Structured Output-Associative Regression

Liefeng Bo
Toyota Technological Institute at Chicago, USA
blf0218@tti-c.org

Cristian Sminchisescu
University of Bonn, Germany
cristian.sminchisescu@ins.uni-bonn.de

Abstract

Structured outputs such as multidimensional vectors or graphs are frequently encountered in real world pattern recognition applications such as computer vision, natural language processing or computational biology. This motivates the learning of functional dependencies between spaces with complex, interdependent inputs and outputs, as arising e.g. *from images and their corresponding 3d scene representations. In this spirit, we propose a new structured learning method*—Structured Output-Associative Regression (SOAR)—*that models not only the input-dependency but also the self-dependency of outputs, in order to provide an output re-correlation mechanism that complements the (more standard) input-based regressive prediction. The model is simple but powerful, and, in principle, applicable in conjunction with any existing regression algorithms. SOAR can be kernelized to deal with non-linear problems and learning is efficient via primal/dual formulations not unlike ones used for kernel ridge regression or support vector regression. We demonstrate that the method outperforms weighted nearest neighbor and regression methods for the reconstruction of images of handwritten digits and for 3D human pose estimation from video in the HumanEva benchmark.*

1. Introduction

We study continuous structured prediction methods inspired by models and representations in computer vision and pattern recognition. In this context, visual pose estimation, segmentation, or object and action recognition can be formulated as learning of complex functional dependencies between multivariate input and output representations.[1] For vision, the input is often an image or a descriptor, *e.g*. a histogram that quantizes the occurrence of color, gradient features or object parts over an image, and the output is a continuous structured scene representation—an object shape, a human pose, or a sequence of temporal state labels. Both inputs and outputs are high-dimensional and structured. Input descriptors reflect correlated spatial image statistics and the outputs (the scene representations) are correlated due to regularities in the 3d world. For example, people most often move on the ground, and their motion is not only temporally coherent but also constrained by physical factors like equilibrium or by functional factors like actions, intentions, or avoidance of obstacles. People also interact with objects made of parts that cannot occur in arbitrary configurations.

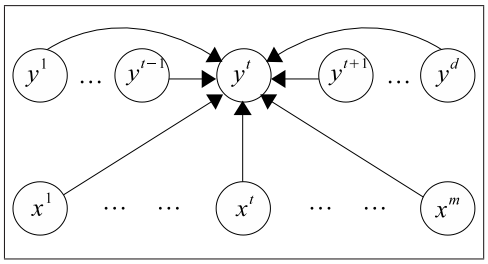

Figure 1. Dependency model for one output variable of SOAR. y^t is the current output, and m and d are dimensionality of inputs and outputs, respectively. Notice y^t not only depends on the input $\mathbf{x} = [x^1, \ldots, x^m]^\top$, but also on the remaining outputs $\mathbf{y}^{-t} = \mathbf{y} \setminus y^t$.

To conclude, in structured vision problems each output depends not only on the input, but also on output components other than itself. Neglecting output correlations, *e.g*. by predicting them independently, thus failing to use the appropriate context, disregards valuable information and can lead to inconsistent, suboptimal estimates. In this paper, we present a novel continuous structured prediction method—**Structured Output-Associative Regression (SOAR)**—which learns functional dependencies where outputs are both input-dependent and self-dependent (fig. 1). This effectively augments a classical input-based regressor with output auto-association, where outputs other than the one being predicted act as auxiliary features. Inference is coupled with the goal to make the prediction resonate with completion from auxiliary outputs. SOAR can be generalized with kernels that can solve non-linear problems and

[1] Vectorized representations are to some degree incidental, as vectors can encode symbolic, structured representations, like graphs, trees or sequences, and these can be compared using sophisticated kernel-based similarity measures, rather than Euclidean ones.

scales well to large datasets. Because learning separates, efficient finite-dimensional primal/dual formulations similar to kernel ridge regression or support vector regression can be used. We show that the algorithm outperforms regression and weighted nearest neighbor methods for the reconstruction of 3d human motion from monocular and multicamera video.

1.1. Related Work

This research primarily focuses on structured prediction and its computer vision applications, most notably 3d human pose estimation. Discriminative methods for 3d prediction rely on a training set of images and ground truth 3d data (*e.g*. human pose obtained from mocap systems) in order to train models of various complexity: nearest-neighbor schemes [20, 17], regression and variants [1], mixture of experts [18, 23, 22] or semi-supervised models [15] have been demonstrated to give good practical performance. Their main advantage stems from being automatic: hand-specifying the initial pose or the camera calibration parameters is no longer necessary. The methods are conceptually simpler than more sophisticated generative methods based on physical models and they are often faster. However, so far discriminative vision methods focused less on prediction with interdependent outputs: many existing algorithms process the outputs as being independent and train separate models for each one (but see [23, 5]). For complex structured problems, this strategy can be both inconsistent and sub-optimal. As the output dimensionality increases, to learn efficiently, one has to consider correlations between outputs, correlations between inputs, as well as joint correlations. There is also a choice of modeling correlations as part of the loss function, or as a form of regularization.

One way to model input correlations for continuous structured prediction is by sharing the weights of correlated features. Other methods rely on low-dimensional input representations: partial least squares and its extensions (sliced-inverse regression) [7, 11] recover a linear subspace that is informative for prediction whereas manifold regression [16] pursues the non-linear case using cross-covariance operators. Kernel dependency estimation (KDE) [27] uses kernel PCA to model (embed) correlations among both inputs and outputs and fits a regression model in the embedded space. A pre-image calculation is required in order to recover the output in the original representation. Cortes and Mohri [8] give a conceptually cleaner reformulation of KDE where the kernel PCA is no longer explicitly required. Applications of KDE methods in vision in conjunction with conditional temporal models are given in [23].

Alternatively to assuming a manifold structure, Micchelli and Pontil [14] used a matrix-valued kernel to model structure in the outputs. Their formulation yields a more expensive training procedure, as the number of variables equals to the dimensionality of the output times the size of training set, and the resulting program is not sparse. There is also a substantial volume of research focusing on the structured prediction case for discrete outputs, both in the probabilistic, maximum likelihood setting, *e.g*. dependency networks, MEMM, CRF [10, 13, 12] and for max-margin losses, *e.g*. SVM-ISOS [26]. Generalizations to continuous outputs are given in [28]. Structural SVMs learn a scoring function so that the pair corresponding to the given input-output training example ranks higher than a pair formed by the given input and any other output.[2] The methods need to handle infinitely many constraints, which leads to semi-infinite programs. Duality theorems similar in spirit to the strong/weak duality for the finite case exist, in principle. In practice, the models are solved using cutting plane algorithms, by creating a nested sequence of successively tighter relaxations of the original optimization problem and finding a small set of active constraints that ensure sufficiently accurate solutions.[3]

2. Structured Output-Associative Regression

In this section we will describe our **Structured Output-Associative Regression Model (SOAR)** with both its kernel ridge regression (§2.1) and support vector regression variants (§2.2). The basic principles are similar, the difference lies in the loss functions used (square loss versus ϵ-insensitive loss) and the optimization methods employed to estimate parameters.

In multiple output regression, we are given a training set of input-output pairs $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$, where N is the size of training set, $\mathbf{y} \in \mathbb{R}^d$ are interdependent outputs, and d is the dimensionality of the output. The goal is to learn the vector-valued function $\mathbf{f} = [f^1, \ldots, f^d]^\top$ which best represents the relationship between inputs and outputs.

In SOAR, the basic idea is that output components other than the one we focus on at a given time can be considered as auxiliary features and used to complement the more standard (for regressive models) input features. We assume that the current output y^j is related to all the other outputs $\mathbf{y}^{-j}$ by a discriminative function f^j with generalized linear form

$$f^j = (\mathbf{u}^j)^\top \phi(\mathbf{x}) + (\mathbf{v}^j)^\top \varphi(\mathbf{y}^{-j}) + b^j \qquad (1)$$

where $\mathbf{y}^{-j}$ is the $(d-1)$-dimensional output vector with the j-th entry removed, $\mathbf{u}^j$ and $\mathbf{v}^j$ are parameter vectors,

[2] Max-margin methods aim to make the true solution out-score any other option, whereas maximum-likelihood criteria (in normalized models) aims at maximizing the volume associated to it. The former rely on global optimization, the latter on integration. For most high-dimensional continuous models, exact calculations of both measures are intractable.

[3] Structural SVM [4] were applied to predict the coordinates of a bounding box of an object (a continuous subroutine within an object detector) by defining a special histogram kernel over image regions contained within the bounding box. Their efficient, tractable branch-and-bound search strategy is however not immediately applicable to general output kernels, *e.g*. the Gaussian, or three-dimensional state spaces as considered here.

with b^j the bias, and $\phi(\mathbf{x})$ and $\varphi(\mathbf{y}^{-j})$ are high dimensional features induced by kernel functions $k^\phi(\mathbf{x}_r, \mathbf{x}_s)$ and $k^\varphi(\mathbf{y}_r^{-j}, \mathbf{y}_s^{-j})$, respectively.

The problem of learning a function from finite samples can suffer from over-fitting without capacity control. Typically we **learn** the parameters of the vector-value function $\mathbf{f}$ by optimizing the regularized empirical risk $\mathcal{R}_L$

$$\{\widehat{\mathbf{u}}^j, \widehat{\mathbf{v}}^j, \widehat{b}^j\}_{j=1}^d = \operatorname{argmin}_{\{\mathbf{u}^j, \mathbf{v}^j, b^j\}_{j=1}^d} \mathcal{R}_L \equiv \tag{2}$$

$$\equiv \sum_{j=1}^{d} \sum_{i=1}^{N} \left\{ L(y_i^j, f^j(\mathbf{x}_i, \mathbf{y}_i^{-j}; \mathbf{u}^j, \mathbf{v}^j, b^j) + \Omega(\mathbf{u}^j, \mathbf{v}^j; \mathbf{C}^j) \right\}$$

where L is the loss function, $\Omega(\mathbf{u}^j, \mathbf{v}^j; \mathbf{C}^j)$ is the regularizer and $\mathbf{C}^j$ are hyperparameters that control the intensity of regularization. Throughout the paper, we assume the following L_2-norm regularizers

$$\Omega(\mathbf{u}^j, \mathbf{v}^j; \mathbf{C}^j) = \frac{C_1^j}{2}||\mathbf{u}^j||^2 + \frac{C_2^j}{2}||\mathbf{v}^j||^2 \tag{3}$$

Notice that $\{\mathbf{u}^i, \mathbf{v}^i, b^i\}$ and $\{\mathbf{u}^j, \mathbf{v}^j, b^j\}$, where $i \neq j$, in (2) are decoupled, so groups of parameters for each dimension can be optimized independently. Unlike classical regression however, we cannot estimate the output directly from the discriminative function and a new input $\mathbf{x}$ since unknown output components are involved in the second term in (1). Therefore, for **inference** we optimize, w.r.t $\mathbf{y}$, the regularized loss defined over the given input $\mathbf{x}$ and the unknown output vector $\mathbf{y}$

$$\widehat{\mathbf{y}} = \operatorname{argmin}_{\mathbf{y}} \mathcal{R}_I \equiv \sum_{j=1}^{d} L(y^j, f^j(\mathbf{x}, \mathbf{y}^{-j}; \widehat{\mathbf{u}}^j, \widehat{\mathbf{v}}^j, \widehat{b}^j)) \tag{4}$$

where we dropped constant regularization terms. There is no closed-form solution for the general case (4): non-linear optimization is required to make predictions. However, for structured problems with medium-sized output dimensionality ($d < 1000$) this aspect is not of major practical concern. The optimizer converges fast, particularly when reasonable initializations are given. In practice, good initializations are obtained from the corresponding independent-output regression model (comparisons are given in §3, see also fig. 2).[4]

Our model covers classical independent-output regression as a special case. If $C_2^j = \infty,\ j = 1 \dots d$, the parameter vector $\mathbf{v}$ is set to 0 to make the loss (2) as small as possible (any nonzero $\mathbf{v}$ leads to infinite loss), which effectively turns-off the auxiliary, self-dependent output term $(\mathbf{v}^j)^\top \varphi(\mathbf{y}^{-j})$. Inference also trivializes: the $\mathbf{y}$ that minimizes (4) is $y^j = (\widehat{\mathbf{u}}^j)^\top \phi(\mathbf{x}) + \widehat{b}^j$ and the minimum of (4) is always 0.

[4]The model works even if some outputs are perfectly correlated in training. The method will not discard their inputs because: (*i*) initialization uses regression with no output self-dependency, hence output will be at least as good as input prediction could deliver, and (*ii*) L2-norm regularizers are not perfect feature selectors (input contributions cannot be turned off).

In principle, we can work with any type of loss. Here, we consider the two most popular ones used in regression: the square loss and the ϵ-insensitive loss, which lead to the SOAR$_{krr}$ and SOAR$_{svr}$ models, respectively.

2.1. SOAR$_{krr}$

In the framework of SOAR, we can extend kernel ridge regression [19] to structured prediction by considering the square loss

$$L(y, f) = \frac{1}{2}(y - f)^2 \tag{5}$$

The regularized loss over the training set is

$$\min_{(\mathbf{u}^j, \mathbf{v}^j)} \ \frac{1}{2}\sum_{i=1}^{N} \xi_i^2 + \frac{C_1^j}{2}||\mathbf{u}^j||^2 + \frac{C_2^j}{2}||\mathbf{v}^j||^2$$

$$s.t. \qquad \xi_i = y_i^j - (\mathbf{u}^j)^\top \phi(\mathbf{x}_i) - (\mathbf{v}^j)^\top \varphi(\mathbf{y}_i^{-j}) \tag{6}$$

In the most cases, the dimensionality of $\phi(\mathbf{x})$ or $\varphi(\mathbf{y}^{-j})$ is much higher than training set size N (*e.g.* for Gaussian kernels, the feature space is infinitely dimensional). Therefore, the optimization problem (6) can be more easily solved in its dual form. The Lagrangian H corresponding to the unconstrained (6) is

$$H = \frac{1}{2}\sum_{i=1}^{N} \xi_i^2 + \frac{C_1^j}{2}||\mathbf{u}^j||^2 + \frac{C_2^j}{2}||\mathbf{v}^j||^2 + \tag{7}$$

$$+ \sum_{i=1}^{N} \alpha_i^j \{y_i^j - (\mathbf{u}^j)^\top \phi(\mathbf{x}_i) - (\mathbf{v}^j)^\top \varphi(\mathbf{y}_i^{-j}) - \xi_i\}$$

where α_i^j are Lagrange multipliers (or dual variables). From the KKT conditions, the partial derivatives of the Lagrangian H w.r.t. the primal variables must vanish

$$\begin{cases} \mathbf{u}^j = \frac{1}{C_1^j}\sum_{i=1}^{N} \alpha_i^j \phi(\mathbf{x}^i) \\ \mathbf{v}^j = \frac{1}{C_2^j}\sum_{i=1}^{N} \alpha_i^j \varphi(\mathbf{y}_i^{-j}) \\ \xi_i = \alpha_i^j, \ \ i = 1 \dots N \end{cases} \tag{8}$$

Substituting (8) into (7), we obtain the dual optimization

$$\min_\alpha \ \frac{1}{2}(\alpha^j)^\top \left(\frac{1}{C_1^j}\mathbf{K}^\phi + \frac{1}{C_2^j}\mathbf{K}^\varphi + \mathbf{I} \right) \alpha^j - \sum_{i=1}^{N} \alpha_i^j y_i^j \tag{9}$$

where $\mathbf{K}^\phi$ and $\mathbf{K}^\varphi$ are $N \times N$ kernel matrices with $\mathbf{K}_{st}^\phi = k^\phi(\mathbf{x}_r, \mathbf{x}_s)$ and $\mathbf{K}_{rs}^\varphi = k^\varphi(\mathbf{y}_r^{-j}, \mathbf{y}_s^{-j})$. Since (9) is an unconstrained convex quadratic program, we obtain a closed-form solution for learning

$$\widehat{\alpha}^j = \left(\frac{1}{C_1^j}\mathbf{K}^\phi + \frac{1}{C_2^j}\mathbf{K}^\varphi + \mathbf{I} \right)^{-1} (\mathbf{Y}^j)^\top \tag{10}$$

where $\mathbf{Y}^j$ is the j-th row of $\mathbf{Y}$ which stores the outputs columnwise.[5] Combining (8) and (10), we obtain the struc-

[5]For large datasets it can be prohibitive to invert the kernel matrix, as this may not fit into memory (this is known as the out-of-memory case).

tured regression function, here referred as $\mathbf{f}_{krr}$

$$f^j_{\text{krr}} = \left\{ \frac{1}{C^j_1} \sum_{i=1}^{N} \widehat{\alpha}^j_i \phi(\mathbf{x}_i) \right\}^\top \phi(\mathbf{x}) + \\ + \left\{ \frac{1}{C^j_2} \sum_{i=1}^{N} \widehat{\alpha}^j_i \varphi(\mathbf{y}_i^{-j}) \right\}^\top \varphi(\mathbf{y}^{-j}) = \\ = \frac{1}{C^j_1} \sum_{i=1}^{N} \widehat{\alpha}^j_i k^\phi(\mathbf{x}, \mathbf{x}_i) + \frac{1}{C^j_2} \sum_{i=1}^{N} \widehat{\alpha}^j_i k^\varphi(\mathbf{y}^{-j}, \mathbf{y}_i^{-j}) \quad (11)$$

The inference process for SOAR$_{krr}$ is

$$\widehat{\mathbf{y}} = \text{argmin}_{\mathbf{y}} \mathcal{R}_I \equiv \frac{1}{2} \sum_{j=1}^{d} (y^j - f^j_{\text{krr}})^2 \quad (12)$$

Notice that optimization variables are inside the kernel function (referred as the pre-image problem in the kernel community, although, strictly, we solve a more general problem than finding a kernel argument from a given target feature space value, as our target is itself unknown). We optimize (12) using a BFGS quasi-Newton optimizer with cubic polynomial line search for optimal step size selection (see also fig. 2). We initialize the optimizer using prediction given by kernel ridge regression trained on independent outputs. In our (rather extensive) experiments, this consistently improved both the accuracy and the speed of prediction compared to random initialization. Our models can also

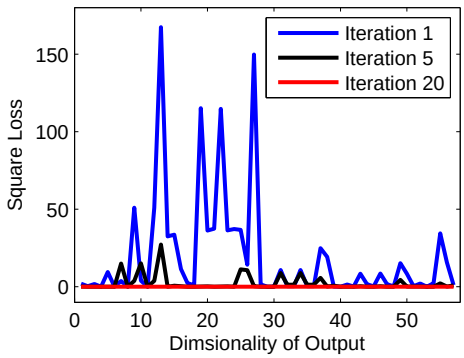

Figure 2. The progress of our BFGS optimizer on a test sample of box motions in HumanEva-I. Initialization is given by kernel ridge regression assuming independent outputs. The horizontal-axis is the dimensionality of the body joint positions and the vertical-axis is the corresponding squared loss $\frac{1}{2}(y^j - f^j_{\text{krr}})^2$. Notice the rapid convergence on this dataset.

be understood as learning a nonlinear mapping over outputs. To make this more clear, we rewrite (6) as

$$\frac{1}{2} \sum_{i=1}^{N} (z^j - (\mathbf{u}^j)^\top \varphi(\mathbf{x}_i))^2 + \frac{C^j_1}{2} ||\mathbf{u}^j||^2 + \frac{C^j_2}{2} ||\mathbf{v}^j||^2 \quad (13)$$

One can then resort on iterative methods such as conjugate gradient or greedy block coordinate descent to solve (9).

with: $z^j = g^j(\mathbf{y}) = y^j - (\mathbf{v}^j)^\top \varphi(\mathbf{y}^{-j})$, where $g^j(\mathbf{y})$ can be viewed as a nonlinear mapping from the d dimensional output space to the d dimensional embedding space. According to this view, our model bears certain resemblance with kernel dependency estimation [27], which directly defines a nonlinear mapping $\varphi(\mathbf{y})$ over outputs instead of learning it. However, the dimensionality of the embedding space in kernel dependency estimation can be infinite (*e.g.* for Gaussian kernels) and the methods resort on kernel principal component analysis to project $\varphi(\mathbf{y})$ to the first p principal components—an operation that incurs additional training/testing costs, as kernel principal component analysis and the dimensionality of the embedding space scale as $O(n^3)$. In contrast, output dimensionality is d in SOAR, and no extra step is required—one only needs to train d separable models and many efficient decomposition algorithms are available such as sequential minimal optimization (SMO) [9] and greedy block coordinate descent.

2.2. SOAR$_{svr}$

For structured-output associative support vector regression models, we consider the ϵ-insensitive loss [25]

$$L(y, f) = \max(0, |y - f| - \epsilon) \quad (14)$$

where ϵ is a small constant. Inference in SOAR-SVM requires to minimize the ϵ-insensitive loss w.r.t. the output vector $\mathbf{y}$

$$\widehat{\mathbf{y}} = \text{argmin}_{\mathbf{y}} \sum_{j=1}^{d} \max(0, |y^j - f^j_{\text{svr}}| - \epsilon) \quad (15)$$

where $\mathbf{f}_{svr}$ is the learned target function. Similarly with SOAR$_{krr}$, we minimize (15) using a sub-gradient optimizer, initialized with prediction given by support vector regressors trained on independent outputs. Detailed derivations for SOAR$_{svr}$ are given in the Appendix.

3. Experiments

In this section, we illustrate our models for the reconstruction of images of handwritten digits and 3d human pose reconstruction in the HumanEva benchmark, where we give results for *both monocular and multicamera* video. For experiments, we use Gaussian kernels $k^\phi(\mathbf{x}_r, \mathbf{x}_s)$ and $k^\varphi(\mathbf{y}_r^{-j}, \mathbf{y}_s^{-j})$. The kernel hyperparameters and the regularizers C^j_1 and C^j_2 are grid-searched by five-fold cross validation (training set), and selected to be the same for all output dimensions.

3.1. Handwritten Digits Reconstruction

We considered a problem of image reconstruction inspired by [27]: given the outer 240 pixel values of a handwritten digit from USPS test set (2007 samples), the goal

Models	Training set size		
	500	2000	7291
NN	0.414	0.372	0.341
KRR	0.367	0.296	0.250
SVR	0.366	0.295	0.250
KDE	0.373	0.304	0.260
SOAR$_{krr}$	0.339	0.278	0.233
SOAR$_{svr}$	**0.332**	**0.273**	**0.230**

Table 1. Comparisons of mean absolute error of the different models on test set. The gray values are normalized to [0, 2], as provided. The lowest error is indicated in bold. NN means nearest neighbor regression, KDE means kernel dependency estimation[27] with 16d latent space obtained by kernel principal component analysis (SOAR also predicts 16d outputs).

is to predict the 16 pixel values lying in the center based on 7291 training samples (we do not assume knowledge of the label of the digit). We compare SOAR with nearest neighbor regression, kernel ridge regression, support vector regression and kernel dependency estimation in table 1. SOAR methods obtain the lowest test error, typically 5-10% improved relative to KRR and SVR.

3.2. Human Pose Estimation

The performance of different models is evaluated on the HumanEva-I dataset [21], which consists of 4 subjects performing 6 predefined actions: walking, jogging, throw-catch, gestures, boxing and combo (walking followed by jogging and then balancing on each one of the two feet). Combo motions of all subjects and all motions of one subject are withheld. Table 2 summarizes the training and validation set.

Features	Action	Subject 1	Subject 2	Subject 3	Total
SC	Walking	1197	870	931	2998
	Jogging	597	789	834	2220
	Gestures	795	893	1096	2784
	Box	783	652	1015	2450
	ThrowCatch	217	1011	0	1228
	Total	3589	4215	3876	11680
BSIFT	Walking	1176	876	895	2947
	Jogging	439	795	831	2065
	Gestures	801	681	214	1696
	Box	502	464	933	1889
	ThrowCatch	217	806	0	1023
	Total	3135	3622	2873	9630

Table 2. Size of training and validation set of each motion of each subject for SC and block SIFT (BSIFT) features on HumanEva-1.

Image Descriptors: We consider two types of features: histograms of shape contexts (SC) and SIFT descriptors extracted on a regular grid and concatenated in a long descriptor BSIFT (block SIFT). For SC [3, 6], we use non-parametric models and adaptive threshold procedures for background subtraction. Edges are extracted from the silhouette image and 400 points are sampled on edges. The shape context descriptor at each image location is computed based on 15 angular bins and 8 radial bins. The SC at each of the 400 points per image are computed every 15th image in training and used to generate a 300-d codebook, learned using k-means (hence the descriptor size is 300). For BSIFT [24, 17], we use background subtraction with risk values suggested in HumanEva's documentation [21]. The silhouette bounding box is divided in a 6 x 5 cell grid, and gradient orientations in each cell are quantized into 9 orientation bins ($0^0 - 180^0$), for a descriptor of length 270.

Pose encoding: Three-dimensional human poses ($\mathbf{y}$) are represented as 60d vectors of three-dimensional body joint positions (20 markers each with $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ coordinates) using 'torsoDistal' as root. All poses are preprocessed by subtracting the root from all the other joint positions in every frame. The prediction error is the Euclidean distance between the estimated joint position and the true position averaged over all joints, per frame [21]: $\mathrm{Err}_{seq} = \frac{1}{T}\sum_{i=1}^{T} D(\mathbf{y}_i, \overline{\mathbf{y}}_i)$, where T is the length of sequence and $D(\mathbf{y}_i, \overline{\mathbf{y}}_i) = \frac{1}{M}\sum_{j=1}^{M} \|m_j(\overline{\mathbf{y}}_i) - m_j(\mathbf{y}_i)\|$, where $m_j(\mathbf{y}_i) \in \mathbb{R}^3$ is a function which extracts the three dimensional coordinates of the jth joint position, M is the number of the positions, and $\|\cdot\|$ the Euclidean distance.

To give additional intuition on why structured prediction is adequate for the problem, we show correlation coefficients corresponding to the five motion types in fig. 3. Entries are computed as: $\rho_{ij} = \frac{\mathrm{cov}(y^i, y^j)}{\sqrt{\mathrm{cov}(y^i, y^i)\mathrm{cov}(y^j, y^j)}}$, where $\mathrm{cov}(\cdot,\cdot)$ is the covariance between two random variables. The light, highly correlated components outside diagonal, indicate non-negligible coupling between joint positions.

We report the average joint position error of WKNN (weighted k-nearest neighbors), KRR (kernel ridge regression with independently trained output dimensions), SOAR$_{krr}$, SVR (support vector regression with independently trained outputs) and SOAR$_{svr}$. In table 3 and 4 we report results on HumanEva's validation set.[6] We run two types of experiments: one trained on a single motion of one subject and tested on the motion of corresponding subject; the other trained and tested on the full dataset (all motions of all subjects). For the latter, we speed-up using a K nearest neighbors preprocessing for a test input, then perform KRR, SOAR$_{krr}$, SVR, or SOAR$_{svr}$ on the reduced set. A similar approach is used in local regression [2]. (While we can run the methods on the full training set, we have found that all methods benefit from k-nearest neighbor preprocessing.) It can be seen that SOAR$_{krr}$ and SOAR$_{svr}$ consistently outperform their counterparts by about 5-10mm, for each mo-

[6]Notice that for this set of experiments we only train/validate on the HumanEva training set, and test on the validation set, a protocol that is methodologically valid. We do this in order to run more extensive experiments with many methods—this would be otherwise impractical, at the current operating speed of HumanEva's online server (for online reports on the test set, see table 5).

Feature	Motion	KRR	SOAR$_{krr}$	SVR	SOAR$_{svr}$
BSIFT(C1)	Walking	58.9	**49.9**	61.5	**47.6**
	Jogging	65.9	**59.5**	63.2	**57.2**
	Gestures	48.1	**44.2**	47.4	**44.7**
	Box	67.8	**61.6**	64.2	**59.8**
	ThrowCatch	95.9	**94.4**	95.6	**93.0**
	Average	67.3	**61.9**	66.4	**60.5**
SC(C1)	Walking	66.4	**55.3**	67.1	**54.1**
	Jogging	75.5	**68.1**	73.2	**66.7**
	Gestures	54.0	**48.3**	53.1	**46.8**
	Box	81.5	**73.3**	78.3	**71.9**
	ThrowCatch	115.9	**111.0**	114.3	**109.5**
	Average	78.7	**71.2**	77.2	**69.8**

Table 3. Models separately trained on each motion of each subject. Evaluation of different models that use BSIFT and SC features on the validation set of HumanEva-I (error reported in mm). 'C1' means that the image feature is extracted from images captured by the first camera *only (monocular experiments)*. Error is averaged over the same motion of the three subjects. 'Average' accumulates averages for all motions of all subjects.

Feature	Motion	KRR	SOAR$_{krr}$	SVR	SOAR$_{svr}$
BSIFT(C1)	Walking	60.2	**53.4**	59.3	**53.7**
	Jogging	55.6	**49.3**	54.1	**48.6**
	Gestures	47.4	**43.2**	46.9	**42.8**
	Box	68.6	**64.3**	68.4	**63.9**
	ThrowCatch	85.9	**76.4**	85.3	**79.8**
	Average	63.5	**57.3**	62.8	**57.8**
SC (C1)	Walking	67.6	**59.8**	66.3	**58.7**
	Jogging	68.4	**62.7**	67.9	**62.1**
	Gestures	53.9	**49.6**	53.5	**48.3**
	Box	81.1	**77.3**	80.4	**75.6**
	ThrowCatch	116.2	**110.3**	115.4	**109.8**
	Average	77.4	**71.9**	76.7	**70.9**

Table 4. Models trained on all subjects and motions. Except for the different training protocol, all features and error measures are identical to the ones described in table 3. Notice the similar performance in both cases, suggesting that the predictor generalizes well to different subjects—no apriori personalized subject/body model is necessary.

tion.

We report the average joint position error of weighted K nearest neighbors (WKNN) and SOAR$_{krr}$ (also using K nearest neighbors) (both are trained on the training+validation set) on the test set, as computed by HumanEva's online evaluation system, in table 5. The temporal evolution of the average joint position error for WKNN and SOAR$_{krr}$ is given in fig. 4. Sample images and qualitative 3D human pose reconstructions rendered from a synthetic viewpoint, obtained from SOAR$_{krr}$ on test set, are shown in fig. 5.

4. Conclusions

We have introduced a novel continuous structured prediction method—Structured Output-Associative Regression (SOAR)—to learn functional dependencies between spaces with complex, interdependent inputs and outputs, as arising in computer vision and machine learning problems. In the model, output components are constrained both by input features and by complementary outputs acting as auxiliary variables. Inference is coupled with the goal to make prediction resonate with completion from auxiliary outputs. SOAR can be generalized with kernels that can solve nonlinear problems, and scales well to large datasets via efficient primal/dual formulations, for both the square and the ϵ-insensitive loss functions. We show that the algorithm significantly outperforms weighted nearest neighbor and regression methods for the reconstruction of images of hand-

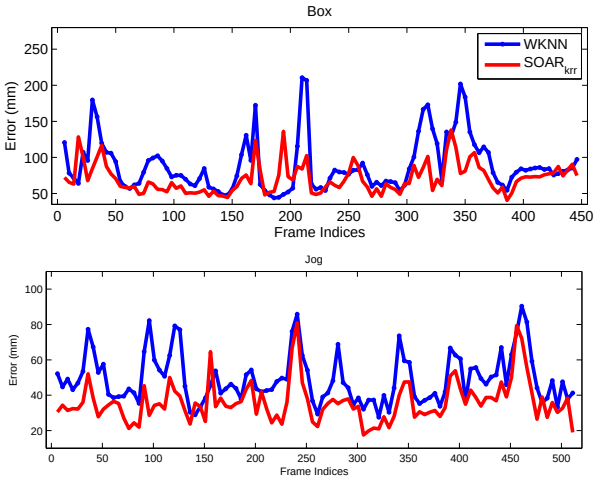

Figure 4. Average joint position test error per frame for two different models, and three different motions, all using BSIFT (C1+C2+C3) image features, as computed by HumanEva's online evaluation system. WKNN and SOAR$_{krr}$ use 25 and 100 nearest neighbors, respectively, both cross-validated.

Figure 5. Qualitative 3d reconstruction results on the HumanEva-1 test set (original images on the top row, 3D reconstructions seen from synthetic viewpoints on the second row).

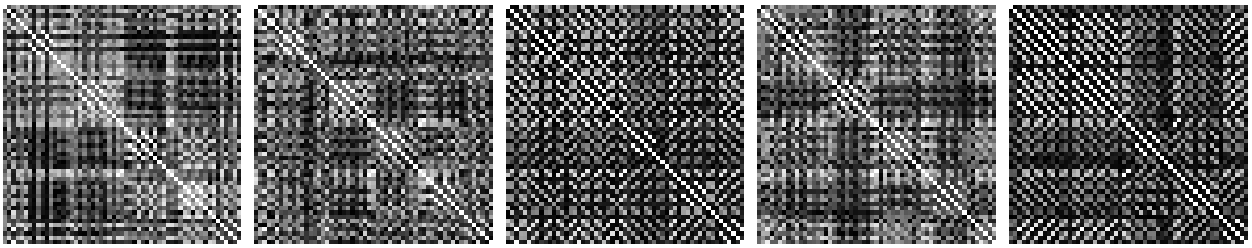

Figure 3. Matrices of correlation coefficients for 3d poses in the training set of HumanEva-I (57d vectors of temporally ordered three-dimensional body joint positions, with root joint 'torsoDistal' removed). From left to right: Box, Gestures, Jogging, ThrowCatch and Walking. Lighter means that corresponding pairs of output variables are more correlated.

Feature	Motion	Subject 1		Subject 1		Subject 3	
		WKNN	SOAR$_{krr}$	WKNN	SOAR$_{krr}$	WKNN	SOAR$_{krr}$
BSIFT(C1)	Walking	47.5(21.1)	45.4(23.2)	46.7(35.2)	39.9(28.0)	66.7(32.0)	43.0(20.3)
	Jogging	54.5(22.4)	46.7(19.3)	43.3(14.4)	37.0(10.3)	56.1(25.4)	46.4(28.9)
	Gestures	23.4(13.5)	21.8(8.0)	75.1(28.1)	74.5(28.1)	75.3(11.1)	64.6(13.6)
	Box	79.7(27.7)	74.7(22.0)	105.8(46.9)	95.7(40.5)	100.4(52.3)	99.3(32.1)
	ThrowCatch	/	/	71.4(34.2)	58.2(25.1)	111.7(32.9)	91.8(29.9)
	Combo	/	/	78.6(48.6)	72.8(52.0)	114.0(77.3)	108.1(82.3)
	Average	51.3	**47.2**	70.1	**63.0**	87.4	**75.5**
BSIFT(C1+C2+C3)	Walking	37.5(12.0)	33.0(11.6)	40.1(23.9)	29.8(13.3)	55.3(25.1)	36.2(16.7)
	Jogging	45.2(13.7)	35.8(11.8)	37.7(12.2)	29.6(7.5)	45.4(18.3)	37.4(14.3)
	Gestures	23.7(7.2)	21.0(3.5)	72.8(26.3)	63.9(21.5)	56.1 (6.4)	48.8(5.2)
	Box	88.7(36.2)	73.9(20.2)	91.8(41.2)	82.2(40.9)	92.3(47.9)	88.1(50.3)
	ThrowCatch	/	/	57.6(23.8)	44.9(23.4)	92.8(31.8)	70.1(32.9)
	Combo	/	/	71.9(52.2)	58.8(50.8)	83.9(53.0)	73.9(52.3)
	Average	48.8	**40.9**	62.0	**51.5**	71.0	**59.1**

Table 5. Models trained on all subjects and motions. Evaluation (BSIFT features) on the test set of HumanEva-I (error reported in mm). Average joint position error is computed by Humaneva's online evaluation system, with lowest error indicated in bold. '/' entries mean that values are not available (no test results returned), 'average' gives averages for different motions of the same subject; 'C1' gives results for image feature extracted from images captured by the first camera *only (monocular experiments)*, notice the difference w.r.t. [17]; for 'C1+C2+C3' image features from three cameras are combined in a single descriptor.

written digits and the estimation of 3D human pose from video in the HumanEva benchmark.

Future Work: We plan to study methods for scaling SOAR to massive datasets, including sparsity (L_1-norm regularizers) and block coordinate descent methods. Iterative output re-estimation procedures, bias and connections with pseudo-likelihood methods are also studied.

Acknowledgements: This work was supported, in part, by the NSF and the EC, under awards 0535140 and MCEXT-025481.

Appendix: Derivations for SOAR$_{svr}$

Given (14), the regularized loss over the training set is

$$\min_{(\mathbf{u}^j,\mathbf{v}^j,b^j)} \sum_{i=1}^{N}(\xi_i+\zeta_i)+\frac{C_1^j}{2}||\mathbf{u}^j||^2+\frac{C_2^j}{2}||\mathbf{v}^j||^2 \quad (16)$$

$$s.t. \quad y_i^j-(\mathbf{u}^j)^\top\phi(\mathbf{x}_i)-(\mathbf{v}^j)^\top\varphi(\mathbf{y}_i^{-j})-b^j\le\epsilon+\xi_i$$
$$y_i^j-(\mathbf{u}^j)^\top\phi(\mathbf{x}_i)-(\mathbf{v}^j)^\top\varphi(\mathbf{y}_i^{-j})-b^j\ge-\epsilon-\zeta_i$$
$$\xi_i,\zeta_i\ge 0,\ \ i=1\dots N$$

The Lagrangian H corresponding to the unconstrained problem (16) is

$$H=\sum_{i=1}^{N}(\xi_i+\zeta_i)+\frac{C_1^j}{2}||\mathbf{u}^j||^2+\frac{C_2^j}{2}||\mathbf{v}^j||^2+ \quad (17)$$
$$+\sum_{i=1}^{N}\alpha_i^j\{y_i^j-(\mathbf{u}^j)^\top\phi(\mathbf{x}_i)-(\mathbf{v}^j)^\top\varphi(\mathbf{y}_i^{-j})-b^j-\epsilon-\xi_i\}$$
$$+\sum_{i=1}^{N}\beta_i^j\{-y_i^j+(\mathbf{u}^j)^\top\phi(\mathbf{x}_i)+(\mathbf{v}^j)^\top\varphi(\mathbf{y}_i^{-j})+b^j-\epsilon-\zeta_i\}$$
$$-\sum_{i=1}^{N}\eta_i\xi_i-\sum_{i=1}^{N}\lambda_i\zeta_i$$

where $\alpha_i^j,\beta_i^j,\eta_i,\lambda_i\ge 0$ are Lagrange multipliers. From the KKT conditions on vanishing partial derivatives of La-

grangian H w.r.t. the primal variables, we obtain

$$\begin{cases} \mathbf{u}^j = \frac{1}{C_1^j}\sum_{i=1}^N(\alpha_i^j - \beta_i^j)\phi(\mathbf{x}_i) \\ \mathbf{v}^j = \frac{1}{C_2^j}\sum_{i=1}^N(\alpha_i^j - \beta_i^j)\varphi(\mathbf{y}_i^{-j}) \\ \sum_{i=1}^N(\alpha_i^j - \beta_i^j) = 0 \\ \alpha_i^j = 1 - \xi_i - \eta_i, \;\; i = 1 \dots N \\ \beta_i^j = 1 - \zeta_i - \lambda_i, \;\; i = 1 \dots N \end{cases} \tag{18}$$

Substituting (18) into (17), we have the dual problem

$$\begin{aligned} H = &\frac{1}{2}(\alpha^j - \beta^j)^\top \left\{ \frac{1}{C_1^j}\mathbf{K}^\phi + \frac{1}{C_2^j}\mathbf{K}^\varphi \right\}(\alpha^j - \beta^j) + \\ &+\epsilon\sum_{i=1}^N(\alpha_i^j + \beta_i^j) - \sum_{i=1}^N y_i^j(\alpha_i^j - \beta_i^j) \\ s.t. \quad &\sum_{i=1}^N(\alpha_i^j - \beta_i^j) = 0, \;\; \alpha_i^j, \beta_i^j \in [0,1] \end{aligned} \tag{19}$$

(19) is a constrained convex quadratic programming with global optimal solution $\{\widehat{\alpha}^j, \widehat{\beta}^j, \widehat{b}^j\}_{j=1}^d$. An interior-point optimizer for (19) requires $O(N^3)$ computational cost and $O(N^2)$ memory storage. Instead, many decomposition algorithms can be used to solve (19) efficiently, with computational cost and memory requirements $O(N^2)$ (roughly) and $O(N)$. Here, we implement a second order SMO algorithm. Inference in SOAR-SVM achieved by minimizing the ϵ-insensitive loss w.r.t. the output vector $\mathbf{y}$, *c.f.* (15), where

$$\begin{aligned} f_{\text{svr}}^j = &\frac{1}{C_1^j}\sum_{i=1}^N(\widehat{\alpha}_i^j - \widehat{\beta}_i^j)k^\phi(\mathbf{x}, \mathbf{x}_i) + \\ &+ \frac{1}{C_2^j}\sum_{i=1}^N(\widehat{\alpha}_i^j - \widehat{\beta}_i^j)k^\varphi(\mathbf{y}^{-j}, \mathbf{y}_i^{-j}) + \widehat{b}^j \end{aligned} \tag{20}$$

References

[1] A. Agarwal and B. Triggs. Recovering 3d human pose from monocular images. *PAMI*, 2006.

[2] C. Atkeson, A. Moore, and S. Schaal. Locally weighted learning. *AI Review*, 11:11–73, 1997.

[3] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *PAMI*, 24(4):509–522, 2002.

[4] M. Blaschko and C. Lampert. Learning to localize objects with structured output regression. In *ECCV*, 2008.

[5] L. Bo and C. Sminchisescu. Twin Gaussian Processes for Structured Prediction. *IJCV*, 2009. Special Issue on Evaluation of Articulated Human Motion and Pose Estimation.

[6] L. Bo, C. Sminchisescu, A. Kanaujia, and D. Metaxas. Fast Algorithms for Large Scale Conditional 3D Prediction. In *CVPR*, 2008.

[7] R. D. Cook. *Regression Graphics*. Wiley Inter-Science, 1988.

[8] C. Cortes, M. Mohri, and J. Weston. A general regression technique for learning transductions. In *ICML*, pages 153–160, 2005.

[9] R. Fan, P. Chen, and C. Lin. Working set selection using second order information for training support vector machines. *JMLR*, 6, 2005.

[10] D. Heckerman, D. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *JMLR*, 1:49–75, 2000.

[11] M. Kim and V. Pavlovic. Dimensionality reduction using covariance operator inverse regression. In *CVPR*, 2008.

[12] S. Kumar and M. Hebert. Discriminative fields for modeling spatial dependencies in natural images. In *NIPS*, 2003.

[13] J. Lafferty, A. Mccallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001.

[14] C. Micchelli and M. Pontil. On learning vector-valued functions. *Neural Computation*, 17(1):177–204, 2005.

[15] R. Navaratnam, A. Fitzgibbon, and R. Cipolla. The Joint Manifold Model for Semi-supervised Multi-valued Regression. In *ICCV*, 2007.

[16] J. Nilsson, F. Sha, and M. I. Jordan. Regression on Manifolds Using Kernel Dimensionality Reduction. In *ICML*, 2007.

[17] R. Poppe. Evaluating example-based human pose estimation: Experiments on humaneva sets. In *HumanEva Workshop CVPR*, 2007.

[18] R. Rosales and S. Sclaroff. Learning Body Pose Via Specialized Maps. In *NIPS*, 2002.

[19] C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *ICML*, 1998.

[20] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *CVPR*, 2003.

[21] L. Sigal and M. Black. Humaneva: Synchronized video and motion capture dataset for evaluation of articulated human motion. Technical report, 2006.

[22] L. Sigal and M. Black. Predicting 3d people from 2d pictures. In *AMDO*, 2006.

[23] C. Sminchisescu, A. Kanaujia, Z. Li, and D. Metaxas. Conditional Visual Tracking in Kernel Space. In *NIPS*, 2005.

[24] C. Sminchisescu, A. Kanaujia, and D. Metaxas. Learning Joint Top-down and Bottom-up Processes for 3D Visual Inference. In *CVPR*, 2006.

[25] A. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.

[26] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 6:1453–1484, 2005.

[27] J. Weston, O. Chapelle, A. Elisseeff, B. Schölkopf, and V. Vapnik. Kernel dependency estimation. In *NIPS*, pages 873–880, 2002.

[28] J. Weston, B. Schölkopf, O. Bousquet, T. Mann, and W. Noble. Joint kernel maps. In *LNCS*, 2005.