Draft - Hierarchical Discriminative learning of Activities and Actions

October 31, 2015

1 Background

We plan to learn an hierarchical model to be applied in video sequences of length (number of frames) T for classification of human activities, each composed by a variable number of atomic actions. In each frame, we define R regions, each characterized by a vector x_r, with $r = 1, \dots, R$. A different action could be assigned to each region in each frame, and the complete sequence must be associated to a single activity.

1.1 Definitions

- T: number of frames of the sequence.
- R: number of spatial regions of each frame.
- C: number of activities
- A: number of atomic actions
- K: size of pose dictionary
- t: index of frame, $t \in \{1, \dots, T\}$.
- r: index of region, $r \in \{1, \dots, R\}$.
- c: index of activity, $c \in \{1, \dots, C\}$.
- a, a': index of atomic action, $a, a' \in \{1, \dots, A\}$.
- k: index of pose, $k \in \{1, \dots, K\}$.

1.2 Notation used in this document

Three shortened notations are used across the document:

- $\sum_x$ refers to $\sum_{x=1}^{X}$, with x some index defined in the last section.
- $\sum_{x,y,\dots}$ refers to $\sum_x \sum_y \cdots$.
- δ_a^b refers to Kronecker delta function $\delta(a = b)$.

2 General model

Energy function:

$$E = E_{\text{activity}} + E_{\text{action}} + E_{\text{pose}} + E_{\text{action transition}} + E_{\text{pose transition}}. \tag{1}$$

At the lowest level of the hierarchy, $Z^\top = (z_{1,1} \dots z_{T,R})$ assigns poses to entries in the dictionary. In particular $z_{t,r} = k$ assigns pose (dictionary word) k to region r in frame t.

$$E_{\text{pose}} = \sum_{r,t} {w^r_{z_{t,r}}}^\top x_{t,r} = \sum_{r,t,k} {w^r_{k,r}}^\top x_{t,r} \delta^k_{z_{t,r}} \tag{2}$$

At second level, we assume that we have action labels for each frame and region. $h^{a,r}(Z,V)$ is the histogram over the pose dictionary, at those frames assigned to action a in region r, with $V^\top = (v_{1,1} \dots v_{T,R})$ the vector of action labels for spatial regions and frames. Each entry k in $h^{a,r}(Z,V)$ is given by:

$$h^{a,r}_k(Z,V) = \sum_t \delta^k_{z_{t,r}} \delta^a_{v_{t,r}} \tag{3}$$

$$E_{\text{action}} = \sum_{r,a} {\beta^r_{a,r}}^\top h^{a,r}(Z,V) = \sum_{r,a,t,k} \beta^r_{a,k} \delta^k_{z_{t,r}} \delta^a_{v_{t,r}} \tag{4}$$

At third level, $h^r(V)$ is the histogram corresponding to region r over the action vocabulary accumulated over all frames. Each entry a in $h^r(V)$ is given by:

$$h^r_a(V) = \sum_t \delta^a_{v_{t,r}} \tag{5}$$

$$E_{\text{activity}} = \sum_r {\alpha^r_y}^\top h^r(V) = \sum_{r,a,t} \alpha^r_{y,a} \delta^a_{v_{t,r}} \tag{6}$$

In terms of action and pose transition, energies are defined as histograms over two consecutive frames, involving actions (V) and poses (Z):

$$E_{\text{action transition}} = \sum_{r,a,a'} \gamma^r_{a',a} \sum_t \delta^{a'}_{v_{t-1,r}} \delta^a_{v_{t,r}} \tag{7}$$

$$E_{\text{pose transition}} = \sum_{r,k,k'} \eta^r_{k',k} \sum_t \delta^{k'}_{z_{t-1,r}} \delta^k_{z_{t,r}} \tag{8}$$

3 Algorithm

3.1 Learning

The input to the training algorithm is a set of features X_i from M video sequences along with annotations at the activity y_i and action V_i levels, and the goal is to find the best parameters α, β, w, γ and η that yield to the best classification of the training set of M videos. We can formulate our model using the following objective function:

$$\min_{\alpha,\beta,w,\gamma,\eta} \Omega(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{w},\boldsymbol{\gamma},\boldsymbol{\eta}) + \frac{C}{M} \sum_{i=1}^{M} \max_{Z,V,y} \left(E(X_i,Z,V,y) + \Delta((y_i,V_i),(y,V)) - \max_{Z_i} E(X_i,Z_i,V_i,y_i) \right), \tag{9}$$

where $\Omega(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{w},\boldsymbol{\gamma},\boldsymbol{\eta})$ is a regularization term over the coefficients of the classifiers, visual dictionary and action/pose transition terms. A loss function that favors predicting the correct labels at activity and actions labels is given by:

$$\Delta((y_i,V_i),(y,V)) = \lambda_1 \delta(y_i \neq y) + \frac{\lambda_2}{|V|} \sum_{r,t} \delta(v_{(t,r)_i} \neq v_{t,r}). \tag{10}$$

By selecting a relatively large λ_1, we give a large penalty when the activity is not predicted correctly. The second term adds a penalty proportional to the number of frames that are not labeled with the correct action according to V_i. We can cast the problem in equation (12) as a Latent Structural SVM formulation; the problem can be solved using Concave-Convex Procedure (CCCP), alternating between maximizing equation (12) with respect to latent variables and afterwards, solving a structural SVM optimization problem treating latent variables as completely observed.

$$Z_i^* = \max_{Z_i} E(X_i, Z_i, V_i, y_i) \tag{11}$$

$$\begin{aligned} &\min_{\alpha,\beta,w,\gamma,\eta,\xi} \Omega(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{w}, \boldsymbol{\gamma}, \boldsymbol{\eta}) + \frac{C}{M}\sum_{i=1}^{M} \xi_i \\ &\text{subject to:} \\ &\qquad E(X_i, Z_i, V_i, y_i) - E(X_i, Z, V, y) \geq \Delta((y_i, V_i), (y, V)) - \xi_i, \ \forall y \in \mathcal{Y}, Z \in \mathcal{Z}, V \in \mathcal{V} \end{aligned} \tag{12}$$

Step 1 Infer latent variables $Z|\alpha, \beta, w, \gamma, \eta$

$$\begin{aligned} Z_i^* &= \max_{Z_i} E(X_i, Z_i, V_i, y_i) && (13) \\ &= \max_{Z_i} \left(\sum_{r,a,t,k} \beta^r_{a,k} \delta^k_{z_{t,r}} \delta^a_{v_{t,r}} + \sum_{r,t,k} {w^r_{k,r}}^\top x_{t,r} \delta^k_{z_{t,r}} + \sum_{r,k,k'} \eta^r_{k',k} \sum_t \delta^{k'}_{z_{t-1,r}} \delta^k_{z_{t,r}} \right) && (14) \\ &= \max_{Z_i} \left(\sum_{r,t,k} \left(\beta^r_{v_{t,r},k} + {w^r_{k,r}}^\top x_{t,r} + \sum_{k'} \eta^r_{k',k} \delta^{k'}_{z_{t-1,r}} \right) \delta^k_{z_{t,r}} \right) && (15) \end{aligned}$$

We can solve Z_i^* separately for each region (given V_i) using dynamic programming. Ommiting the sequence index i, for each region r we find Z_r^* solving:

$$Z_r^* = \operatorname*{arg\,max}_{Z=(z_1,\dots,z_T)} \sum_t \beta^r_{v_{t,r},z_t} + {w^r_{z_t}}^\top x_{t,r} + \eta^r_{z_{t-1},z_t} \tag{16}$$

We split the sum in equation (16) in terms of pair-wise relations, and optimize it using a backward recursion

$$f_t(z_t) = \max_{z_{t+1} \in \{1,\dots,K\}} g(z_t, z_{t+1}) + f_{t+1}(z_{t+1}) \tag{17}$$

for all $z_t = 1, \dots, K$, where $g(z_t, z_{t+1})$ defines the *benefit* of choosing the visual word z_t given the visual word in the next frame z_{t+1}, and is given by

$$g(z_t, z_{t+1}) = \beta^r_{v_{t+1,r},z_{t+1}} + {w^r_{z_{t+1}}}^\top x_{t+1,r} + \eta^r_{z_t,z_{t+1}}. \tag{18}$$

Using dynamic programming, we find the optimal sequence Z_r^* for each region in $\mathcal{O}(TK^2)$ operations, whereas the greedy approach is $\mathcal{O}(TK)$.

Step 2 Get $\alpha, \beta, w, \gamma, \eta|Z$

Solve the problem in Equation 12:

$$\begin{aligned} &\min_{\alpha,\beta,w,\gamma,\eta,\xi} \Omega(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{w}, \boldsymbol{\gamma}, \boldsymbol{\eta}) + \frac{C}{M}\sum_{i=1}^{M} \xi_i \\ &\text{subject to:} \\ &\qquad E(X_i, Z_i^*, V_i, y_i) - E(X_i, Z, V, y) \geq \Delta((y_i, V_i), (y, V)) - \xi_i, \ \forall y \in \mathcal{Y}, Z \in \mathcal{Z}, V \in \mathcal{V} \end{aligned} \tag{19}$$

We can solve the problem in Equation (19) using a coordinate descend approach, solving a reduced problem for each parameter, maitaining the constraints of Equation (19). We start by formulating the 1-*slack* version [5] of problem (19):

$$\begin{aligned}
&\min_{\alpha,\beta,w,\gamma,\eta,\xi} \Omega(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{w},\boldsymbol{\gamma},\boldsymbol{\eta}) + C\xi \\
&\text{subject to:} \\
&\frac{1}{M}\sum_{i=1}^{M} E(X_i, Z_i^*, V_i, y_i) - E(X_i, Z, V, y) \geq \frac{1}{M}\sum_{i=1}^{M} \Delta((y_i, V_i),(y,V)) - \xi,\ \forall y \in \mathcal{Y}, Z \in \mathcal{Z}, V \in \mathcal{V}
\end{aligned} \tag{20}$$

which can be solved using the iterative cutting-plane algorithm [5], finding in each iteration the most violated constraint given the solution for the parameters from the previous iteration. Using the property of "loss-augmented inference" of structural SVM [4], finding the most violated constraint for an instance i is given by

$$(\hat{y}, \hat{V}, \hat{Z}) = \arg\max_{y,V,Z} \quad \Delta((y_i, V_i),(y,V)) + E(X_i, Z, V, y) \tag{21}$$

$$\begin{aligned}
= \arg\max_{y,V,Z} \quad & \lambda_1 \delta(y \neq y_i) + \frac{\lambda_2}{RT}\sum_{r,t}\delta(v_{t,r} \neq v_{(t,r)_i}) \\
& + \sum_{r,a,t} \alpha^r_{y,a}\delta^a_{v_{t,r}} + \sum_{r,a,t,k} \beta^r_{a,k}\delta^k_{z_{t,r}}\delta^a_{v_{t,r}} + \sum_{r,t,k} {w^r_k}^\top x_{t,r}\delta^k_{z_{t,r}} \\
& + \sum_{r,t,a,a'} \gamma^r_{a',a}\delta^{a'}_{v_{t-1,r}}\delta^a_{v_{t,r}} + \sum_{r,t,k,k'} \eta^r_{k',k}\delta^{k'}_{z_{t-1,r}}\delta^k_{z_{t,r}}
\end{aligned} \tag{22}$$

$$\begin{aligned}
= \arg\max_{y,V,Z} \quad & \lambda_1 \delta(y \neq y_i) + \sum_{r,t}\left(\frac{\lambda_2}{RT}\delta(v_{t,r} \neq v_{(t,r)_i}) + \alpha^r_{y,v_{t,r}} + \beta^r_{v_{t,r},z_{t,r}} + {w^r_{z_{t,r}}}^\top x_{t,r}\right. \\
& \left. + \gamma^r_{v_{t-1,r},v_{t,r}} + \eta^r_{z_{t-1,r},z_{t,r}}\right)
\end{aligned} \tag{23}$$

It is important to note that the regularizer $\Omega(\cdot)$ does not play any role in solving equation (21). We can solve (21) by exhaustively enumerating all values of y, and solving the following for every instance i for a query class y:

$$\begin{aligned}
\hat{V}_y, \hat{Z}_y = \arg\max_{V,Z} \quad & \sum_{r,t}\left(\frac{\lambda_2}{RT}\delta(v_{t,r} \neq v_{(t,r)_i}) + \alpha^r_{y,v_{t,r}} + \beta^r_{v_{t,r},z_{t,r}} + {w^r_{z_{t,r}}}^\top x_{t,r}\right. \\
& \left. + \gamma^r_{v_{t-1,r},v_{t,r}} + \eta^r_{z_{t-1,r},z_{t,r}}\right).
\end{aligned} \tag{24}$$

We can solve Equation (24) in every region using dynamic programming. We split the sum in Equation (24) in terms of pair-wise relations, and optimize it using a backward recursion

$$F_t(v_t, z_t) = \max_{\substack{v_{t+1} \in \{1,\dots,A\} \\ z_{t+1} \in \{1,\dots,K\}}} G((v_t, z_t),(v_{t+1}, z_{t+1})) + F_{t+1}(v_{t+1}, z_{t+1}) \tag{25}$$

for all $v_t = 1, \dots, A$ and $z_t = 1, \dots, K$, where G defines the *benefit* of choosing the action v_t and visual word z_t given the action and visual word in the next frame $t+1$. For each region r, G is given by

$$G^r((v_t, z_t),(v_{t+1}, z_{t+1})) = G^r_s((v_t, z_t),(v_{t+1}, z_{t+1})) + G^r_n(v_{t+1}, z_{t+1}) \tag{26}$$

$$\begin{aligned}
G^r_s((v_t, z_t),(v_{t+1}, z_{t+1})) &= \gamma^r_{v_{t,r},v_{t+1,r}} + \eta^r_{z_{t,r},z_{t+1,r}} \\
G^r_n(v_{t+1}, z_{t+1}) &= \frac{\lambda_2}{RT}\delta(v_{t+1,r} \neq v_{(t+1,r)_i}) + \alpha^r_{y,v_{t+1,r}} + \beta^r_{v_{t+1,r},z_{t+1,r}} + {w^r_{z_{t+1,r}}}^\top x_{t+1,r}
\end{aligned} \tag{27}$$

Using dynamic programming, we find the optimal sequence $\hat{V}_y$ and $\hat{Z}_y$ for each region in $\mathcal{O}(TA^2K^2)$ operations, whereas the greedy approach is $\mathcal{O}(TAK)$. The use of dynamic programming instead of a greedy approach is based on the fact that we must find the most violated constraint (and not a approximation) for every cutting plane step; otherwise, we have an early convergence to a suboptimal weight vector that can't reproduce the correct action labeling: while in learning we get all constraints satisfied (because V_i is given), some sequences are wrongly classified when training data is tested. We split G into a sequential term G_s and a non-sequential term G_n. It will be important in the implementation section of this document.

3.2 Inference

The input to the inference algorithm is a novel video sequence with features X. The task is to infer the best activity label y^* and the best action labeling V^*. However, we also need to estimate the best assignment of features to dictionary entries.

$$y^*, V^*, Z^* = \arg\max_{y,V,Z} E(X,Z,V,y) \tag{28}$$

We can solve this by exhaustively enumerating all values of y, and solving the following at each step:

$$V_y^*, Z_y^* = \arg\max_{V,Z} E(X,Z,V,y) \tag{29}$$

The last expression is the same in Equation (24) setting $\lambda_1 = \lambda_2 = 0$. For each y, we must find V_y^* and Z_y^*:

$$\begin{aligned} V_y^*, Z_y^* = \arg\max_{V,Z} \quad \sum_{r,t} \Big(& \alpha^r_{y,v_{t,r}} + \beta^r_{v_{t,r},r,z_{t,r}} + {w^r_{z_{t,r}}}^\top x_{t,r} \\ & + \gamma^r_{v_{t-1,r},v_{t,r}} + \eta^r_{z_{t-1,r},z_{t,r}} \Big) . \end{aligned} \tag{30}$$

3.3 Regularizers

The regularizer $\Omega(\cdot)$ plays the role of generating classifiers with certain desired properties. For instance, using a quadratic regularizer

$$\Omega(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{w}, \gamma, \boldsymbol{\eta}) = \frac{1}{2}||\boldsymbol{\alpha}||_2^2 + \frac{1}{2}||\boldsymbol{\beta}||_2^2 + \frac{1}{2}||\boldsymbol{w}||_2^2 + \frac{1}{2}||\gamma||_2^2 + \frac{1}{2}||\boldsymbol{\eta}||_2^2 \tag{31}$$

will tend to produce classifiers with small weights, oriented to reduce over-fitting. In sparse representation literature, it is common to orient the coefficients of the classifiers to have a sparse representation, using the benefits of L_1 norm as in Lasso regularizer. In our model, an interesting sparsification could be applied to $\boldsymbol{\beta}$, the classifiers of atomic actions that uses histograms of poses, with the aim that not every pose have influence in every action. Also, if we use the constraint $\beta \geq 0$ we encourage the classifiers to use only "present" poses. The overall goal is to have few shared poses in every action. The regularizer using sparsity on $\boldsymbol{\beta}$ could be stated as

$$\Omega(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{w}, \gamma, \boldsymbol{\eta}) = \frac{1}{2}||\boldsymbol{\alpha}||_2^2 + \frac{\mu}{2}||\boldsymbol{\beta}||_2^2 + \rho||\boldsymbol{\beta}||_1 + \frac{1}{2}||\boldsymbol{w}||_2^2 + \frac{1}{2}||\boldsymbol{\gamma}||_2^2 + \frac{1}{2}||\boldsymbol{\eta}||_2^2 \tag{32}$$

We include both L_1 and L_2 regularizers on β, wighted by a parameter ρ. In practice, the inclusion of the L_2 norm makes the objective function of the dual differentiable (we will see this in the implementation section).

4 Implementation

We will give implementation details for the algorithm. There is two phases that consume most of the computing time: the dynamic program (DP) to find the most violated constraint, and the optimization given the working set of constraints.

4.1 Fast searching of most violated constraint

The database Composable Activities has 16 complex activities, 26 actions, and 700 sequences of 400 frames in average, and every frame is divided into 4 regions. If we use dynamic programming "as is" for finding the most violated constraint, we will have to compute about 3×10^{12} scores (assuming that frames are sub-sampled at 8X). This is a much lower number of calculations than using exhaustive search, but it is still very high, and consumes several CPU minutes. As our algorithm is iterative, for 5000-10000 cutting-plane iterations of 5 minutes each we will wait for several days only for an outer iteration. In a previous implementation, we proposed to use only the actions that actually appear in the each activity (recall that we mush find the most violated constraint by searching all the activities), using the fact that in average 4 actions are used per activity, resulting in a 40X decrease in computing time. The complete DP was used only for some iterations. In practice, this method makes the associated α of the not-considered actions to be zero. In our new implementation we use non-negativity constraint in α to mimic this behavior, producing that every activity will weight only actions composing the activity, but not punishing others.

In our current implementation, we no longer force using only the actions "permitted" by each activity, because of the erratic behavior when the activity is not defined, so we could not test the model for zero-shot learning of new activities. Instead, we use the non-sequential terms G_s computed previously to the DP as indicators for reducing the number of (a, k) combinations computed for each frame. We realized that for almost 80% of the frames, the action a and pose k of the most violated constraint in frame t is between the first five highest scores of G_s, and 99.98% for the 150 top scores of G_s. With this in mind, we first use DP with only P top score combinations for every frame (20, 50 and finally 150). The order of DP is proportional to $(AK)^2$, and with this method is reduced to be proportional to P^2. In practice it behaves quite good, since for most cases the iterations for $P = 50$ and $P = 150$ are very few.

4.2 Optimization

In our previous CVPR paper, we use the quadratic regularizer, so the optimization is very fast using the implemented SMO code of SVM^{struct} of T. Joachims. For the new regularizer using a mix of L_2 and L_1 norms, it is desirable to have a similar execution time, as the number of cutting plane iterations for the complete algorithm is relatively high (about 10^6 iterations).

4.2.1 Primal formulation

One first alternative is to solve the primal of (20) using an off-the-shelf optimization code. As we use 1-*slack* formulation, it is easy to incorporate ξ to the optimization as a new variable, using an augmented weight vector

$$\begin{aligned} \boldsymbol{W}_{aug}^\top &= [\boldsymbol{\alpha}^\top \ \boldsymbol{\beta}^\top \ \boldsymbol{w}^\top \ \boldsymbol{\gamma}^\top \ \boldsymbol{\eta}^\top \ \xi] = [\boldsymbol{W} \ \xi] \\ \boldsymbol{W}_{-\beta} &= [\boldsymbol{\alpha}^\top \ \boldsymbol{w}^\top \ \boldsymbol{\gamma}^\top \ \boldsymbol{\eta}^\top], \end{aligned} \tag{33}$$

so the formulation of the primal is

$$\begin{aligned} &\min_{\boldsymbol{W}_{-\beta},\beta,\xi} \textit{obj-primal}(\boldsymbol{W}_{-\beta}, \beta, \xi, \rho) = \frac{1}{2}||\boldsymbol{W}_{-\beta}||_2^2 + \frac{\mu}{2}||\boldsymbol{\beta}||_2^2 + \rho||\boldsymbol{\beta}||_1 + C\xi \\ &\text{subject to:} \end{aligned} \tag{34}$$

$$\begin{aligned} -\boldsymbol{W}_{aug}^\top \begin{bmatrix} \bar{\Psi}_i(y, V, Z) \\ \xi \end{bmatrix} + \bar{\Delta}_i(y, V) \le 0, &\quad \forall y \in \mathcal{Y}, Z \in \mathcal{Z}, V \in \mathcal{V} \\ -\beta_{r,a,k} \le 0 & \\ -\alpha_{r,c,a} \le 0 & \\ -\xi \le 0 & \end{aligned}$$

where

$$\begin{aligned}\bar{\Psi}_i(y,V,Z) &= \frac{1}{M}\sum_{i=1}^{M} E(X_i,Z_i^*,V_i,y_i) - E(X_i,Z,V,y)\\ \bar{\Delta}_i(y,V) &= \frac{1}{M}\sum_{i=1}^{M} \Delta((y_i,V_i),(y,V))\end{aligned} \tag{35}$$

If we check the gradient of the objective function, we have

$$\frac{\partial \textit{ obj-primal}(\boldsymbol{W}_{-\beta},\beta,\xi,\rho)}{\partial W_i} = \begin{cases} W_i & \text{if } W_i \in \{\alpha,w,\gamma,\eta\}\\ \mu W_i + \rho\frac{W_i}{|W_i|} & \text{if } W_i \in \beta \end{cases} \tag{36}$$

Note that the gradient is not differentiable for $\beta_i = 0$. As we use the condition $\beta_i \geq 0$, it is not a problem since we have no longer an undefined gradient.

The problem in (34) can be solved by a generic nonlinear optimization package that supports linear constraints and bound constraints. We tested using NLOPT, using the Augmented Lagrangian method provided by the library, and using a LBFGS method for solving the unconstrained problem. The code produces good results in general, but the speed of the optimization slows down every time a new constraint is added; for a few constraints (< 100) is not a problem, but for a high number of constraints the execution time of the optimizer was problematic.

4.2.2 Dual formulation

In SVM, an interesting property of the dual formulation is that the number of dimensions of the problem is equal to the number of constraints, and involves solving a QP with a single linear constraint (in 1-*slack* formulation) and bound constraints. The same principle could be applied for our regularizer. We will see that including a mix of L_2 and L_1 norms, compared to use only L_1, carries at least two benefits: first, the objective function of the dual is differentiable (that is, has a continuous gradient), and also the regularization term appears in the objective function, and not as a set of non-linear constraints as usual when only L_1 norm is used. This nice properties allow us to use a general purpose optimization library in solving the dual problem.

Recalling from convex optimization theory, the dual formulation for the problem

$$\begin{aligned}&\min_{x\in \mathrm{dom} f} f(x)\\ &\text{s. to} \quad Qx - b \leq 0\end{aligned} \tag{37}$$

is given by

$$\begin{aligned}&\max_{\phi\in \mathrm{dom} f^*} -f^*(-Q^\top\phi) - b^\top\phi\\ &\text{s. to} \quad \phi \geq 0\end{aligned} \tag{38}$$

where f^* is the *convex conjugate* function of f, given by $f^*(y) = \sup_{x\in \mathrm{dom} f}\{y^\top x - f(x)\}$. The convex conjugate function is well defined only when it is bounded.

The convex conjugate functions can be added for independent variables, so we are able to treat separately $\{\alpha,w,\gamma,\eta\}$, $\{\beta\}$ and $\{\xi\}$. Also, non-negativity constraints of α, β and ξ can be arsorbed into the linear constraints $Qx - b \leq 0$; as we are using 1-*slack* formulation, ξ can be incorporated in $\boldsymbol{W}$.

Convex conjugate function for $\frac{1}{2}||x||_2^2$

$$f(x) = \frac{1}{2}||x||_2^2 \Rightarrow f^*(y) = \sup_{x\in \mathrm{dom} f}\{y^\top x - \frac{1}{2}||x||_2^2\} \Rightarrow x* = y \Rightarrow f^*(y) = \frac{1}{2}||y||_2^2 \tag{39}$$

Convex conjugate function for $\frac{\mu}{2}||x||_2^2 + \rho||x||_1$

$$f(x) = \frac{\mu}{2}||x||_2^2 + \rho||x||_1 \Rightarrow f^*(y) = \sup_{x \in \text{dom} f} \{y^\top x - \frac{\mu}{2}||x||_2^2 - \rho||x||_1\} \tag{40}$$

As we are adding independent variables, we can formulate the convex conjugate function as

$$\begin{aligned} f^*(y) &= \sum_{i=1}^{M} f_i^*(y_i) \\ &= \sum_{i=1}^{M} \sup_{x_i \in \text{dom} f_i} \{y_i x_i - \mu x_i^2 - \rho|x_i|\} \end{aligned} \tag{41}$$

With some math, we have

$$f_i^*(y_i) = \begin{cases} 0 & \text{for } |y_i| < \rho \\ \frac{1}{2\mu}(|y_i| - \rho)^2 & \text{for } |y_i| < \rho,\, 0 \le \rho < 1 \end{cases} \tag{42}$$

$$f^*(y) = \frac{1}{2\mu} \sum_{i=1}^{M} [\max(0,\ |y_i| - \rho)]^2 \tag{43}$$

Convex conjugate function for Cx

$$f(x) = Cx \Rightarrow f^*(y) = 0 \text{ only for } y = C \tag{44}$$

Now, we formulate the constraints if a matrix form $Qx - b \le 0$. The constraints of the primal are

$$\begin{aligned} -\boldsymbol{W}_{aug}^\top \begin{bmatrix} \bar{\Psi}_i(y, V, Z) \\ \xi \end{bmatrix} + \bar{\Delta}_i(y, V) \le 0, \quad \forall y \in \mathcal{Y}, Z \in \mathcal{Z}, V \in \mathcal{V} \\ -\beta_{r,a,k} \le 0 \\ -\alpha_{r,c,a} \le 0 \\ -\xi \le 0 \end{aligned} \tag{45}$$

that we can formulate as a set of linear constraints $-\tilde{\boldsymbol{Q}}\boldsymbol{W}_{aug} - \tilde{\boldsymbol{b}} \le \boldsymbol{0}$:

$$-\tilde{\boldsymbol{Q}} = \begin{bmatrix} -A & -B & -W & -G & -N & -X \\ -I_A & 0 & 0 & 0 & 0 & 0 \\ 0 & -I_B & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} = \begin{bmatrix} -\tilde{A} & -\tilde{B} & -\tilde{W} & -\tilde{G} & -\tilde{N} & -\tilde{X} \end{bmatrix} \qquad \tilde{\boldsymbol{b}} = \begin{bmatrix} -\bar{\Delta}_i(y, v) \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{46}$$

The dual problem can now be stated as (considering ϕ as the dual variables)

$$\begin{aligned} &\min_{\phi} f^*(\tilde{\boldsymbol{Q}}^\top \phi) + \tilde{\boldsymbol{b}}^\top \phi \\ &\text{subject to} \\ &\phi_i \ge 0 \end{aligned} \tag{47}$$

Using the results for $f^*(y)$, the dual problem became

$$\begin{aligned} &\min_{\phi} g(\phi) = \frac{1}{2}||\tilde{A}^\top \phi||_2^2 + \frac{1}{2}||\tilde{W}^\top \phi||_2^2 + \frac{1}{2}||\tilde{G}^\top \phi||_2^2 + \frac{1}{2}||\tilde{N}^\top \phi||_2^2 + \frac{1}{2\mu} \sum_{r,a,k} [\max(0, |\tilde{B}_{r,a,k}^\top \phi| - \rho)]^2 + \tilde{b}^\top \phi \\ &\text{subject to} \\ &\sum_{i=1}^{M} \phi_{c_i} \le C, \quad \phi_{c_i} \ge 0, \quad \phi_{A_j} \ge 0 \text{ for } j = \{1, \ldots, \dim(\boldsymbol{\phi})\}, \quad \phi_{B_j} \ge 0 \text{ for } j = \{1, \ldots, \dim(\boldsymbol{\beta})\} \end{aligned} \tag{48}$$

Where ϕ_c refers to the dual variables associated to the original linear constraints, while ϕ_A and ϕ_B with the non-negativity constraint of $\boldsymbol{\phi}$ and $\boldsymbol{\beta}$. The linear constraint $\sum_{i=0}^{M} \phi_{c_i} \leq C$ is produced by the linear term in the primal formulation, $C\xi$. Recall that $f^*(y) = 0$ only for $y = C$; in this case, $y = -\tilde{X}^\top \phi = \sum_{i=1}^{M} \phi_{c_i} + \phi_\xi = C$. As ϕ_ξ must be always greater or equal to zero, it is enough to consider the constraint $\sum_{i=0}^{M} \phi_{c_i} \leq C$ and eliminate ϕ_ξ of the problem. The gradient of the objective function of the dual formulation is given by

$$\frac{\partial g(\phi)}{\partial \phi} = \tilde{U}\tilde{U}^\top \phi + \frac{1}{\mu} \sum_{r,a,k} \max(0, |\tilde{B}_{r,a,k}^\top \phi| - \rho) \tilde{B}_{r,a,k} \tilde{B}_{r,a,k}^\top \phi + \tilde{b} \tag{49}$$

where

$$\tilde{U} = \begin{bmatrix} \tilde{A} & \tilde{W} & \tilde{G} & \tilde{N} \end{bmatrix} \tag{50}$$

In the equations, (r, a, k) acts as the column index of $\tilde{B}$.

It is worth to note that for $\rho = 0$ and $\mu = 1$, we recover the original quadratic formulation. For $\mu = 0$, the formulation is still useful, since the terms associated with $\tilde{B}$ disappear from the objective function, but it adds $R \times A \times K$ non-linear constraints $|\tilde{B}_{r,a,k}^\top \phi| \leq \rho$. Using only L_1 therefore is not the best idea since it force us to use 5000 non-linear constraints for $k = 50$ in Composed Activities dataset. The gradient of the objective function is continuous,

In the actual implementation, only the dual variables associated to constraints are computed, as ϕ_A and ϕ_B can be inferred as they try to make the associated original weight to zero, so $\phi_A = \max(0, -A^\top \phi_c)$, $\phi_B = \max(0, -B^\top \phi_c)$.

To recover the primal variables from dual variables, we use the solutions for x^* for $f^*(y)$. For quadratic terms (L_2 norm), we have $x^* = y = -Q^\top \phi$, so we have $\boldsymbol{\alpha} = \tilde{A}^\top \phi$, $\boldsymbol{w} = \tilde{W}^\top \phi$, $\boldsymbol{\gamma} = \tilde{G}^\top \phi$ and $\boldsymbol{\eta} = \tilde{N}^\top \phi$. For $\boldsymbol{\beta}$, we have $y - \mu x^* - \rho \frac{x^*}{|x^*|} = 0$ for $|y| > \rho$, and $x^* = 0$ otherwise. Noting that x^* and y must have the same sign, we can solve for $\boldsymbol{\beta}$ as

$$\boldsymbol{\beta}_i = \begin{cases} 0 & \text{if } |\tilde{B}_i^\top \phi| \leq \rho \\ \frac{\tilde{B}_i^\top \phi}{\mu[1 + \frac{\rho}{|\tilde{B}_i^\top \phi| - \rho}]} & \text{if } |\tilde{B}_i^\top \phi| > \rho \end{cases} \tag{51}$$

4.3 Implementation details

5 Improving the algorithm

We have the following ideas to improve the algorithm:

1. Treat V as latent variables, allowing to use more general databases where only high level annotations are available, and also for semi-supervised learning.

2. Increase the discriminativity in the lower level of the hierarchy: use *few* poses per atomic action, trying to share the same poses for an atomic action across the activities. In the current model, all poses can be used by every action, lowering the discriminative power of the pose classifiers .

3. Improve the representation of local features. Try to get a less noisy and more informative descriptor related to several local descriptors instead of a single descriptor, i. e. related to a subsequence in videos.

4. Create a *general* framework for (semi)supervised hierarchical models: local features in the lower level (local parts in objects, poses in activities), use annotated data in the highest level (scene, activity), and incorporate mid-level features as latent variables (objects in a scene, atomic actions for activities). Also, create a general treatment of the spatial pooling and the relations between the regions (spatial relations).

5. Include in the model a more efficient way to treat the local features. The original model uses all frames in the energy function, even if some poses are noisy or non-informative.

6. Let the model estimate an appropriate number of low level representations infer the number of parts/poses.

5.1 Promote more discriminative pose classifiers

In our original model, every frame *must* be labeled with one out of K poses. During learning, this means that every frame influences the pose classifiers according to its pose label z. As our model is based in average pooling, this means that the pose classifiers are trained over averaged pose descriptors; moreover, when using 1-*slack* formulation (to use cutting-plane algorithm), we use the averaged descriptor of all frames with the same label of the complete training dataset! We think that a better strategy would be training pose classifiers using only a subset of frames, enforcing to use a relatively small number of labeled frames per video. The proposed change in the model is to include a new pose label $(K+1)$, learn a weight θ that behaves as a pose score for all *unlabeled* frames (i.e. with label $K+1$), and change the loss function in Eq. (10) to add a penalty if the number of labeled frames per action is above a threshold λ_4. The new energy equations are:

$$E_{\text{pose}} = \sum_{r,t}\left[\sum_{k=1}^{K} w_k^{r\top} x_{t,r}\delta(z_{t,r}=k) + \theta\delta(z_{t,r}=K+1)\right] \tag{52}$$

$$E_{\text{action}} = \sum_{r,a,t}\sum_{k=1}^{K+1}\beta_{a,k}^{r}\delta(z_{t,r}=k)\delta(v_{t,r}=a) \tag{53}$$

$$E_{\text{activity}} = \sum_{r,a,t}\alpha_{y,a}^{r}\delta(v_{t,r}=a) \tag{54}$$

$$E_{\text{action transition}} = \sum_{r,a,a'}\gamma_{a',a}^{r}\sum_{t}\delta_{v_{t-1,r}}^{a'}\delta_{v_{t,r}}^{a} \tag{55}$$

$$E_{\text{pose transition}} = \sum_{r}\sum_{k=1}^{K+1}\sum_{k'=1}^{K+1}\eta_{k',k}^{r}\sum_{t}\delta_{z_{t-1,r}}^{k'}\delta_{z_{t,r}}^{k} \tag{56}$$

Defining $J(r,a) = \sum_{t=1}^{T}\sum_{k=1}^{K}\delta(v_{t,r}=a)\delta(z_{t,r}=k)$ as the number of labeled frames of action a in region r, the new loss function is written as:

$$\Delta((y_i,V_i),(y,V,Z)) = \lambda_1\delta(y_i\neq y) + \frac{\lambda_2}{T}\sum_{r,t}\delta(v_{(t,r)_i}\neq v_{t,r}) + \frac{\lambda_3}{T}\sum_{r,a}l(r,a) \tag{57}$$

where

$$l(r,a) = \left(\max\{0, J(r,a)-\lambda_4\}\right)^2 \tag{58}$$

It is important to note that in Eq. (77) we are not using the imputed latent variables Z_i, so we can keep using the structural SVM framework for solving our model. The intuition behind the new model is to add a *benefit* when a frame is not labeled (or, more precisely, labeled as $K+1$). Other alternative is just to omit certain frames; in this case, we could have balance issues since some videos could have much more labeled frames than others. This is not the case in the presented model, as we are not omitting frames; instead, we are treating those frames labeled as $K+1$ as non-informative frames. Every non-informative frame adds a score θ to the energy function, and the number of unlabeled frames are used in the in the action classifiers β as a new bin in the histogram of poses relative to actions $h^{a,r}(Z,V)$ (Eq. (3)). With this setup, as the model must keep its discriminative power, the best poses will be selected to learn the pose classifiers, leaving the rest of unlabeled frames not feeding any pose classifier but participating in the energy score.

The equations for inferring the labels Z_i^* and finding the arguments of the most violated constraint $(\hat{y}\hat{V},\hat{Z})$ are very similar, so the complexity of the model is kept.

5.2 Algorithm treating V as latent variables

In the previous algorithm we assumed atomic action labels V for every frame. Now we propose to treat the action labels V as latent variables. Every sequence is assumed weakly labeled with the activity label.

The first approach for using V as latent is to know in advance the number of atomic actions (in future models we could estimate the number of atomic actions). In this sense, we have two issues to solve:

- We are no longer *allowed* to use the "real" value of V (V_i in the original formulation) in the loss funtion $\Delta(y_i, V_i, y, V)$.
- As we don't have annotated data for V, we have to guess initial values for the latent variables V.

The first issue could be solved by regularizing in the number of actions present in every sequence. We aim to have a low number of actions in every sequence, and also that the same actions appear in the same activity. We could model this behavior regularizing α using l_1/l_2 norm, forcing to use few actions in every activity, allowing sharing actions across activities. The second issue is more problematic. One possible way to initialize V could be to use simple pre-trained action classifiers, and use a sliding window scheme to get initial action labels. We also need a way to restrict action transitions of every video, to penalize noisy labeling of actions. We propose a term to get a good behavior, that penalizes action transition between consecutive frames. The new loss function is given by

$$\Delta(y_i, y, V) = \lambda_1 \delta(y_i \neq y) + \frac{\lambda_2}{|V|} \sum_{r,t} \delta(v_{t-1,r} \neq v_{t,r}). \tag{59}$$

The model of Equation (12) is similar but now including the l_1/l_2 regularizer in β. The steps of finding the most violated constraint and infer latent variables also differs. The model states as:

For finding the most violated constraint, we use the "loss-augmented inference" as in Equation (21), using the new loss function:

$$\begin{aligned}
(\hat{y}, \hat{V}, \hat{Z}) = \underset{y,V,Z}{\arg\max} \quad & \Delta(y_i, y, V) + E(X_i, Z, V, y) && (60)\\
= \underset{y,V,Z}{\arg\max} \quad & \lambda_1 \delta(y_i \neq y) + \frac{\lambda_2}{RT} \sum_{r,t} \delta(v_{t-1,r} \neq v_{t,r}) \\
& + \sum_{r,a,t} \alpha_{y,r,a} \delta^a_{v_{t,r}} + \sum_{r,a,t,k} \beta_{a,r,k} \delta^k_{z_{t,r}} \delta^a_{v_{t,r}} + \sum_{r,t,k} w_{k,r}^\top x_{t,r} \delta^k_{z_{t,r}} && (61)\\
& + \sum_{r,t,a,a'} \gamma_{a',a,r} \delta^{a'}_{v_{t-1,r}} \delta^a_{v_{t,r}} + \sum_{r,t,k,k'} \eta_{k',k,r} \delta^{k'}_{z_{t-1,r}} \delta^k_{z_{t,r}} \\
= \underset{y,V,Z}{\arg\max} \quad & \lambda_1 \delta(y_i \neq y) + \sum_{r,t} \Bigg(\frac{\lambda_2}{RT} \delta(v_{t-1,r} \neq v_{t,r}) + \alpha_{y,r,v_{t,r}} + \beta_{v_{t,r},r,z_{t,r}} + w_{z_{t,r}}^\top x_{t,r} && (62)\\
& + \gamma_{v_{t-1,r},v_{t,r},r} + \eta_{z_{t-1,r},z_{t,r},r} \Bigg)
\end{aligned}$$

We can solve this by exhaustively enumerating all values of y, and solving the following for every instance i for a query class y:

$$\begin{aligned}
\hat{V}_y, \hat{Z}_y = \underset{V,Z}{\arg\max} \quad \sum_{r,t} \Bigg(& \frac{\lambda_2}{RT} \delta(v_{t-1,r} \neq v_{t,r}) + \alpha_{y,r,v_{t,r}} + \beta_{v_{t,r},r,z_{t,r}} + w_{z_{t,r}}^\top x_{t,r} \\
& + \gamma_{v_{t-1,r},v_r,r} + \eta_{z_{t-1,r},z_r,r} \Bigg).
\end{aligned} \tag{63}$$

We can solve Equation (63) in every region using dinamic programming. We split the sum in Equation (63) in terms of pair-wise relations, and optimize it using a backward recursion equal to Equation (25), with G given by

$$\begin{aligned}
G((v_t, z_t), (v_{t+1}, z_{t+1})) = & \frac{\lambda_2}{RT} \delta(v_{t,r} \neq v_{t+1,r}) + \alpha_{y,r,v_{t+1,r}} + \beta_{v_{t+1,r},r,z_{t+1,r}} + w_{z_{t+1,r}}^\top x_{t+1,r} \\
& + \gamma_{v_{t,r},v_{t+1,r},r} + \eta_{z_{t,r},z_{t+1,r},r}
\end{aligned} \tag{64}$$

NOTE: It would be better if we know in advance how many actions appear in every sequence, to penalize the difference of the number of actions predicted on the sequence and the real number of actions. The proposed loss function penalizes every action transition, so in principle a vector of the same action for the entire sequence could be the best sequence of actions... I hope these sequences have low energy. If not, some changes should be done in the loss function.

We infer latent variables the same way as Equations (60) to (64) with $\lambda_1 = \lambda_2 = 0$, and using only the correct activity class y_i. zz

5.3 Algorithm for unstructured videos

There is some issues to solve to deal with unconstrained RGB videos using a somewhat modified model, but not ”too” modified. I think the main changes could be:

- Change video description. In particular, change from a frame-based feature (geometry and appearance) to a region-based descriptor that encodes what's happening inside the region, with a good balance between description (trajectories, appearance, locations) and representativeness. I think a good place to start is using Dense Trajectories, excluding the background and low-motion trajectories, and (AS idea) apply a human detector with high recall/low precision to exclude non-human trajectories, keeping a relatively small set of "valid" trajectories for each video. We could even apply a upper-body detector and/or a face detector, since there is some actions/activities that do not involve the whole body. Then, normalize the box dimensions of the appearance descriptors according to the "size" of the cloud of valid trajectories (resembling zoom pooling), and also get a normalized location of the trajectories. I think we could explore a way to include scores for full body / mid body / face / other objects in the videos and then use that info to bias the learning or include the detectors scores augmenting the region descriptor.

- Starting from the region descriptor, the idea is using a similar three-level hierarchical model like our previous model. The initial step is be adding a new fixed lower level of region description (in practice the model is a 4-level hierarchy: trajectories, poses, actions and activities). For every training video, sparse codes of every valid trajectory must be computed, and then the codes must be pooled in each region to create a region descriptor. Note that each coefficient of the representation encodes the shape of the trajectory and appearance (as Wang's paper). I propose to augment the descriptor with the normalized location of the center of the trajectory; then, each coefficient also will encode the normalized location. I propose also to use a "big" dictionary in level 0, since there is a lot of things to encode (shape, appearance and location).

- Treat actions as latent variables. In most of action datasets there is no mid-level actions that compose an activity (at least no annotated data). To deal with the complexity of encoding the activities from poses, I think we must keep the mid-level of actions. When annotations exist, as in Composable Activities, CAD120, or Breakfast dataset, we could use the actions labels to guide the learning stage; but when there is no labels, we still could use our model to infer the actions, at an extra cost of initializing the action labels (even when there is no clue of how to do it).

- Use"garbage collectors' in the pose and action levels to deal with uninformative regions. We could use the pose $K+1$ and action $A+1$ as our previous model, learning weights θ_{pose} and θ_{action} for starting. When there is annotated data of when the activity/action occurs, we can use directly that info during learning; otherwise, i.e. the video is weakly labeled, a first step would be to initialize the non-informative regions regarding to actions.

- Apply sparseness to every classifier (w, β, α), In particular:
  - In w, it would allow to each pose be specialized in a subgroup of low-level primitives.
  - In β, it will produce the actions (mid-level) to be fed with a low number of poses.
  - In α, the sparsity will help to bound the number of actions for each activity.

- Explore a different formulation in transition terms for poses and actions. In our previous model, this was the bottleneck of the computing time, since the complexity was proportional to A^2K^2. We apply approximations to deal with the excessive running time for learning. The main speed-up factor is to precompute AK scores for every frame (non-sequential terms), store the highest P scores for every frame (keeping a list of (a_p, k_p), $p \in \{1, \dots, P\}$), and then use the list for each frame to run the dynamic program, so now the complexity is proportional to P^2. Also, we use multi-threading in finding the most violated constraint and for computing the objective function (and its gradient) during the weight vector update. Now, I think a good idea would be to limit the transitions in terms of "informative" (I) and "non-informative" (NI) poses and actions, and learn only a few weight parameters:

  - Weights for going from $I \rightarrow NI$ and $NI \rightarrow I$.
  - If region is I, weights to keep the pose/action $I_t == I_{t-1}$ and to change the pose/action $I_t \sim= I_{t-1}$.

  So, we could learn only 4 transitions for poses and 4 for actions, making the dynamic program mush easier to solve.

- Add some energy terms related to the activity, for example:
  - A "global appearance" descriptor for the complete video. Some activities are likely to be executed in the same location (gym, road, park, mountain, etc), so there is information not related no movements but the appearance of the background.
  - Add object detectors to the model. In the paper of CAD120, adding object detectors improve the accuracy of action retrieval, specially when including interactions between objects and actions.

In order to foster a well-behaved model, some regularizations/constraints/penalties could be added:

- Minimum duration of actions
- Limit high-frequency transitions. Related to duration, it could be the case of several different actions with minimum duration interleaved in a video, so this is not desirable.
- Add penalties related to the diversity of poses and diversity of actions in the same video.

In the following, each change will be explained.

Video description Our previous model was based on frames and a geometric + appearance descriptor, using mainly Composable Activities dataset, which allows to get good performance with complete semantic interpretation. For more, unstructured and longer videos, a frame-based descriptor will no longer be practical. We propose to include a new level in the hierarchy (level 0), building an aggregated video description, useful for the next levels of the hierarchy.

For each video, we first compute trajectories similar to H. Wang's Dense Trajectories, excluding trajectories from the estimated background and those with little motion. Taking the (x, y) location of the center of the trajectories, compute the parameters of a 2D Gaussian distribution of the trajectory centers in the whole video, to get (μ_x, μ_y, σ_x, σ_y); then, normalize the center location of each trajectory using the 2D Gaussian. The appearance descriptor of every trajectory in the video is computed using a scaled spatial box depending on σ_x and σ_y. Once all descriptors are computed (trajectory shape, HOG, HOG and MBH), the descriptor is augmented with the normalized location of the center of the trajectory. For every video we have N_i trajectories with descriptors f_n, $n \in \{1, \ldots, N_i\}$.

The next step is to compute a "big" dictionary $\boldsymbol{D}_0$ (maybe in the order of tens of thousands of entries), computing in a first step class-specific dictionaries D_o^y , and then concatenating all atoms in a unique dictionary D_0. Once D_0 is computed, every trajectory should be coded into a sparse vector a_n, using an L_1 minimization problem (Lasso) for each video i:

$$\min_{\boldsymbol{D}_0, \boldsymbol{A}_i} ||\boldsymbol{F}_i - \boldsymbol{D}_0 \boldsymbol{A}_i||_2^2 + \lambda_A ||\boldsymbol{A}_i||_1, \;\; \boldsymbol{F}_i = [f_1 \quad \ldots \quad f_{N_i}], \;\; \boldsymbol{A}_i = [a_1 \quad \ldots \quad a_{N_i}]. \tag{65}$$

Other coding scheme (proposed by AS), that can be used complimentary to the creation of the dictionary D_0, is an extension of a R. Vidal paper [REF to be included] of coding a video with a sparse reconstruction using descriptors of the same video as atoms of the dictionary, and applying a sort of group sparsity in the reconstruction coefficients, taking the rows of the reconstruction coefficients as the groups, to select distinctive descriptors of each video. AS propose to apply the same philosophy but using all videos of the class as the target to be reconstructed, selecting descriptors for each video that are informative for all videos of the class. Defining F_y as the matrix containing all descriptors (or a "big" representative sample) of class y, the proposed coding for each video is stated as

$$\min_{\boldsymbol{D}_0, \boldsymbol{B}_i} ||\boldsymbol{F}_y - \boldsymbol{F}_i \boldsymbol{B}_i||_2^2 + \lambda_B ||\boldsymbol{B}_i||_{1,2}^{rows} \tag{66}$$

This scheme allows to select informative descriptors that are representative for several videos of the same class, avoiding to use descriptors specific to a single video. Using these class-relevant descriptors for each video, we can compute the dictionary D_0 as explained before.

Once the dictionary $\boldsymbol{D}_0$ and the sparse vectors $\boldsymbol{A}$ are computed for every video, the next step is to split the video into R temporal regions (uniform non-overlapping splitting for starting), and using a pooled descriptor x_r for every region. These pooled vectors x_r are the descriptors used by the next level of the hierarchy. Note that each coefficient in the descriptor encodes trajectory shape, trajectory appearance and also location of the important information of every region.

Energy of a video The energy of the "level 1" of the hierarchy is related to learn a set of linear classifiers conforming a region dictionary, that we can interpret as coding building blocks that compose more complex actions. Using the same names as our previous model, we can cast the regions belonging to one out of K *poses*. We use a fixed number of K linear classifiers in this level, and use a special label $K + 1$ indicating when a region is non-informative about any pose. When the pose of region r, z_r, is one of the K valid poses, the region score is the dot product of the linear classifier assigned to the region according to z_r; otherwise, when $z_r = K + 1$, we use a fixed real number θ_{pose}. The energy equation for pose level is

$$E_{\text{pose}} = \sum_{r=1}^{R} \left(\sum_{k=1}^{K} w_k^\top x_r \delta(z_r = k) + \theta_{\text{pose}} \delta(z_r = K + 1) \right) \tag{67}$$

For the action level, we define a set of A latent actions (not latent when there exist mid-level annotations), using a special action label $A + 1$ to assign those regions for which action classifiers does not produce a "good enough" score (that we treat as non-informative). A first approach is to use histograms of poses in those regions labeled as informative and belonging to the same action, and learn linear classifiers β for every action, using the pose histograms as inputs. Similar to poses, we add a fixed score θ_{action} for those non-informative regions. Also, we need to promote temporal continuity for actions, to avoid noisy labels when imputing latent actions, so we add a continuitity term that foster the actions to be grouped in neighboring regions. The energy for action level is then

$$E_{\text{action}} = \sum_{r=1}^{R} \left(\sum_{a=1}^{A} \sum_{k=1}^{K+1} \beta_{a,k} \delta(z_r = k) \delta(v_r = a) + \theta_{\text{action}} \delta(v_r = A + 1) \right) + \theta_{\text{cont}} \sum_{a=1}^{A+1} \left(\sum_{r=1}^{R-1} \delta(v_r = v_{r+1}) \right)^2 \tag{68}$$

For activity level we use the same activity energy equation as our previous model, including terms to quantify the contribution of non-informative regions to the global activity and a global appearance descriptor (for the moment, represented by a vector g_i) built, for example, by a BoW representation of shapes and color, or the pooled scores of several object classifiers (we need to further investigate about this representation). The energy equation of this level is:

$$E_{\text{activity}} = \sum_{r=1}^{R} \sum_{c=1}^{L} \sum_{a=1}^{A+1} \alpha_{c,a}^{act} \delta(c = y) \delta(v_r = a) + \alpha_y^{app\top} g_i \tag{69}$$

For transition terms, the action continuity term acts as an inertial force to keep the current action label in the next region. In our effort of simplifying the transition terms, we must add energy terms associated to transitions between informative and non-informative poses and actions, and terms associated to keep or change the action or pose between two informative ones.

$$
\begin{aligned}
E_{\text{action-transition}} = \sum_{r=1}^{R} \Big(& \gamma_1 \delta(v_r < A+1)\delta(v_{r+1} = A+1) + \gamma_2 \delta(v_r = A+1)\delta(v_{r+1} < A+1) \\
& + \delta(v_r < A+1)\delta(v_{r+1} < A+1) \Big[\gamma_3 \delta(v_r = v_{r+1}) + \gamma_4 \delta(v_r \neq v_{r+1}) \Big] \\
& + \gamma_5 \delta(v_r = A+1)\delta(v_r = v_{r+1}) \Big)
\end{aligned}
\tag{70}
$$

$$
\begin{aligned}
E_{\text{pose-transition}} = \sum_{r=1}^{R} \Big(& \eta_1 \delta(z_r < K+1)\delta(z_{r+1} = K+1) + \eta_2 \delta(z_r = K+1)\delta(z_{r+1} < K+1) \\
& + \delta(z_r < K+1)\delta(z_{r+1} < K+1) \Big[\eta_3 \delta(z_r = z_{r+1}) + \eta_4 \delta(z_r \neq z_{r+1}) \Big] \\
& + \eta_5 \delta(z_r = K+1)\delta(z_r = z_{r+1}) \Big)
\end{aligned}
\tag{71}
$$

The final energy of each video is stated as

$$
E_{\text{video}} = E_{\text{activity}} + E_{\text{action}} + E_{\text{pose}} + E_{\text{action-transition}} + E_{\text{pose-transition}} \tag{72}
$$

Optimization model The energy of each video gives us a score of how well a sequence of poses and actions matches the activity, which regions must be selected to be informative, and also quantifies how well "behaved" actions are according to their temporal continuity. For each video, the score of the well-behaved labeling must be always greater to the score of incorrect labeling, in terms of the activity prediction and the behavior of actions and poses. We can also penalize those bad-behaved sequences with a loss function that acts as a minimum margin between correct and incorrect labels. We define the loss-function penalizing three types of sources of bad behavior: when the activity label is incorrect, when an action time span is too small for temporally consecutive regions, and when a single action is described with "a lot" of poses. Defining $J(a)$ as the diversity of learned poses for the regions labeled with action a,

$$
J(a) = \sum_{k=1}^{K} \delta \left(\left[\sum_{r=1}^{R} \delta(v_r = a)\delta(z_r = k) \right] > 0 \right), \tag{73}
$$

the new loss function for latent actions, where d_{min} is defined as the minimum duration in terms of number of consecutive regions, is written as:

$$
\Delta_1((y_i),(y,V,Z)) = \lambda_1 \delta(y_i \neq y) + \frac{\lambda_3}{R} \sum_{r=1}^{R} d(r) + \lambda_4 \sum_{a=1}^{A} l(a) \tag{74}
$$

where $d(r)$ is a binary variable indicating if any action that starts in region r in the video has a duration less than d_{min}, written as

$$
d(r) = \delta(v_{r-1} \neq v_r) \sum_{a=1}^{A+1} \delta(v_r = a) \delta \left[\left(\sum_{r'=r}^{r+d_{min}-1} \delta(v'_r = a) \right) < d_{min} \right] \tag{75}
$$

and $l(a)$ is a quadratic function on the diversity of poses in the action a in a video, in the part that exceed a fixed number of poses λ_5, written as

$$
l(a) = \left(\max\{0, J(a) - \lambda_5\} \right)^2. \tag{76}
$$

If there exist mid-level annotation (actions), the loss function can penalize also the incorrect action labeling, as

$$
\Delta_2((y_i,V_i),(y,V,Z)) = \lambda_1 \delta(y_i \neq y) + \frac{\lambda_2}{R} \sum_{r=1}^{R} \delta(v_{r_i} \neq v_r) + \frac{\lambda_3}{R} \sum_{r=1}^{R} d(r) + \lambda_4 \sum_{a=1}^{A} l(a) \tag{77}
$$

The optimization model is then formulated as a Latent Structural SVM formulation; the problem can be solved using Concave-Convex Procedure (CCCP), alternating between maximizing equation (79) with respect to latent variables and afterwards, solving a structural SVM optimization problem treating latent variables as completely observed.

$$(V_i^*, Z_i^*) = \arg\max_{V,Z} E(X_i, Z, V, y_i) \tag{78}$$

$$\begin{aligned} &\min_{\alpha,\beta,w,\gamma,\eta,\xi} \Omega(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{w}, \boldsymbol{\gamma}, \boldsymbol{\eta}) + \frac{C}{M}\sum_{i=1}^{M} \xi_i \\ &\text{subject to:} \\ &\quad E(X_i, Z_i, V_i, y_i) - E(X_i, Z, V, y) \geq \Delta_1((y_i), (y, V, Z)) - \xi_i,\ \forall y \in \mathcal{Y}, Z \in \mathcal{Z}, V \in \mathcal{V} \end{aligned} \tag{79}$$

where

$$\begin{aligned} \Omega(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{w}, \boldsymbol{\gamma}, \boldsymbol{\eta}) = &\frac{\mu_\alpha}{2}||\boldsymbol{\alpha}^{act}||_2^2 + \rho_\alpha||\boldsymbol{\alpha}^{act}||_1 + \frac{1}{2}||\boldsymbol{\alpha}^{app}||_2^2 + \frac{\mu_\beta}{2}||\boldsymbol{\beta}||_2^2 + \rho_\beta||\boldsymbol{\beta}||_1 \\ &+ \frac{\mu_w}{2}||\boldsymbol{w}||_2^2 + \rho_w||\boldsymbol{w}||_1 + \frac{1}{2}||\boldsymbol{\gamma}||_2^2 + \frac{1}{2}||\boldsymbol{\eta}||_2^2 + \frac{1}{2}||\boldsymbol{\theta}||_2^2 \end{aligned} \tag{80}$$

defines a regularizer with sparse components in α, β and w, using the *Elastic Net Regularizer*, building a convex and differentiable objective function.

Learning

Inference

Comments

- There is some research to do about the background subtraction, since using Ransac to compute an homography over two consecutive frames is very expensive. I think there might be a better way to exclude background movements.
- An interesting extension is to augment the level 0 descriptor with several human/object detector scores, to encode trajectory, appearance, location and “objectness” of every tracked trajectory.
- There is no optimizations included in the document, they will be described later.

References

[1] Wang, J., Liu, Z., Wu, Y., Yuan, J. (2012). Mining actionlet ensemble for action recognition with depth cameras. *In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on (pp. 1290-1297)*. IEEE.

[2] Chaudhry, R., Ofli, F., Kurillo, G., Bajcsy, R., Vidal, R. Bio-inspired Dynamic 3D Discriminative Skeletal Features for Human Action Recognition.

[3] Wang, C., Wang, Y., Yuille, A. L. An approach to pose-based action recognition.

[4] Yu, Chun-Nam John, and Thorsten Joachims. ”Learning structural svms with latent variables.” Proceedings of the 26th Annual International Conference on Machine Learning. ACM, 2009. 4

[5] Joachims, Thorsten, Thomas Finley, and Chun-Nam John Yu. ”Cutting-plane training of structural SVMs.” Machine Learning 77.1 (2009): 27-59. 4