines on the most frequent classes (such as "Open Door", "Sit Down" and "Stand Up").

Semi-supervised Setup. Figure 6 (right) illustrates the performance of our model when some supervised data is available. The fraction of the supervised data is given on the x-axis. First, note that our semi-supervised method (red) is always and consistently (Cf error bars) above the square loss baseline (blue). Of course, during the optimization, our method has access to weak annotations over the whole dataset, and to full annotations on the *Sup* set whereas the SL baseline has access only to the latter. This demonstrates the benefits of exploiting temporal constraints during learning. The semi-supervised Bojanowski et al. baseline (orange) has low performance, but it improves with the amount of full supervision provided.

7 Classification Experiments

The experiments in the previous section evaluate the quality of the recovered assignment matrix Z. Here we evaluate instead the quality of the recovered classifiers on a held-out test set of data for an action classification task. We recover these classifiers as explained later in this section. We can treat them as K independent, one-versus-rest classifiers and use them to score the samples from the test set. We evaluate this performance by computing per-class precision and recall and report the corresponding average precision for each class.

Experimental setup. The models are trained following the procedure described in the previous section. To test the performance of our classifiers, we

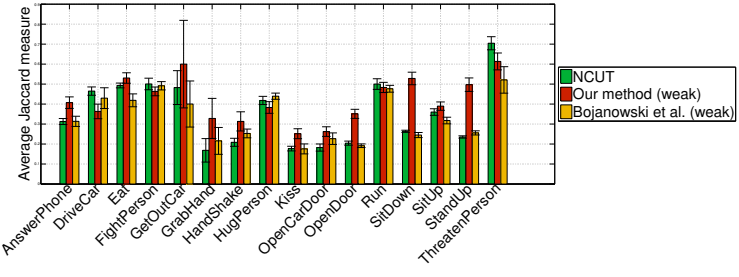

Fig. 7. Alignment performance for various weakly-supervised methods for all classes.

use the held out set of clips. This set is made of 10% of the clips from the original data. The clips from this set are identical in nature to the ones used to train the models. We also perform multiple random splits to report results with error bars.

Recovering the classifiers. One of the nice features of our method is that we can estimate the implicit classifiers corresponding to our solution Z^*. We do so using the expression from Eq. 7.

Baselines. We compare the classifiers obtained by our method to those obtained by the Bojanowski et al. baseline [5]. We also compare them to the classifiers learned using the (SL) baseline.

Weakly Supervised Setup. Classification results are presented in Fig. 8 (left). We observe a behavior similar to the action labeling experiment. But the supervised classifier (SL) trained on the *Sup* set using the square loss (blue) always performs worse than our model (red). This can be explained by the fact that the proposed model makes use of mode data. Even though our model has only access to weak annotation, it can prove sufficient to train good classifiers. The weakly-supervised method from Bojanowski et al. (orange) is performing worst, exactly as in the previous task. This can be explained by the fact that this method does not have access to full supervision or ordering constraints.

Semi-supervised Setup. In the semi-supervised setting (Fig. 8 (left)), our method (red) performs better than the supervised SL baseline (blue). The action model we recover is consistently better than the one obtained using only fully supervised data. Thus, our method is able to perform well semi-supervised learning. The Bojanowski et al. baseline (orange) improves when the fraction of annotated examples increases. Nonetheless, we see that making use of ordering constraints as used by our method signigicantly improves over simple linear inequalities ("at least one" constraints as formulated in [5]).

Acknowledgements. This work was supported by the European integrated project AXES, the MSR-INRIA laboratory, EIT-ICT labs, a Google Research Award, a PhD fellowship from the EADS Foundation, the Institut Universitaire de France and ERC grants ALLEGRO, VideoWorld, Activia and Sierra.

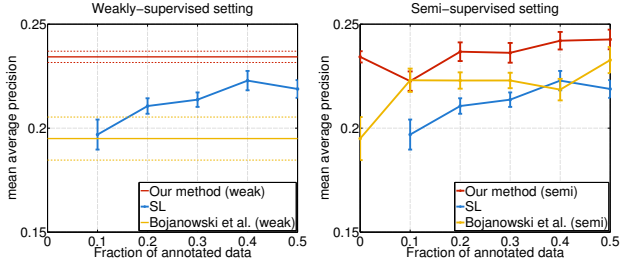

Fig. 8. Classification performance for various models. **Left**: weakly-supervised methods. **Right**: semi-supervised methods. See project webpage [1] for qualitative results.

References

1. http://www.di.ens.fr/willow/research/actoraction/
2. Amer, M.R., Todorovic, S., Fern, A., Zhu, S.C.: Monte carlo tree search for scheduling activity recognition. In: ICCV (2013)
3. Bach, F., Harchaoui, Z.: DIFFRAC: a discriminative and flexible framework for clustering. In: NIPS (2007)
4. Bertsekas, D.: Nonlinear Programming. Athena Scientific (1999)
5. Bojanowski, P., Bach, F., Laptev, I., Ponce, J., Schmid, C., Sivic, J.: Finding Actors and Actions in Movies. In: ICCV (2013)
6. Bojanowski, P., Lajugie, R., Bach, F., Laptev, I., Ponce, J., Schmid, C., Sivic, J.: Weakly Supervised Action Labeling in Videos Under Ordering Constraints. In: arXiv (2014)
7. Duchenne, O., Laptev, I., Sivic, J., Bach, F., Ponce, J.: Automatic annotation of human actions in video. In: ICCV (2009)
8. Frank, M., Wolfe, P.: An algorithm for quadratic programming. Naval Research Logistics Quarterly (1956)
9. Gold, B., Morgan, N., Ellis, D.: Speech and Audio Signal Processing - Processing and Perception of Speech and Music, Second Edition. Wiley (2011)
10. Guo, Y., Schuurmans, D.: Convex Relaxations of Latent Variable Training. In: NIPS (2007)
11. Harchaoui, Z.: Conditional gradient algorithms for machine learning. In: NIPS Workshop (2012)
12. Hastie, T., Tibshirani, R., Friedman, J.: The elements of statistical learning: data mining, inference and prediction. Springer (2009)
13. Hongeng, S., Nevatia, R.: Large-scale event detection using semi-hidden markov models. In: ICCV (2003)
14. Hubert, L., Arabie, P.: Comparing partitions. Journal of classification (1985)
15. Ivanov, Y.A., Bobick, A.F.: Recognition of visual activities and interactions by stochastic parsing. PAMI (2000)
16. Jaccard, P.: The distribution of the flora in the alpine zone. New Phytologist (1912)
17. Jaggi, M.: Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In: ICML (2013)
18. Joulin, A., Bach, F., Ponce, J.: Discriminative Clustering for Image Co-segmentation. In: CVPR (2010)
19. Joulin, A., Bach, F., Ponce, J.: Multi-class cosegmentation. In: CVPR (2012)
20. Khamis, S., Morariu, V.I., Davis, L.S.: Combining per-frame and per-track cues for multi-person action recognition. In: ECCV (2012)
21. Kwak, S., Han, B., Han, J.H.: Scenario-based video event recognition by constraint flow. In: CVPR (2011)
22. Laptev, I., Marszalek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. In: CVPR (2008)
23. Laxton, B., Lim, J., Kriegman, D.J.: Leveraging temporal, contextual and ordering constraints for recognizing complex activities in video. In: CVPR (2007)
24. Liu, J., Kuipers, B., Savarese, S.: Recognizing human actions by attributes. In: CVPR (2011)
25. Nguyen, M.H., Lan, Z.Z., la Torre, F.D.: Joint segmentation and classification of human actions in video. In: CVPR (2011)
26. Niebles, J.C., Chen, C.W., Li, F.F.: Modeling Temporal Structure of Decomposable Motion Segments for Activity Classification. In: ECCV (2010)

27. Rabiner, L.R., Juang, B.H.: Fundamentals of speech recognition. Prentice Hall (1993)
28. Rohrbach, M., Regneri, M., Andriluka, M., Amin, S., Pinkal, M., Schiele, B.: Script Data for Attribute-Based Recognition of Composite Activities. In: ECCV (2012)
29. Ryoo, M.S., Aggarwal, J.K.: Recognition of composite human activities through context-free grammar based representation. In: CVPR (2006)
30. Sadanand, S., Corso, J.J.: Action bank: A high-level representation of activity in video. In: CVPR (2012)
31. Shi, J., Malik, J.: Normalized Cuts and Image Segmentation. In: CVPR (1997)
32. Sivic, J., Everingham, M., Zisserman, A.: "Who are you?" - Learning person specific classifiers from video. In: CVPR (2009)
33. Tang, K., Fei-Fei, L., Koller, D.: Learning latent temporal structure for complex event detection. In: CVPR (2012)
34. Vu, V.T., Bremond, F., Thonnat, M.: Automatic video interpretation: A novel algorithm for temporal scenario recognition. In: IJCAI (2003)
35. Wang, H., Kläser, A., Schmid, C., Liu, C.L.: Action recognition by dense trajectories. In: CVPR (2011)
36. Wang, H., Schmid, C.: Action Recognition with Improved Trajectories. In: ICCV (2013)
37. Xu, L., Neufeld, J., Larson, B., Schuurmans, D.: Maximum Margin Clustering. In: NIPS (2004)