Mach Learn (2018) 107:1229–1256
https://doi.org/10.1007/s10994-018-5698-0

Output Fisher embedding regression

Moussab Djerrab[1] · **Alexandre Garcia**[1] · **Maxime Sangnier**[2] · **Florence d'Alché-Buc**[1]

Received: 13 February 2017 / Accepted: 13 February 2018 / Published online: 17 May 2018

Abstract We investigate the use of Fisher vector representations in the output space in the context of structured and multiple output prediction. A novel, general and versatile method called *output Fisher embedding regression* is introduced. Based on a probabilistic modeling of training output data and the minimization of a Fisher loss, it requires to solve a pre-image problem in the prediction phase. For Gaussian Mixture Models and State-Space Models, we show that the pre-image problem enjoys a closed-form solution with an appropriate choice of the embedding. Numerical experiments on a wide variety of tasks (time series prediction, multi-output regression and multi-class classification) highlight the relevance of the approach for learning under limited supervision like learning with a handful of data per label and weakly supervised learning.

Keywords Fisher vector · Structured output prediction · Output kernel regression · Small data regime · Weak supervision

Editors: Kurt Driessens, Dragi Kocev, Marko Robnik-Šikonja, Myra Spiliopoulou.

✉ Moussab Djerrab
mdjerrab@enst.fr

Alexandre Garcia
algarcia@enst.fr

Maxime Sangnier
maxime.sangnier@upmc.fr

Florence d'Alché-Buc
fdalche@enst.fr

[1] Télécom ParisTech, Université Paris-Saclay, 75013 Paris, France

[2] UPMC Univ Paris 06, CNRS, Sorbonne Universités, 75005 Paris, France

1 Introduction

Recent years have witnessed an explosion of interest for predicting outputs with some structure, whether explicit (Bakir et al. 2007; Nowozin and Lampert 2011) or implicit (Vinyals et al. 2015; Lebret et al. 2015), in various areas such as computer vision, natural language processing and bioinformatics. It is generally acknowledged that learning to predict complex outputs raises important difficulties: (i) the output space does not enjoy a vector space structure, (ii) the output structure has to be taken into account in the model itself and in an appropriate loss function, (iii) prediction and learning phases are usually very expensive, (iv) the supervision is a demanding job and data are expensive to label, (v) as a consequence, training data might be partially or weakly labeled.

Regarding the two first issues, significant progress has been made through the abundant literature of structured output prediction. The so-called energy-based learning methods (Lafferty et al. 2001; Tsochantaridis et al. 2005; LeCun and Huang 2005; Chen et al. 2015) build a score function on pairs of input/output and make a prediction by extracting the output that obtains the best score. Interestingly, some of these approaches have been extended to also take into account the hidden structure that links inputs to the outputs (see for instance Yu and Joachims 2009; Zhu et al. 2010). Regarding point (iv), an extension of structured output learning with latent output variable has also been developed in Vezhnevets et al. (2012) to address weakly supervised learning in the context of semantic segmentation of images.

However, although very elegant, these methods are very expensive both at the prediction and the learning stage.

Another line of research, mainly represented by the family of Output Kernel Regression (OKR) tools (Cortes et al. 2005; Geurts et al. 2006; Brouard et al. 2011; Kadri et al. 2013; Brouard et al. 2016) but also close approaches based on an output distance (Kocev et al. 2013; Pugelj and Džeroski 2011), has been explored to avoid the cost of maximizing a score for each prediction. In the case of OKR methods, the structured output prediction problem is converted into a regression problem with an output feature map associated to a kernel. Thus, learning does not require to solve the prediction problem in the original output space and therefore is less demanding in terms of computational load. Yet, as in energy-based methods, the prediction stage is expensive: a solution has to be found in the original output space (Kadri et al. 2013) and this is generally addressed by solving a pre-image problem (Honeine and Richard 2011). A close approach emphasizing the role of the output feature map has also been very recently studied under the angle of regularization in Ciliberto et al. (2016).

Other works have been developed around the prediction of unstructured or semi-structured outputs such as phrases and sentences. They mainly concern automatic captioning in images or videos for which the most relevant methods combine discriminant approaches and generative models in a single architecture (Vinyals et al. 2015) or hierarchically (Lebret et al. 2015).

Now regarding the two last issues, (iv) and (v), there exists many fields of application where labeling requires experts and would benefit from methods that are able to cope with less training data. Video Labeling (Ponce-López et al. 2016) and Drug Activity Prediction (Su et al. 2010) are examples of such involved tasks. In this work, we take inspiration from the field of Structured Output Prediction to define a class of methods able to take into account explicit as well as implicit structure of output data in supervised tasks. We expect that accounting for hidden structure of the output does help when learning with a handful of labeled data, a small data regime often called *one-shot learning* (Fei-Fei et al. 2006) in case of classification tasks. We propose to address the problem of complex output prediction by learning the

relationship between an input and an output, the latter seen as a realization of a generative probabilistic model. For this purpose, we introduce a novel, general and versatile method called *output Fisher embedding regression* (OFER) based on the minimization of a surrogate loss based on Fisher Embedding and a closed-form solution for the corresponding pre-image problem. OFER relies on three main steps. First, a generative parametric probabilistic model of the training outputs is estimated and each training output is embedded into a vector space using the well-known Fisher vector, i.e. the gradient in the parameters of the generative model introduced in the seminal work of Amari (1998). Specifically, we use a well chosen linear transformation of the Fisher vector as the new target and we call it, Output Fisher Embedding. The training dataset is therefore transformed and in the second step, a vector-valued function is learned to predict the Output Fisher Embedding from the input. Eventually, to make a prediction in the original output space, one has to solve a pre-image problem at the third step. By electing an appropriate linear transformation of the Fisher vector, we show that the pre-image problem admits a closed-form solution in the case of Gaussian Mixture models while the Fisher Gaussian State-Space models enjoy a recursive solution based on Viterbi algorithms. Therefore, for outputs that can be seen as realizations of one of these two models, prediction is not expensive and point (iii) is no more an issue. Moreover, when using a Fisher embedding of a mixture model, a single training instance brings information about the prediction of outputs belonging to the same mixture component. This makes the approach well adapted to learning from an handful of training data, a variant of one-shot learning (Fei-Fei et al. 2006). Knowing how to learn under this "small data regime", is especially useful in application fields characterized by point (iv), when supervision is expensive. Eventually, the interpretability of each term of the Fisher embedding in the case of Gaussian mixture model also opens the door to a variant of weakly supervised learning supervision. To our knowledge, Output Fisher Embedding is the first use of Fisher vectors in output embeddings whilst Fisher kernels (inner product of Fisher vectors) (Jaakkola and Haussler 1998; Hofmann 2000; Siolas and d'Alché-Buc 2002; Perronnin et al. 2010) have been widely exploited to handle input data.

The paper is structured as follows. In Sect. 2, we introduce the OFER approach in the case of full supervision. Section 3 introduces Output Fisher Embeddings for Gaussian Mixture models and State-Space Models while Sect. 4 presents the solutions of the corresponding pre-image problems. Section 5 discusses the supervised learning problem and introduces a scenario for weakly supervised learning. Section 6 presents an extended experimental study on synthetic datasets and on a large variety of tasks with 5 different real datasets. Conclusion and perspectives are drawn in Sect. 7. The Appendix contain additional experimental results and an empirical study of computation times for all tested methods in all tasks.

2 Output Fisher embedding regression

2.1 Brief reminder about Fisher scores and kernels

Fisher vectors allow one to encode any realization $z \in \mathbb{R}^d$ of a parametric probabilistic model as a feature vector. Given a model p_θ of parameter vector θ, the Fisher vector is defined as:

$$\phi_{Fisher}(z) = \nabla_\theta \log p_\theta(z).$$

A Fisher vector encodes in each of its coordinates the contribution of each parameter to the value of the probability density on z, $p_\theta(z)$. We call the function $\phi_{Fisher} : \mathcal{Y} \mapsto \mathbb{R}^m$,

the *Fisher feature map*. In supervised learning, Fisher vectors have been associated to kernel machines with the so-called Fisher kernel to improve the encoding of the input characteristics:

$$k_{Fisher}(z, z') = \phi_{Fisher}(z)^T I^{-1} \phi_{Fisher}(z'),$$

where $I = \mathbb{E}_{z\sim p_\theta}[\phi_{Fisher}(z)\phi_{Fisher}(z)^T]$ is the Fisher information matrix, i.e. the covariance matrix of the Fisher score. Referring to the work by Amari in differential geometry and the idea of natural gradient (Amari 1998), k_{Fisher} is indeed the proper inner product to compare two samples taking into account the manifold of the parameter space of their probability distribution. In practice, I^{-1} is either empirically estimated from the training data or even omitted to give rise to the naïve Fisher kernel: $k_{Fisher}(z, z') = \phi_{Fisher}(z)^T \phi_{Fisher}(z')$. Support Vector Machines (SVM) based on (naïve) Fisher kernels have proved to provide accurate classifications in many fields such as remote protein homologies detection (Hou et al. 2003), document categorization (Hofmann 2000; Siolas and d'Alché-Buc 2002) or more recently image classification (Perronnin et al. 2010; Oneata et al. 2013). A recent work (Sydorov et al. 2014) has also bridged Fisher Kernel SVM and Convolutional Neural Networks, emphasizing the similarity between the two approaches for image classification. In this work, we propose a completely different use of the Fisher vectors in order to embed outputs with some structure in an appropriate feature space.

2.2 Regression with output Fisher embeddings

Denote $\mathcal{X}$ (resp. $\mathcal{Y}$) the input set (resp. output set) of a prediction problem. Assume that both $\mathcal{X}$ and $\mathcal{Y}$ are sample spaces. Let us call $\mathcal{S}_\ell = \{(x_i, y_i), i = 1, \ldots, \ell\}$, the training set. Inspired by the framework of Output Kernel Regression (Cortes et al. 2005; Geurts et al. 2006), Input Output Kernel Regression (Brouard et al. 2016) and the recent work of Ciliberto et al. (2016), Output Fisher Embedding Regression (OFER) aims at encoding the output data into a vector space where the inner product is relevant, helping the learning task at hand. This transformation of the output variable yields a new surrogate loss, able to take into account the inherent structure of the output.

The method, depicted in Fig. 1, relies on three steps:

1. Defining an output feature map, called here, Output Fisher embedding, $\phi^M_{Fisher} : \mathcal{Y} \to \mathbb{R}^p$, indexed by a matrix M of size $(m \times p)$ defined as: $\phi^M_{Fisher}(y) = M\phi_{Fisher}(y) = M\nabla_{\hat{\theta}} \log p_{\hat{\theta}}(y)$, where:

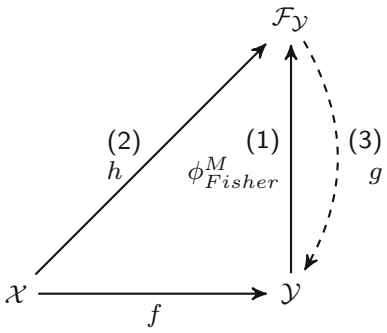

Fig. 1 Steps of output Fisher embedding regression

- $p_\theta(y)$ is the probability density of a parametric probabilistic model of parameter $\theta \in \mathbb{R}^m$ and $\hat{\theta}$ is an estimate of θ obtained from the training output set $\mathcal{Y}_\ell = \{y_1, \ldots, y_\ell\}$. Note that in some cases, a larger dataset including $\mathcal{Y}_\ell$ may be used instead of $\mathcal{Y}_\ell$.
- An appropriate choice of the linear transformation M allows one to simplify the pre-image problem.

2. Training a learning algorithm on a class $\mathcal{H}$ of functions $h : \mathcal{X} \to \mathbb{R}^p$ and dataset $\mathcal{S}_\ell$ to approximate the relationship between $x \in \mathcal{X}$ and $\phi^M_{Fisher}(y)$ by minimizing

$$\lambda_0 \Omega(h) + \frac{1}{2\ell} \sum_{i=1}^{\ell} L^M_{Fisher}(y_i, h(x_i)),$$

where $L^M_{Fisher} : \mathcal{Y} \times \mathbb{R}^p \to \mathbb{R}^+$ is a surrogate loss defined as:

$L^M_{Fisher}(y_i, h(x_i)) = \|\phi^M_{Fisher}(y_i) - h(x_i)\|^2$.

3. Given x, making a prediction in the original output space $\mathcal{Y}$ by solving a pre-image problem: $y^* \in \arg\min_{y \in \mathcal{Y}} L^M_{Fisher}(y, h(x))$.

OFER can be seen as an instance of the general framework of Output Kernel Regression with the output Fisher kernel k_{Fisher}. Let us recall that in Output Kernel Regression (OKR), one makes use of the kernel trick in the output space, allowing for predicting complex outputs with squared loss applied in output feature space. If h is the function to be learned, for a given pair (x, y), the local square loss $L_{OKR}(y, h(x)) = \|\phi(y) - h(x)\|^2_{\mathcal{F}_Y}$ can be computed using the kernel $k(y, y') = \langle \phi(y), \phi(y') \rangle_{\mathcal{F}_Y}$ as soon as $h(x)$ writes in terms of $\phi(y_i), i \in \{1, \ldots, n\}$. This is the case for the following methods: (i) Kernel Dependency Estimation KDE (Cortes et al. 2005) that is the first example of how to use two scalar-valued kernels, one defined on the input space and the other on the output space, (ii) Input Output Kernel Regression (Brouard et al. 2011, 2016) that can be seen as a generalization of KDE where the input kernel is an operator-valued kernel and the output kernel a scalar-valued one, (iii) all Output Kernel Tree-based approaches such as OK3 (Geurts et al. 2006) and OKBoost (Geurts et al. 2007) that extend tree-based methods for regression to kernelized outputs. However all these methods require to solve a difficult pre-image problem.

In contrast, OFER directly exploits an explicit and finite dimensional feature map for the output kernel, the Output Fisher embedding. The vector-valued function h in Step 2 can be computed using any learning method devoted to vector-valued functions or by any learning method devoted to scalar-valued functions using parallel coordinate-wise computations as opposed to using the kernel trick in the output space.

Now, the size of matrix M (in particular the value p), involved in the surrogate Fisher loss, directly influences the time complexity of the learning phase. Strictly speaking, the Fisher vector is a very sparse representation that is unnecessary long in the context of prediction. In this paper, we show that the case $M = I$ yielding a classic pre-image problem with no analytical solution can be replaced by a proper choice of $M \neq I$ in order to obtain a closed-form solution, easier to compute (see Sect. 4) and with better empirical performance.

3 Fisher embeddings

In this paper, we illustrate the principle of output Fisher embedding regression with two generative models: the Gaussian Mixture Model, to take into account the hidden components of random vectors and the Gaussian State Space Model, to represent hidden processes in time

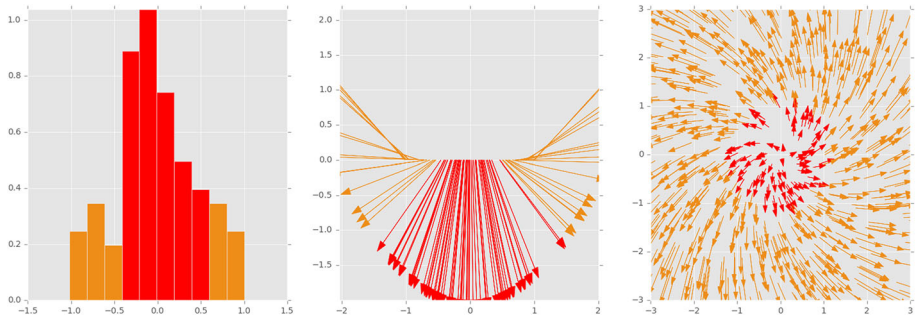

Fig. 2 Gaussian Fisher embeddings for 500 samples of $\mathcal{N}(0, 0.5)$ and of $\mathcal{N}\left(\binom{0}{0}, 5 \times Id\right)$. The left hand side figure represent the histogram of a 1D Gaussian distribution and the central figure is a plot of the corresponding Fisher vectors for each sample. The right figure is the same plot for a 2D Gaussian. In both figures, the color red stresses the points that are in the central interval mode $[\pm\sigma]$ (Color figure online)

series. In the following, we assume that the generic parameter θ is known. A building block of these models is the Gaussian vector for which we briefly present the Fisher feature map.

3.1 Gaussian vector

Let us assume that $Y \sim \mathcal{N}(\mu, \Sigma)$ and $\mu \in \mathbb{R}^d$ and Σ is a $d \times d$ positive definite matrix, with $d \in \mathbb{N}^*$. Denote $p_\theta(y) = \frac{1}{\sqrt{2\pi|\Sigma|^{d/2}}} e^{-\frac{1}{2}(y-\mu)^T \Sigma^{-1}(y-\mu)}$ the probability density. Then, the Fisher feature map takes the following form:
$\forall y \in \mathcal{Y}$,

$$\phi_{Fisher}(y) = \begin{pmatrix} \phi_\mu(y) \\ \phi_\Sigma(y) \end{pmatrix} = \begin{pmatrix} \Sigma^{-1}(y-\mu) \\ vec\left(-\Sigma^{-1} + \phi_\mu(y)\phi_\mu(y)^T\right) \end{pmatrix}. \tag{1}$$

Let us notice that $\phi_\Sigma(y)$ can be directly derived from the value of $\phi_\mu(y)$. Figure 2 intends to provide a geometric intuition of the Fisher embedding.

3.2 Gaussian mixture model

Let us now consider a Gaussian Mixture Model in $\mathbb{R}^d$ with the following probability density p_θ:

$$p_\theta(y) = \sum_{i=1}^{C} \pi_i \, p_{\theta_i}(y),$$

where each component has a Gaussian density: $p_{\theta_i}(y) = \mathcal{N}(\mu_i, \Sigma_i)$, $\theta = ((\pi_j, \theta_j), j = 1, \ldots, C)$, Hence, the Fisher vector writes as:

$$\phi_{Fisher}(y) = \begin{pmatrix} \varphi_\pi(y) \\ \varphi_\mu(y) \\ \varphi_\Sigma(y) \end{pmatrix},$$

where $\varphi_\pi(y) \in \mathbb{R}^C$, $\varphi_\mu(y) \in \mathbb{R}^d$ and $\varphi_\Sigma(y) \in \mathbb{R}^{d^2}$. We have:

$$\begin{aligned} \varphi_\pi(y) &= \nabla_\pi(\log p_\theta(y)) \\ &= \left(\frac{p_{\theta_1}(y)}{p_\theta(y)}, \ldots, \frac{p_{\theta_C}(y)}{p_\theta(y)}\right)^T, \end{aligned}$$

and, using the previous notation for the Gaussian Fisher embedding:

$$\varphi_{\mu,j}(y) = \nabla_{\mu_j}(\log p_\theta(y)) = \pi_j \frac{p_{\theta_j}(y)}{p_\theta(y)} \phi_{\mu_j}(y),\ \forall j = 1, \ldots, C$$
$$\varphi_{\Sigma,j}(y) = vec\left(\pi_j \frac{p_{\theta_j}(y)}{p_\theta(y)} \phi_{\Sigma_j}(y)\right),\ \forall j = 1, \ldots, C.$$

The jth coefficient of vector $\varphi_\pi(y)$ is non-negative and represents some contribution of y with respect to the jth component. In the case where y does not contribute to this component at all, the value of the derivative would be equal to zero. As for the other parts of the Fisher vector, only the derivatives related to the component that most likely have generated y have an absolute value far from 0. The Fisher vector is therefore generally very sparse for GMM.

3.3 Gaussian state-space model

Gaussian State-Space Models allow to represent partially observed stochastic continuous time processes. They are defined by the two following equations:

$$m_t = Am_{t-1} + Q\eta_t \quad \text{(State equation)},$$
$$z_t = Cm_t + Ru_t \quad \text{(Observation equation)},$$

where $z_t \in \mathbb{R}^p$ is the observation at time t and $m_t \in \mathbb{R}^d$ is the hidden state at time t. $A \in \mathbb{R}^{d\times d}$, $Q \in \mathbb{R}^{d\times d}$, $C \in \mathbb{R}^{p\times d}$ and $R \in \mathbb{R}^{p\times p}$ are matrices to be estimated. Noise η_t (resp. u_t) is assumed to be independently and identically distributed according $\mathcal{N}(0, I_d)$ (resp. $\mathcal{N}(0, I_p)$). At each time step, the observations and the hidden states are thus distributed as follows:

$$m_t \sim \mathcal{N}(Am_{t-1}, Q),$$
$$z_t \sim \mathcal{N}(Cm_t, R).$$

From these assumptions, we derive the Fisher vector for $y^T = (z_1, \ldots, z_T, m_1, \ldots, m_T)$:

$$\phi_{Fisher}(y)^T = (\phi_{0,1}(y), \phi_{1,1}(y), \ldots, \phi_{1,T}(y)\phi_{2,1}(y), \ldots, \phi_{2,T}(y)), \tag{2}$$

where $\phi_{0,1}(y) = \nabla_{\mu,\Sigma} \log p(m_1)$ is the Fisher vector corresponding to the initial hidden state, $\phi_{1,t}(y) = \nabla_{(Am_{t-1},Q)} log(p(m_t|m_{t-1})), t = 2, \ldots, T$ is the Fisher vector corresponding to the transition probability density between hidden states and $\phi_{2,t}(y) = \nabla_{(Cm_{t-1},R)} log(p(z_t|m_t))$, for $t = 2, \ldots, T$, is the Fisher vector for the emission probability density.

4 Prediction by solving the pre-image problem

Now let us assume that θ is known and function h has been learned. Let us consider (x, y^{true}) a pair of input/output. In order to predict the value of y^{true}, we want to find y^* such that:

$$\text{Find } y^* \in \arg\min_{y\in\mathcal{Y}} \|M\phi_{Fisher}(y) - h(x)\|^2. \tag{3}$$

In the following, we study the pre-image problem and show sufficient conditions on the input representation $h(x)$ such that the minimization problem in Eq. 3 admits a unique closed-form solution in the case of Gaussian model and Gaussian Mixture models and can be obtained

recursively in the case of a Gaussian State-Space Model. As a preliminary step, we study the simple case of a Gaussian model which is not interested itself but is a building block for the two other pre-image problems.

4.1 Pre-image for the Gaussian Fisher embedding

In the case of the Gaussian Fisher embedding, we have : $h(x) = \begin{pmatrix} h_1(x) \\ h_2(x) \end{pmatrix}$ with $h_1(x) \in \mathbb{R}^d$ and $h_2(x) \in \mathbb{R}^{d^2}$. From this representation $h(x)$, we want to make a prediction y^* of y^{true} by solving the pre-image problem for the full Fisher embedding:

$$\hat{y} \in \underset{y \in \mathcal{Y}}{argmin}\ L^I_{Fisher}(y, h(x)), \tag{4}$$

where $L^I_{Fisher}(y, h(x)) = \|\Sigma^{-1}(y-\mu) - h_1(x)\|^2 + \cdots + \|vec(\Sigma^{-1}(y-\mu)(y-\mu)^T\Sigma^{-1} - \Sigma^{-1}) - h_2(x)\|^2$.

If the prediction $h(x)$ was perfect, we would have $h(x) = \phi_{Fisher}(y^{true})$, then the minimum $L^I_{Fisher}(\hat{y}, h(x)) = 0$ would be attained by the closed-form solution $\hat{y} = \sigma h_1(x) + \mu$ and $\hat{y}$ would be equal to the exact solution y^{true}. However in practice, the prediction $h(x)$ is not perfect and the previous equation is not satisfied. Instead we have: $\exists(\epsilon_1, \epsilon_2) \in \mathbb{R}^d \times \mathbb{R}^{d^2}$, a pair of non-null vectors, such that: $h_1(x) = \varphi_\mu(y^{true}) + \epsilon_1$ and $h_2(x) = \varphi_\Sigma(y^{true}) + \epsilon_2$. Therefore, in general, the minimum of the loss function is greater than 0 and moreover, the loss is non non-convex. The best we can do to minimize i,t is to apply a gradient descent method to find a local minimum. The final prediction $\hat{y}$ is thus affected by two sources of error here: the $\epsilon = (\epsilon_1, \epsilon_2)$ error due to the lack of accuracy of h and the error driven by the approximate solution of this pre-image problem.

Another approach consists in taking advantage of the re-parameterization of ϕ_{Fisher} we pointed out in Sect. 3.1. We forget about $\varphi_\Sigma(y)$ that we can reconstruct from a proxy of $\varphi_\mu(y)$ if needed and we now learn $g_1 : \mathcal{X} \to \mathbb{R}^d$ by solving:

$$g_1 = \arg\min_f \lambda_0 \Omega(f) + \|\phi^M_{Fisher}(y_i) - f(x_i)\|^2 \tag{5}$$

with

$$\phi^M_{Fisher}(y) = \varphi_\mu(y) = \Sigma^{-1}(y - \mu),$$

and M is the matrix that projects the complete $\phi_{Fisher}(y)$ on the d first coordinates. In this case, to solve the pre-image problem for a given x, we only have to find the solution of the following problem :

$$y^* \in \underset{y \in \mathcal{Y}}{argmin}\ \|\phi^M_{Fisher}(y) - g_1(x)\|^2 \tag{6}$$

that admits a closed-form solution:

$$y^* = \Sigma g_1(x) + \mu. \tag{7}$$

If one assumes that $g_1(x) = \varphi_\mu(y^{true}) + \epsilon'_1$, the solution y^* is thus only affected by the prediction error ϵ'_1. More precisely, we have:

$$\|y^{true} - y^*\| = \|\Sigma\epsilon'_1\| \tag{8}$$

We expect that the error of the reduced Fisher embedding, $\Sigma\epsilon'_1$, is easier to control than the two other errors appearing in the Full Fisher embedding pre-image problem and for which no analytical form is available. In Sect. 6.2, we show empirical comparisons in Table 1 for

simulated data to illustrate this claim. The reduced variant of Fisher Embedding is chosen in the real experiments.

4.2 Pre-image for the Gaussian mixture model Fisher embedding

Assume a Gaussian Mixture model with C components in $\mathbb{R}^d$. Again, for any input vector x we denote $h(x) = (h_1(x), h_2(x), h_3(x))$ where $h_1(x)$ is vector of sizea $d\text{-}dimensional$ vector, $h_2(x)$ is a Cd block vector that can be decomposed into C vectors $h_{2,j}(x)$, $j = 1, \ldots, C$ of dimension d and v_3, another block vector that can be decomposed into C vectors $h_{3,j}(x)$, $j = 1, \ldots, C$ of dimension d^2. In the case $M = I$, the pre-image for the Gaussian Mixture Model Fisher embedding consists in minimizing the following expression with respect to $y \in \mathcal{Y}$:

$$L^I_{Fisher}(y, h(x)) = \|\varphi_\pi(y) - h_1(x)\|^2 + \sum_{j=1}^{C} \|\varphi_{\mu,j}(y) - h_{2,j}(x)\|^2 + \sum_{j=1}^{C} \|\varphi_{\Sigma,j}(y) - h_{3,j}(x)\|^2. \tag{9}$$

Similarly to the Gaussian Fisher Embedding, we notice that, when omitting the third term, we get a closed-form-solution problem. Indeed:
$\|\varphi_\pi(y^*) - h_1(x)\|^2 + \sum_{j=1}^{C} \|\varphi_{\mu,j}(y^*) - h_{2,j}(x)\|^2 = 0,$
implies that: $\begin{cases} \varphi_\pi(y^*) = h_1(x) \\ \forall j = 1, \ldots, C, \varphi_{\mu,j}(y^*) = h_{2,j}(x) \end{cases}$
which leads to the following system of equations:

$$\begin{cases} \varphi_\pi(y^*) = h_1(x) \\ \sum_{j=1}^{C} \varphi_{\mu,j}(y^*) = \sum_{j=1}^{C} h_{2,j}(x) \end{cases}.$$

According the definition of φ_π, φ_μ and Eq. 1, we have:

$$\begin{cases} \forall j = 1, \ldots, C, \frac{p_{\theta_j}(y^*)}{p_\theta(y^*)} = h_{1,j}(x) \\ \sum_{j=1}^{C} \pi_j \frac{p_{\theta_j}(y^*)}{p_\theta(y^*)} \Sigma_j^{-1}(y^* - \mu_j) = \sum_{j=1}^{C} h_{2,j}(x) \end{cases},$$

with the following closed-form solution:

$$y^* = \left(\sum_{j=1}^{C} \pi_j h_{1,j}(x) \Sigma_j^{-1} \right)^{-1} \left(\sum_{j=1}^{C} h_{2,j}(x) + \left(\sum_{j=1}^{C} \pi_j h_{1,j}(x) \Sigma_j^{-1} \mu_j \right) \right). \tag{10}$$

To exploit the hidden structure of outputs data, we therefore propose to use the following (reduced) Fisher embedding for the GMM:

$$\phi^M_{Fisher}(y) = (\varphi_\pi(y), \sum_{j=1}^{C} \varphi_{\mu,j}(y))^T = M\phi_{Fisher}(y),$$

with the block matrix $M = \begin{pmatrix} M_1 & 0 & 0 \\ 0 & M_2 & 0 \\ 0 & 0 & 0 \end{pmatrix}$, where $M_1 = I_C$ and M_2, a block matrix of size $dC \times d$ is defined as: $M_2 = (I_d I_d \ldots I_d)$. This embedding keeps on taking into account the mixture structure while drastically reducing the dimension of the outputs from $C(1+d+d^2)$

to $(C+d)$ and allowing for a closed-form pre-image. As for the Gaussian Fisher Embedding, the prediction error $\|y^{true} - y^*\|$ associated to y^* only comes from the learning phase while the (local) minimizer of the full Fisher pre-image loss in Eq. 9 will convey two sources of error: the one induced by the learning of h and the one resulting from the approximate resolution of the full pre-image problem itself. This remark is illustrated in Sect. 6.2 where Tables 3 and 4 present an empirical comparison of performance of OFER with the full Fisher embedding for GMM and the reduced Fisher embedding on toy datasets.

4.3 Pre-image for the Gaussian state space model Fisher embedding

The pre-image problem for the Gaussian State Space Model embedding benefits from the previous solution and is sequentially solved over time. The solution at step $t-1$ is leveraged to solve the problem at step t:

$$L_m(m_t) = \|M_1\phi_{1,t}(m_t) - h_{1,t}(x)\|^2$$
$$L_z(z_t) = \|M_2\phi_{2,t}(z_t) - h_{2,t}(x)\|^2,$$

where the predicted vectors is $h(x) = (h_0(x), h_{1,1}(x), \ldots, h_{1,T}(x), h_{2,1}(x), \ldots, h_{2,T}(x))$. As a matter of simplification we chose a reduced form ϕ^M_{Fisher} with $M = diag(M_0, M_1, \ldots, M_1, M_2 \ldots, M_2)$. For each matrix M_1 and M_2, as done in the Gaussian case, we choose a projection on the first d , and respectively V, dimensions:

$$M_1\phi_{1,t}(m_t) = \frac{\partial log(p(m_t|m_{t-1}))}{\partial Am_{t-1}} = Q^{-1}(m_t - Am_{t-1}), \tag{11}$$
$$M_2\phi_{2,t}(z_t) = \frac{\partial log(p(z_t|m_t)}{\partial Cm_t} = R^{-1}(z_t - Cm_t), \tag{12}$$

Hence, at each step, the pre-image problem becomes :

$$L_m(m_t) = \|Q^{-1}(m_t - A\hat{m}_{t-1}) - h_{1,t}(x)\|^2$$
$$L_z(z_t) = \|R^{-1}(z_t - C\hat{z}_{t-1}) - h_{2,t}(x)\|^2,$$

where $\hat{m}_t, \hat{z}_t$ represent the pre-image solutions at time t. For $\hat{m}_0$, we solve the first pre-image step depending on the probability distribution of the initial stats (Gaussian or GMM). Then, Algorithm 1 is applied to solve sequentially the following problem: $\underset{(m,z)}{argmin} \sum_{t=1}^{T}(\mathcal{L}_m(m_t) + L_z(z_t))$.

Algorithm 1 Main Pre-image for GSSM

Require: $h(x) = (h_0(x), h_{1,1}(x), \ldots, h_{1,T}(x), h_{2,1}(x), \ldots, h_{2,T}(x))$ and $\hat{m}_0$
for $t = 1$ **to** T **do**
$\hat{m}_t = A\hat{m}_{t-1} + Qh_{1,t}(x)$
$\hat{z}_t = C\hat{z}_t + Rh_{2,t}(x)$
end for

5 Learning to predict output Fisher embeddings

5.1 Supervised learning

Supervised learning of Output Fisher Embeddings (Step 2 of OFER) can be achieved by solving a ridge regression problem within a class of vector-valued functions, $\mathcal{H}$. Given λ_0, a positive scalar, we search:

$$\underset{h\in\mathcal{H}}{\operatorname{argmin}}\ \lambda_0\|h\|^2_{\mathcal{H}} + \frac{1}{2\ell}\sum_{i=1}^{\ell} L^M_{Fisher}(y_i, h(x_i)). \tag{13}$$

In this work, $\mathcal{H}$ was chosen as the Reproducing Kernel Hilbert Space associated to the matrix-valued kernel $K : \mathcal{X}\times\mathcal{X}\mapsto\mathcal{L}(\mathbb{R}^p)$. Matrix-valued kernels provide a generalization of the scalar-valued kernels (Micchelli and Pontil 2005; Álvarez et al. 2012) to build vector-valued functions. In this work, we chose $K(x, x') = I_p \times k(x, x')$ where k is the classic scalar-valued Gaussian kernel. This choice simply yields to p independent Kernel Ridge Regression models.

5.2 Learning in the small data regime with OFER-GMM

We would like to emphasize the relevance of our approach for learning with a handful of labeled data, a regime we call *small data regime* learning. As stated in Sect. 2.1, Output Fisher Embedding provides a richer information than the sole output. A training labeled example (x_i, y_i), converted into $(x_i, \phi^M_{Fisher}(y_i))$, conveys additional information about its structure. A simple example is the Fisher vector associated to a mixture model for encoding a vector y. Let us focus on $\varphi_\pi(y)$. The mixture component k to which y belongs with the highest probability corresponds to a high absolute value of the kth coordinate of the vector of $\varphi_\pi(y)$. Then, if two outputs y_i and y_j belong to the same component, learning to predict $\varphi_\pi(y_i)$ will also bring information about predicting $\varphi_\pi(y_j)$ and vice-versa. Therefore, our framework has appealing features for learning using only a handful of labeled examples per class if the target problem is classification or a small training data if the task is regression. This is empirically tested in the experimental section.

5.3 Weakly supervised learning in the case of OFER-GMM

We notice that, for a given y, each coordinate of the Fisher vector is interpretable as it reflects some structure of the output data. Taking the simple example of a mixture model, Output Fisher Embedding encodes the cluster-structure of the data since the first coordinates of the vector $\varphi_\pi(y)$ indicate the membership to each cluster. This remarks has inspired us a new scenario for weakly supervised learning. We imagine that we have access to two kinds of supervisors: expert supervisors who are able to label data in a precise way with their exact class and non-expert supervisors who are just able to associate one cluster to a given data. Note that the cluster has not even to be named, the content of clusters (for instance, images) can be presented to the non-expert supervisor. To recap, the scenario for weakly supervised learning writes as follows:

- A subsample $\{x_1, \ldots, x_\ell\}$ of the training input data set has been accurately labeled by an expert supervisor, providing the corresponding training sample $\mathcal{Y}_\ell = \{y_1, \ldots, y_\ell\}$.
- The training output sample $\mathcal{Y}_\ell$ is used to estimate the parameters of a (Gaussian) mixture model with C components.

– A matrix M of the form described in Sect. 4.2 is defined, yielding an interpretable Output Fisher embedding where the first C coefficients encode the membership of the given y to each of the C components of the mixture.
– The remaining training input data are presented to non-expert supervisors who are invited to indicate to which output component the presented input data can be associated, and therefore, only fulfill the first C coefficients of the Fisher vector instead of providing the full corresponding output object. From this weak supervision, one can define an additional training set with weak labels. To describe this set we follow the notations of Sect. 4.2 and introduce the matrix $A = \left[\begin{array}{c|c} I_C & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0}_{p-C} \end{array}\right]$, and $A^c = I_p - A$, its complement. Then the additional weak training set $\mathcal{W}_w$ is defined as:

$$\mathcal{W}_w = \left\{ \left(x_i, A\phi^M_{Fisher}(y_i), i = \ell+1, \ldots, \ell+w \right\} \right.,$$

where each $y_i, i = \ell+1, \ldots, \ell+w$ is supposed to be the true unknown output associated with each $x_i, i = \ell+1, \ldots, \ell+w$.

The learning task is now enriched in such ways that both kinds of examples, fully labeled and weakly labeled examples can be used during the training phase. Interestingly for the implementation, the following equality holds:

$$\begin{aligned} L^M_{Fisher}(y,x) &= \|\phi^M_{Fisher}(y) - h(x)\|^2 = \|(A+A^c)\phi^M_{Fisher}(y) - (A+A^c)h(x)\|^2 \\ &= \|A\phi^M_{Fisher}(y) - Ah(x)\|^2 + \|A^c\phi^M_{Fisher}(y) - A^c h(x)\|^2 \\ &= L^M_{Fisher,A}(y, h(x)) + L^M_{Fisher,A^c}(y, h(x)). \end{aligned}$$

Thus, the new learning task expresses as:

$$\begin{aligned} \underset{h\in\mathcal{H}}{\operatorname{argmin}} \frac{1}{2\ell}\sum_{i=1}^{\ell} L^M_{Fisher,A}(y_i, h(x_i)) + \lambda_2 \sum_{i=\ell+1}^{\ell+w} L^M_{Fisher,A}(y_i, h(x_i)) \\ + \frac{1}{2\ell}\sum_{i=1}^{\ell} L^M_{Fisher,A^c}(y_i, h(x_i)) + \lambda_0 \|h\|^2_{\mathcal{H}}. \end{aligned} \tag{14}$$

6 Numerical results

6.1 Context of the empirical study

In this section, we first explore the relevance of output fisher embedding regression on synthetic datasets and then, we investigate its behaviour on real datasets in three contexts: *fully supervised learning*, *small data regime learning* and *weakly supervised learning*.

To show the versatility of our framework, we study and compare OFER in a wide variety of real tasks. A structured output prediction task where the goal is to predict a short time series from an input text related to Kogan et al. (2009) is first explored. Then, OFER is tested on supervised learning tasks without explicit structure. Numerical results are provided on two multiple output regression tasks for which labels are usually difficult to obtain: a Video Score Prediction task extracted from Ponce-López et al. (2016) and a Drug Activity Prediction task presented in Su et al. (2010). Eventually we turn an image classification problem into a Word Prediction task in an appropriate semantic space and present numerical results on two

datasets, namely the *Caltech101*[1] and the AWA.[2] In all numerical results unless otherwise noted, train/test splits are generated 10 times and average performance with the empirical standard deviation are presented. We used bold font to point out the best performance in a row when it is statistically significant based on Student's t-tests with an α-risk equal to 10^{-3}. For sake of space, an empirical comparison of computation time in learning and test phases is presented and discussed in the Appendix.

6.1.1 Generative model estimation

Gaussian models are chosen isotropic in all experiments. The parameter θ is estimated using a procedure based on maximum likelihood criterion. More precisely, an Expectation-Maximization (EM) algorithm is applied to estimate the Gaussian mixture model while Kalman filtering/EM is used to estimate the State-Space Model.

6.1.2 OFER learning

In all experiments, the function h in Eq. 13 is learned using OFER implemented through Kernel Ridge Regression, called appropriately for multiple outputs in the case of fully and weakly supervised learning. The choice of kernel k in $K(x, x') = Ik(x, x')$ is linear by default if not precised. For purpose of comparison in regression and time series prediction, we used three baselines : multi-output Kernel Ridge Regression (m-KRR) and multi-output Random Forest for regression (m-RF) and variants of Input Output Kernel Regression (IOKR). As for the classification task, we used multiclass SVM, Multilayer perceptron (MLP) as well as IOKR. For each problem at hand, we precise which output kernel was used. IOKR is associated to a pre-image problem which is solved for each test data using a gradient descent implemented in the Open Source library scipy: https://www.scipy.org/. Apart from the synthetic dataset experiments, OFER is always used with the reduced Fisher Embedding explained in Sect. 4.

6.1.3 Implementation

All the codes are written in Python and call the appropriate functions of scikit-learn library[3] except for the Kalman filtering/EM procedure which relies on PyKalman.[4]

6.1.4 Time complexity

An empirical comparison of computation time in learning and test phases has been conducted for all real tasks and all the tested methods. For sake of space, it is presented and discussed in the Appendix.

6.2 Numerical results on synthetic datasets

We report numerical results on synthetic datasets in order to answer three questions:

1. Does OFER with reduced Fisher embedding perform better than OFER with full Fisher embedding in the Gaussian context?

[1] http://www.vision.caltech.edu/Image_Datasets/Caltech101/.

[2] http://attributes.kyb.tuebingen.mpg.de/.

[3] http://scikit-learn.org/.

[4] https://pykalman.github.io/.

Table 1 Average relative root mean squared errors (RRMSE) on test set obtained by OFER with the full Gaussian Fisher pre-image solution and the reduced Gaussian Fisher pre-image solution over 10 train/test splits of dataset $\mathcal{T}_1$

Average RRMSE on test set (mean ± std)			
Embedding/# Train	10	50	100
ϕ_{Fisher}	1.15 ± 0.02	1.01 ± 0.01	1.01 ± 0.01
ϕ^M_{Fisher}	$\mathbf{1.09 \pm 0.02}$	$\mathbf{0.89 \pm 0.004}$	$\mathbf{0.83 \pm 0.001}$

Table 2 Average training and test computation times for OFER-GMM with full and reduced Fisher embeddings over 10 train/test splits of dataset $\mathcal{D}_1$

		Computation time (s) of Table 1	
		Training time	Test time
ϕ_{Fisher}	10	$3.35 \times 10^{-4} \pm 7.01 \times 10^{-5}$	57.1 ± 14.7
	50	$8.20 \times 10^{-4} \pm 2.20 \times 10^{-4}$	80.4 ± 16.0
	100	$2.45 \times 10^{-3} \pm 5.75 \times 10^{-4}$	67.1 ± 21.7
ϕ^M_{Fisher}	10	$\mathbf{3.50 \times 10^{-4} \pm 8.74 \times 10^{-5}}$	$\mathbf{6.19 \times 10^{-7} \pm 5.74 \times 10^{-7}}$
	50	$\mathbf{7.60 \times 10^{-4} \pm 1.96 \times 10^{-4}}$	$\mathbf{1.19 \times 10^{-6} \pm 1.14 \times 10^{-6}}$
	100	$\mathbf{1.64 \times 10^{-3} \pm 4.06 \times 10^{-4}}$	$\mathbf{5.72 \times 10^{-7} \pm 5.22 \times 10^{-7}}$

2. Does OFER with reduced Fisher embedding perform better than OFER with full Fisher embedding in the GMM context?
3. How does OFER-GMM performance evolve with a growing number of training data and a growing number of clusters

We generated a synthetic dataset $\mathcal{D}_1$ by using a Gaussian mixture model (4 clusters) in $\mathbb{R}^{10}$ to get the outputs and by computing the projection of outputs by kernel PCA to obtain the corresponding input representation in a 5-dimensional space. When OFER is based on the full Fisher embedding, each pre-image problem is solved by applying a stochastic gradient descent algorithm.

Table 1 reports experiments on dataset $\mathcal{D}_1$ and shows that OFER with reduced Gaussian Fisher embedding leads to better performance than OFER with the full Gaussian embedding. We also see in Table 2 that the computation times in training and test phases differ. This is mostly due to the resolution of the pre-image problem. The pre-image with the reduced Fisher vector is both faster and more accurate. Similar conclusions can be drawn from Table 3 where OFER-GMM with reduced Fisher embedding exhibits better performance than OFER-GMM with full Fisher embedding.

For the last question we generated a dataset $\mathcal{D}_2$ similarly to $\mathcal{D}_1$ with output data generated in $\mathbb{R}^5$ from 10 clusters. Table 4 presents the Means Square Error obtained by OFER-GMM with $C = 1, 2, 3$ and using an increasing number of training samples taken from a total training set containing 50% of the dataset $\mathcal{D}_2$ and with a fixed test set containing the 50% remaining samples. Multiple KRR is used here as a baseline. The results are averaged over 10 different splits. First, if we consider Table 4 row by row, we see that OFER-GMM behaves consistently with a test error decreasing with the number of training data. Second, if we observe the table column by column, we see that the impact of a larger number of components in the mixture depends on the number of available training data. In regression, we need a sufficient number

Table 3 Relative root mean squared errors (RRMSE) obtained by OFER-GMM with the full Fisher pre-image solution and the reduced Gaussian Fisher pre-image solution on dataset $\mathcal{D}_1$ for 3 values of C=2,4,10

RRMSE on test sets ± std				Test time (s)	
# modes	# Train	ϕ_{Fisher}	ϕ^M_{Fisher}	ϕ_{Fisher}	ϕ^M_{Fisher}
2-GMM	10	1.17 ± 0.09	$\mathbf{1.13 \pm 0.11}$	124.01 ± 31.27	**0.014***
	50	1.14 ± 0.06	$\mathbf{1.03 \pm 0.002}$	80.54 ± 22.68	**0.015***
	100	1.18 ± 0.08	$\mathbf{1.03 \pm 0.002}$	83.30 ± 31.66	**0.016***
4-GMM	10	1.50 ± 0.19	1.66 ± 0.20	276.54 ± 84.17	**0.023***
	50	1.44 ± 0.08	$\mathbf{1.19 \pm 0.02}$	289.91 ± 90.78	**0.028***
	100	1.33 ± 0.10	$\mathbf{1.19 \pm 0.01}$	290.70 ± 27.07	**0.025***
10-GMM	10	1.84 ± 1.04	1.34 ± 0.02	1857.38 ± 232.84	**0.12***
	50	1.79 ± 1.23	1.59 ± 0.19	2193.76 ± 1336.08	**0.13***
	100	1.65 ± 0.61	$\mathbf{1.27 \pm 0.01}$	2494.62 ± 860.46	**0.13***

The last two columns report computation times in seconds, for both methods on the Test part
*Means that the standard deviation is of order 10^{-5}

of training data if one wants to take benefit from a more complex output model. It is also the case that low complexity GMM Fisher embedding brings a significant improvement when the training dataset are really small (Table 5).

Eventually, we present another set of numerical results on a simulated dataset is presented to show the performance of OFER on sequential outputs prediction. Times series of length 10 were generated using a Gaussian state space model. Each time series comes with both latent data and observed data. The parameters of the model have been chosen to keep steady-state time series. Once the dataset has been generated, we cut each time series into two parts. The first half plays the role of input data and the second half, the output data. As learning algorithms we have chosen the multi-output regression (M-KRR) and and the multiple output Random Forest (m-RF) and the Input Output Kernel Regression, with the global alignment kernel as the output kernel (IOKR2) as chosen in Table 6. Hence the goal is to measure the effect of OFER-GSSM on both algorithm to answer the prediction task. Hyper parameters have been selected through a 5-Fold cross-validation step. We express the results in terms of Relative Root Mean Squared Error (RRMSE) on the Latent space and the Observed data. Again we omit to present the standard deviations since they are smaller than 10^{-3}. Experiments have been repeated 10 times.

6.3 From text to time series

6.3.1 Task description

In finance, institutional actors provide legal reports such as the so-called "10-K forms" to evaluate the inner risk that a given company bears in the near future (usually 6 months ahead). Predicting the market risk indicator for a company from legal report is of high interest since it provides a mean to take into account the dependency of the market to unquantifiable objects through a numerical indicator. Let us recall the definition of the short-return log-volatility at time t for a given company j: $y_{t,j} = log\left(\sqrt{\sum_{i=1}^{\tau}(r_{t-i,j} - \bar{r}_j)}\right)$, where $r_{i,j} = \frac{p_{i,j} - p_{i-1,j}}{p_{i-1,j}}$ is the return at time i for an asset of a company j , and $p_{i,j}$ is the price of the asset. The output is the log-volatility over the time period from $t - \tau$ to t.

Table 4 OFER-GMM relative root mean square error on test set according the number of training examples and the number of components in GMMs for dataset $\mathcal{D}_2$

# training ex	12	18	30	60	120	180	300
RRMSE on test set: mean± std							
m-KRR	0.94 ± 0.46	0.76 ± 0.61	1.05 ± 0.36	0.82 ± 0.26	0.53 ± 0.09	0.40 ± 0.15	0.35 ± 0.07
OFER 1-GMM	0.85 ± 0.61	0.32 ± 0.61	0.94 ± 0.33	0.71 ± 0.24	0.53 ± 0.09	0.39 ± 0.14	0.35 ± 0.08
OFER 2-GMM	0.93 ± 0.61	0.80 ± 0.63	1.07 ± 0.40	0.66 ± 0.25	0.64 ± 0.11	0.41 ± 0.18	0.35 ± 0.12
OFER 3-GMM	1.12 ± 1.09	0.97 ± 1.11	1.13 ± 0.36	0.93 ± 0.31	0.63 ± 0.13	0.42 ± 0.18	0.34 ± 0.11

Table 5 Comparison of IOKR2,OFER-GSSM, m-KRR, and m-RF on the time series dataset of different sizes

Train size	Relative root mean squared error, $\pm std \leq 10^{-3}$			
	IOKR2	OFER-GSSM	m-KRR	m-RF
100	2.42	2.45	2.45	**2.37**
200	2.44	2.46	2.41	**2.37**
500	2.36	**2.26**	2.32	2.34
1000	2.36	**2.24**	2.27	2.31

Results of expressed in terms of average relative root mean squared error over 10 experiments. Standard deviations are all of the order of 1×10^{-2}

Table 6 Comparison of OFER and competitors on the text-to-time-series task in terms of 5-CV relative root mean squared error

Year/coordinate	2001	2002	2003	2004	2005	**Global**
5-CV relative root mean squared error, $\pm$ std $\leq 10^{-3}$						
m-KRR	1.27	**1.33**	1.47	1.30	1.71	1.42
m-RF	1.19	1.37	1.41	1.32	1.20	1.30
IOKR	1.16	1.34	1.32	1.30	1.24	1.27
IOKR-DTW	1.20	1.38	1.33	1.32	1.15	1.28
IOKR-GAK	1.20	1.38	1.32	1.29	**1.13**	1.26
OFER-GSSM	**1.11**	1.37	**1.31**	**1.28**	1.17	**1.25**

The "Global" column refers to the prediction of the short output time-series

To test our approach, we use the dataset described in Kogan et al. (2009) for which we modify both inputs and outputs to get a set of reports as input and a time series as output. The dataset $\mathcal{S} = \{(x_i, y_i)\}$ contains 500 companies, for each one we have the 10-K form reports from 2001 to 2005 and the Short-return log-volatility following each period with no overlap. We design a new learning task that consists of predicting a short-return log-volatility time series of length 5 from the 5 10-K form text reports corresponding to the 5 years, from 2001 to 2005. Given a company i, each yearly text report is represented using the classic TF-IDF using a dictionary of size 70×10^3, yielding to input vector x_i of size 35×10^4 while y_i is a vector of size 5.

6.3.2 Fully supervised learning

To apply OFER, we model the short output training time series with a linear state-space model. We use L^M_{Fisher} and the diagonal block matrix M writes as $M = diag(M_0, M_1, M_2, \ldots)$ defined in Eqs. 11 and 12. OFER based on a Gaussian State-Space Model is called OFER-GSSM. As announced we compare OFER-GSSM with m-KRR, m-RF and IOKR. In m-KRR, IOKR and OFER-GSSM, kernel k was chosen linear, giving rise to $K(x, x') = Id \times k(x, x') = Id \times (x^T x')$ with our notations of Eq. 13. As for the output kernel of IOKR, we have made three choices : a Gaussian kernel (IOKR) to simply take into account nonlinear relationships between outputs, and two kernels devoted to time-series similarity: FastDTW introduced in Salvador and Chan (2004) (IOKR-DTW) and the Global Alignement Kernel defined in Cuturi et al. (2007) (IOKR-GAK). The two last kernels take into account the time series structure and provide a fair comparison to our method while neither m-KRR, m-RF nor IOKR do take into account the hidden structure of the output time-series. Note

that in the single-output regression problem solved in Kogan et al. (2009), KRR exhibits the same performance as Support Vector Regression (SVR). On Table 6, 5-fold Cross-Validation Relative Root Mean Squared Errors (5-CV RRMSE) are reported with standard deviations. These last information are omitted whenever they are smaller than 10^{-3}. We observe that OFER-GSSM globally outperforms the other methods. We also give the details of the 5-CV Relative Root MSE (RRMSE), coordinate by coordinate.

Moreover, note that using Table 16 in the Appendix that reports computation time of all methods for training and test phase, we observe that OFER-GSSM exhibits a test time nearly as short as the faster method (m-RF) for a training phase that costs half of the training time for m-RF. OFER-GSSM provides here the best trade-off between accuracy and computational time.

6.4 Multi-output regression tasks

6.4.1 Scores prediction in personality videos

Task description
The second dataset is the development set of the First Impression Chalearn Challenge[5] (Ponce-López et al. 2016). It contains 6000 videos of 15 seconds interviews annotated with the Big 5 Traits that correspond to 5 continuous scores widely used in the personality analysis literature (John and Srivastava 1999). These annotations were obtained by fitting a Bradley-Terry-Luce model on pairwise preferences given by annotators from the Amazon Mechanical Turk. For each video, the task is then to predict 5 personality scores belonging to the interval $[0, 1]$ that have been inferred from a set of pairwise preferences among experts.

In this study, we only focus on videos for which at least one of the personality scores is far from the mean. We expect that these scores correspond to a strong agreement during the annotation step and could have been given by an expert. Relying on this assumption, we create two datasets of different sizes by rejecting the samples for which no personality score belongs to the interval: $[0, t] \cup [1 - t, 1]$ for different values of the agreement threshold t. In the following experiments, we choose the thresholds $t_1 = 0.9$ and $t_2 = 0.75$. The two choices lead to the construction of two training datasets S_1 and S_2, respectively of size 100 and 1000, with $S_1 \subset S_2$. The remaining set of 5000 examples is split in two. One will be used as a common test set and the other as a set from which we draw data with weak labels.

Following the baseline proposed during the challenge, we encode the videos by taking the first frame of each video and encoding it taking the fc8 feature map representation of the BVLC CaffeNet model.[6] We use (Gaussian) Kernel Ridge Regression (m-KRR) in the baseline approach as well as in OFER-GMM. We also use as baseline the Input and Output Kernel Regression (IOKR) and the multi-output Random Forest (m-RF). For IOKR we took as input and output kernel the linear kernel which show the best choice ofr this task.All the hyperparameters are selected using a 5-CV procedure.

Fully supervised learning In the following, we assess performance of OFER-GMM on the two proposed vectorial regression tasks. Relative Root Mean Square errors presented in Table 7 are computed based on the true Big 5 Traits.

[5] See http://gesture.chalearn.org/2016-looking-at-people-eccv-workshop-challenge/data-and-description.

[6] http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.

Table 7 Comparison of (Gaussian) m-KRR, m-RF, IOKR and OFER-GMM on the two video-to-score prediction tasks

Task	m-KRR	IOKR	m-RF	OFER-GMM
RRMSE on test set: mean $\pm$ std				
Task 1	$0.44 \pm 4 \times 10^{-2}$	$0.44 \pm 6 \times 10^{-2}$	$0.43 \pm 2 \times 10^{-2}$	$0.44 \pm 4 \times 10^{-2}$
Task 2	$0.42 \pm 4 \times 10^{-2}$	$0.42 \pm 4 \times 10^{-2}$	$0.43 \pm 2 \times 10^{-2}$	$0.42 \pm 4 \times 10^{-2}$

Table 8 Impact of the number of weakly supervised examples on weak OFER-GMM measured by RRMSE on test sets. GMM is omitted for sake of clarity

Task	OFER-GMM	wOFER +644	wOFER+1288	wOFER+2500
RRMSE on test set: mean $\pm$ std				
Task 1	$0.44 \pm 4 \times 10^{-2}$	$0.36 \pm 2 \times 10^{-3}$	$0.31 \pm 1 \times 10^{-4}$	$\mathbf{0.29 \pm 1 \times 10^{-4}}$
Task 2	$0.42 \pm 4 \times 10^{-2}$	$0.35 \pm 2 \times 10^{-3}$	$0.28 \pm 1 \times 10^{-3}$	$\mathbf{0.127 \pm 1 \times 10^{-4}}$

We see evidence that the four approaches perform equally well, meaning that in this fully supervised context, OFER-GMM does not bring any advantage. The relative easiness of the task may explain this result.

Weakly supervised learning We now investigate the behaviour of OFER-GMM in the context of weakly supervised learning.

Table 8 presents results obtained with a Gaussian mixture model whose number of components and all other hyperparameters have been selected using a grid search strategy with a cross-validation procedure on the training set. Adding weakly supervised examples that only provide the membership to one of the two components of the Gaussian Mixture model allow to drastically improve the performance of OFER. Information conveyed by these examples consolidates the learned function.

6.4.2 Drug activity prediction

We present here, another multi-output regression task in order to assess OFER. We have chosen a dataset related to Drug Activity Prediction introduced by Su et al. (2010) and also studied as an application of IOKR in Brouard et al. (2016). In this problem the main goal is to predict activities of molecules on cancer cell lines. The input set, $\mathcal{X}$ corresponds to the set of 2303 molecules where each molecule is represented as a graph labeled by atoms. In all the experiments we directly used a representation of the training data as a Gram matrix with Tanimoto kernel (Brouard et al. 2016). Output data are a set of 59 scores of activity for each molecule. We have chosen the "Non-zero Activity" dataset version where are only present active molecules for at least one cell. The dataset has been split 10 times into a train/test sets (60, 40%), and we train our model with different size of train sets (10, 20, 100, 300, 500). The remaining part of the train dataset is used to build the weak labeled of size 100, 500, 800. As done in the Personality Video problem, results are presented in terms of RRMSE with a standard deviation computed on 10 times repetitions. Baseline methods are the same

Table 9 Comparison of (Gaussian) m-KRR, RF, IOKR and OFER-GMM on drug activity prediction

Train size	m-KRR	IOKR	m-RF	OFER-GMM
RRMSE on test set: mean ± std (%)				
10	0.24 ± 0.006	0.24 ± 0.006	0.23 ± 0.033	**0.22 ± 0.009**
20	0.22 ± 0.004	0.22 ± 0.004	0.22 ± 0.018	**0.21 ± 0.008**
100	0.22 ± 0.003	0.22 ± 0.003	0.21 ± 0.007	**0.20 ± 0.004**
300	0.20 ± 0.002	0.20 ± 0.002	0.19 ± 0.004	**0.19 ± 0.002**
500	0.19 ± 0.002	0.19 ± 0.002	0.19 ± 0.003	0.19 ± 0.001

Table 10 Impact of the number of weakly supervised examples on weak OFER-GMM measured by RRMSE on test sets. GMM is omitted for sake of clarity

Train size	OFER-GMM	wOFER+100	wOFER+500	wOFER+800
RRMSE on test set: mean ± std				
10	0.22 ± 0.009	0.20*	0.20*	**0.19***
20	0.21 ± 0.008	0.20*	0.19*	0.19*
100	0.20 ± 0.004	0.20*	0.19*	**0.18***
300	0.19 ± 0.002	0.18*	0.18*	**0.17***
500	0.19 ± 0.001	0.18*	0.17*	**0.16***

*Means that Std is lower than 10^{-4}

than those used in Table 7, except for the IOKR baseline where the output kernel is linear here. Tables 9 and 10 sum up the results we obtained for this problem.
For this problem, OFER enables us to get slightly better performances in terms of RRMSE and also to get smaller standard deviations. A notable fact is that the selected number of components in the mixture model, selected by cross-validation, is 6 which corresponds to 6 clear levels of scores when observing the data.

6.5 Multi-class classification

Task description
We now propose to address an image multiclass classification problem cast into a word prediction task. The idea is that object classes have semantics and by replacing classes by names (words), we allow the learning algorithm to take advantage of this semantics. We considered two real datasets: the first one is the well-known **Caltech101**[7] image dataset consisting of images of 101 classes and the second one is **Animals With Attributes** dataset (AWA,[8] Akata et al. 2016) consisting of images of 50 animal classes. In both case, the final goal is to automatically predict the name of the animal/object present in a given image. Note that for AWA dataset, our purpose is not to work on information transfer but only have a dataset where the names of the objects might be meaningful to cluster. So we do not use at all the attributes information about animals as studied in Akata et al. (2016). We also emphasize

[7] http://www.vision.caltech.edu/Image_Datasets/Caltech101/.

[8] http://attributes.kyb.tuebingen.mpg.de/.

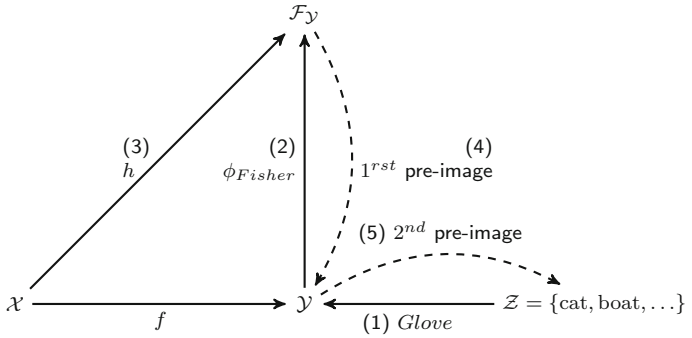

Fig. 3 Scheme of the structured learning task with the output kernel trick. The numbering describes the order of computation in order to set the whole approach

that the binary nature of the Attribute Label Embedding and Hierarchical Label Embedding proposed in Akata et al. (2016) do not lead to a dense output space where our method applies.

In both applications, we aim at showing that the Fisher Output Embedding allows us to encode meaningful dependencies in the output space taking into account the semantic properties of the words describing the categories. The scheme proposed in Fig. 1 is slightly modified into Fig. 3 to support two steps in the pre-image processs.

Following the new scheme, $\mathcal{X}$ is the input feature space based on the fc7 feature map of the VGG19 pretrained on the ILSVRC2014.[9] The target set for Caltech101 is now $\mathcal{Z} = \{\text{cat}, \text{boat}, \ldots\}$ and $\mathcal{Z} = \{\text{lion}, \text{tiger}, \ldots\}$ for AWA. Each class in $\mathcal{Z}$ is seen as a symbol and not a word. An intermediary semantic vector space Y is defined using the following three-steps feature construction:

- First, we built the "Wiki corpus" by scraping the wiki pages, containing the textual descriptions of the 101 (resp. 50) classes. We have kept only the introductory paragraph.
- Second, for each word of the "Wiki corpus" we retrieve the GLOVE (Pennington et al. 2014) semantic representations. That is to say that we get a 50-dimensional vector for each word. We convert the corpus into a TF-IDF matrix.
- Third, we represent each class as a weighted combination of its GLOVE represented word, with the weight of the TF-IDF.

Using this semantic encoding of words, $\mathcal{Y}$ is a 50-dimensional vector space. Now the Fisher embedding applies on vectors of $\mathcal{Y}$, transforming the 50-dimensional vectors of $\mathcal{Y}$ into $50+C$-dimensional vectors of $\mathcal{F}_\mathcal{Y}$, denoting C, the number of GMM components. Once the GMM and the function h are learned, the prediction for a given input image x is obtained by first computing the image $h(x)$, then computing the pre-image $\hat{y}$ in $\mathcal{Y}$ and eventually, picking the class in $\mathcal{Z}$ whose semantic representation in $\mathcal{Y}$ is the closest to the prediction $\hat{y}$. For all experiments, the evaluation metrics are the classification accuracy of the multi-class classifier. Therefore, in case of M classes, a random classifier would have an accuracy of $\frac{1}{M}$.

Learning in small data regime

Our framework (OFER-GMM) is compared to two simple multiclass classifiers: SVM (m-SVM) with a one-versus-all strategy and Multiclass Random Forest (m-RF) and a multiple output Kernel Ridge Regression with linear kernel (Sem-KRR) working in the semantical output space $\mathcal{Y}$. Let us first report the performance of multiclass Random Forest: on Caltech101, from 1 to 10 examples per class the classification accuracy of m-RF did not exceed

[9] http://image-net.org/challenges/LSVRC/2014/.

1.6% with a std of 0.45 . On AWA, the mean accuracy varies from 10 to 18% which is again very low and far from the performance of the other methods.

For both datasets, multiclass Random Forest has been tested but it suffers from the limited number of training examples per label. We also made comparisons with a method devoted to structured outputs: Input Output Kernel Regression (IOKR). IOKR is implemented here using an identity-decomposable operator-valued kernel defined on the input space and a scalar-valued output kernel, here taken as Gaussian one on the semantic space $\mathcal{Y}$. As the number of target classes is limited, the minimization problem involved in pre-image problem is exactly solved.

We also tried to compare our method with methods from the deep learning community (Vinyals et al. 2016; Sohn et al. 2015). In Sohn et al. (2015) the authors developed a deep conditional generative model for structured output variables using Gaussian latent variables (CVAE). However their approach estimates distributions from very large datasets to capture global regularities. On the other hand, we designed OFER as a framework tailored for small to medium sized dataset that also allows the user to take advantage of weak annotations. Consequently when it comes to the empirical approach, we cannot really compare with CVAE since we do not work on the same type of problems. When trying to apply CVAE in the small data regime, the algorithm does not converge to a solution and gives an accuracy close to 0%.

Nonetheless, in order to have a deep learning baseline, we attempted to finetune the pretrained VGG19 network but obtained our best results with all the convolutional layers frozen. We refer to this solution as MLP.

A 5-Cross-Validation based on the training set was performed to select the hyperparameters, among which the number of clusters for the OFER-GMM (from 1 to 10). We repeat 10 times the experiments (10 different pairs of training and test sets) to get a mean accuracy and a standard deviation on the test sets. Each test set contains 5472 examples for Caltech101 and 9143 for AWA. Tables 11 and 12 present the performance of these 4 classifiers in terms of classification accuracy for each dataset, Caltech101 and AWA. A first observation is that all the four methods that make use of semantic embedding, namely MLP, Sem-IOKR, Sem-KRR and OFER-GMM outperfom m-SVM when the number of labeled examples per class is very low: typically $\#ex < 7$ for Caltech101 and $\#ex < 14$ for AWA. This threshold correspond in each case to a train set of size 700. The idea that consists of replacing indexes of object classes by class names seems therefore relevant in this application. The lack of labeled data for a given class is bridged by the information conveyed by other labels in semantically close classes. Now, in case of a relatively large set of labeled examples per classes (10), OFER-GMM is outperformed by m-SVM in Caltech101 experiments and it is not worth applying the OFER framework in this case. The results on AWA datasets seem less clear due to the large standard deviations. OFER-GMM gets a better performance than its competitors in case of a number of examples per class less than 14. In particular, OFER-GMM beats Sem-KRR each time.

The conclusion of those results is that our framework OFER outperforms the other baselines.

Weakly Supervised learning

In the following experiments, we measure how the use of weak supervision impacts the performance of OFER-GMM. We follow exactly the same scenario (growing number of training set and same constant size test set) than previously except that we use an additional weakly supervised dataset of 700, 1500 and 2500 instances. The output of these training examples is of size the number of GMM components and contains the information of the component to which belongs the output. Hyperparameter λ_2 that leverages the importance of

Table 11 Comparison between OFER-GMM, Sem-KRR, Sem-IOKR, Multi-layer Perceptron (MLP) and SVM on Caltech101 with a growing number of labeled examples per class

#ex/class	Classification accuracy on test set: mean ± std (%) - Caltech101				
	m-SVM	MLP	Sem-IOKR	Sem-KRR	OFER-GMM
1	9.61 ± 3.98	10.80 ± 3.78	13.40 ± 2.22	14.83 ± 4.02	**38.22 ± 2.87**
2	23.24 ± 2.03	12.73 ± 4.46	18.80 ± 2.10	19.96 ± 2.68	**43.89 ± 2.24**
3	33.89 ± 1.79	17.25 ± 3.18	22.51 ± 1.81	22.71 ± 2.33	**46.33 ± 2.44**
4	42.23 ± 1.8	25.14 ± 1.89	22.83 ± 1.41	24.52 ± 1.93	**48.41 ± 2.25**
5	47.63 ± 2.87	27.04 ± 2.45	24.90 ± 1.27	25.91 ± 1.28	**49.40 ± 2.09**
7	**55.19 ± 2.43**	35.84 ± 3.48	26.84 ± 0.92	27.42 ± 1.59	50.39 ± 2.04
10	**58.55 ± 1.84**	44.93 ± 3.80	31.27 ± 1.84	29.49 ± 1.39	50.49 ± 1.07

Table 12 Comparison between OFER-GMM, Sem-KRR ,Sem-IOKR, MLP, m-SVM and m-RF on AWA dataset with a growing number of labeled example per class

#ex/class	Classification accuracy on test set: mean ± std (%) - AWA				
	m-SVM	MLP	Sem-IOKR	Sem-KRR	OFER-GMM
1	1.91 ± 0.13	22.18 ± 3.98	0.26 ± 0.17	21.85 ± 3.04	**23.00 ± 3.11**
2	2.12 ± 0.15	30.62 ± 3.59	6.43 ± 1.15	29.77 ± 2.33	**31.05 ± 2.17**
3	6.33 ± 1.12	34.06 ± 3.65	14.93 ± 1.56	33.21 ± 2.02	**34.79 ± 2.11**
4	12.87 ± 0.70	38.02 ± 2.31	20.81 ± 1.94	37.01 ± 1.97	**37.91 ± 2.07**
5	18.53 ± 1.69	39.36 ± 2.04	26.26 ± 1.74	39.97 ± 1.75	**40.74 ± 1.70**
7	26.85 ± 2.24	41.36 ± 1.86	33.82 ± 1.37	42.89 ± 0.93	**43.24 ± 0.58**
10	37.36 ± 1.14	43.88 ± 1.80	42.61 ± 1.3	44.21 ± 1.00	**45.23 ± 1.00**
11	40.55 ± 0.56	44.50 ± 2.38	44.91 ± 1.4	45.67 ± 0.94	**46.26 ± 1.08**
14	47.00 ± 1.00	46.25 ± 1.57	**49.7 ± 0.89**	47.08 ± 1.24	47.38 ± 1.29

the weakly supervised term is chosen by cross-validation. Table 13 (resp. Table 14) gathers the performance of OFER-GMM weak variants compared to the best competitor: m-SVM for Caltech101 and mlp for AWA. As expected, adding weakly labeled data generally improves the classification accuracy but also drastically reduces the standard deviation. The information brought by the membership to the components of the GMM, even if very weak, allows to diffuse information about classes of the same component. More insights on the impact of the hyperparameter λ_2 on accuracy can be found in Table 15 in the Appendix.

7 Conclusion

Output Fisher embedding regression is a general framework able to account for explicit and implicit structure in output data by exploiting Fisher vectors in the output space. We have shown that the corresponding pre-image problems admit closed-form solutions in the case of Gaussian Mixture Models and Gaussian State-Space models, allowing for a fast prediction phase compared to Structured Output Prediction method such as IOKR.

Table 13 Impact of weak supervision (wOFER+ data weak size) on OFER for the classification task using growing sizes of weakly supervised dataset (700, 1500, 2500)

#ex/class	Classification accuracy on test set: mean ± std (%) - Caltech101				
	m-SVM	OFER-GMM	wOFER+700	wOFER +1500	wOFER+2500
1	9.61 ± 3.98	38.22 ± 2.87	39.19 ± 0.20	39.90 ± 0.20	**40.15 ± 0.20**
2	23.24 ± 2.03	43.89 ± 2.24	44.35 ± 0.01	44.82 ± 0.02	**45.16 ± 0.01**
3	33.89 ± 1.79	46.33 ± 2.44	47.01 ± 0.01	47.56 ± 0.01	**47.83 ± 0.01**
4	42.23 ± 1.8	48.41 ± 2.25	48.83 ± 0.01	49.32 ± 0.01	**50.24 ± 0.01**
5	47.63 ± 2.87	49.40 ± 2.09	50.32 ± 0.01	51.13 ± 0.01	**51.65 ± 0.01**
7	**55.19 ± 2.43**	50.39 ± 2.04	51.04 ± 0.01	51.63 ± 0.01	52.35 ± 0.01
10	**58.55 ± 1.84**	50.49 ± 1.07	51.14 ± 0.01	52.13 ± 0.01	52.83 ± 0.01

For sake of simplicity, the suffix GMM is omitted

Table 14 Impact of weak supervision (wOFER+ data weak size) on OFER for AWA using growing sizes of weakly supervised dataset (700,1500, 2500)

#ex/ class	Classification accuracy on test set: mean ± std (%) - AWA				
	MLP	OFER	wOFER+700	wOFER+1500	wOFER+2500
1	22.18 ± 3.98	23.00 ± 3.11	24.19 ± 0.10	24.90 ± 0.09	**25.67 ± 0.10**
2	30.62 ± 3.59	31.05 ± 2.17	32.71 ± 0.01	33.36 ± 0.01	**34.04 ± 0.01**
3	34.06 ± 3.65	34.79 ± 2.11	36.22 ± 0.01	37.03 ± 0.01	**37.63 ± 0.01**
4	38.02 ± 2.31	37.91 ± 2.07	38.53 ± 0.01	39.32 ± 0.01	**40.14 ± 0.01**
5	39.36 ± 2.04	40.74 ± 1.70	41.35 ± 0.01	42.22 ± 0.02	**43.16 ± 0.01**
7	41.36 ± 1.86	43.24 ± 0.58	44.00 ± 0.01	44.96 ± 0.01	**45.83 ± 0.01**
10	43.88 ± 1.80	45.23 ± 1.00	46.13 ± 0.01	47.18 ± 0.01	**47.84 ± 0.01**
11	44.50 ± 2.38	46.26 ± 1.08	47.37 ± 0.01	48.02 ± 0.01	**48.94 ± 0.01**
14	46.25 ± 1.57	47.38 ± 1.29	49.04 ± 0.01	49.63 ± 0.01	**50.35 ± 0.01**

OFER has been applied on a wide variety of tasks, showing its versatility. Beyond Structured Output Regression like time series prediction, OFER is also of interest for regression and multi-classification tasks when the training dataset is small. OFER-GMM, based on the Fisher embedding of Gaussian mixture model, especially exhibits a very interesting behaviour in multi-classification tasks when learning from a handful of data per label. The Fisher embedding seems to remedy to the lack of examples per label, overcoming the performance of the other approaches, including the ones that take advantage of a semantic encoding. It is also important to notice the flexibility of OFER-GMM whose number of components selected by cross-validation is adapted to the observed outputs.

Additionally, the interpretability of OFER-GMM outputs opens the door to learning under weak supervision, avoiding the cost of expert labeling. This is observed in multi-output regression as well as the multi-classification tasks.

We identify at least two attractive use cases for OFER. The first one is in Structured Output Prediction when one wants to avoid an expensive prediction phase while accounting for the structure. The second one is in small data regime for any problem that can be cast into multi-output regression where the training outputs can be clustered.

First working perspectives concern the improvement of the whole approach by learning jointly the parametric probabilistic model and the predictive function. Second, this paper has focused on vectorial outputs and time series but the framework is more general. An attractive direction is to apply this approach on more complex outputs in applications where the training dataset is often very limited such as bioinformatics and chemoinformatics.

Acknowledgements The authors are very grateful to Slim Essid, Chloé Clavel (LTCI, Télécom Paristech) and Zoltán Szabó (CMAP, Ecole Polytechnique) for fruitful discussions about this work. Moussab Djerrab is supported by the Télécom ParisTech Machine Learning for Big Data Chair.

Appendix

A: Additional insights on multiclass classification as a word prediction task

Hyperparameter λ_2 leverages the importance given to the weakly supervised data in the case of wOFER+2500. Table 15 shows the evolution of accuracy when λ_2 is varying while λ_0 is fixed. We show on both datasets AWA and Caltech101 that a good balance between the weak supervision term and the other terms has to be found as it has already been noticed in semi-supervised learning litterature.

B: Empirical study of time complexity

To complete the experimental results, we provide here a comparative empirical study of time complexity using the same computer (with no GPU) in a simple configuration (2.9 GHz processor with 8Gb RAM). As we mainly target small regime data, the learning time is not an issue. Conversely, time in test phase reflects the cost of prediction and is the one that really matters. For sake of completeness, both learning and test times are reported for the different methods as well as the different tasks in Tables 16, 17, 18, 19, and 20.

In all cases, the reported times are consistent and show that OFER exhibits a test time of the order of m-KRR, usually slighlty higher due to the additional pre-image resolution.

Table 15 Impact of the value of λ_2 in wOFER+2500 on the performance

	λ_2	
	Mean accuracy % ± std	wOFER + 2500
Caltech101	0	29.49 ± 1.39
	0.001	45.23 ± 0.01
	0.01	**52.83 ± 0.01**
	0.1	52.35 ± 0.01
	1	41.14 ± 0.01
	10	27.51 ± 0.01
AWA	0	47.08 ± 1.24
	0.001	49.97 ± 0.01
	0.01	50.06 ± 0.01
	0.1	**50.35 ± 0.01**
	1	44.38 ± 0.01
	10	21.61 ± 0.01

Table 16 Computation times of all methods tested on text-to-time-series with training and test set used in Table 6

Computation time (s), mean ± std Time-series prediction		
Method/phase	Learning	Test
m-KRR	2.13 ± 0.54	0.12 ± 0.01
m-RF	57.30 ± 7.46	0.08 ± 0.01
IOKR	2.23 ± 0.64	2.63 ± 1.12
IOKR-DTW	43.51 ± 5.37	1.33 ± 0.76
IOKR-GAK	3.63 ± 0.58	1.38 ± 0.43
OFER-GSSM	19.62 ± 3.23	0.13 ± 0.01

Table 17 Computation time in learning on drug activity prediction (500 training data) and video score prediction (1000 training data)

Computation time (s) in learning phase: mean ± std		
Method/task	Drug activity	Video score
m-KRR	0.74 ± 0.01	1.33 ± 0.01
IOKR	1.63 ± 0.01	1.82 ± 0.01
m-RF	0.56 ± 0.01	1.31 ± 0.01
OFER-GMM	0.60 ± 0.01	1.41 ± 0.02

Table 18 Computation time in test phase on drug activity prediction (1000 test data) and video score prediction (3000 test data) for all tested methods

Computation time in test phase : mean ± std		
Method/task	Drug activity	Video score
m-KRR	0.014 ± 0.01	0.035 ± 0.01
IOKR	2.13 ± 0.01	3.67 ± 0.02
m-RF	0.011 ± 0.01	0.017 ± 0.01
OFER-GMM	0.015 ± 0.01	0.035 ± 0.01

Table 19 Computation time in learning phase on AWA (14 examples per class in the training set) and Caltech101 (10 examples per class in the training set) for all tested methods

Computation time (s) in learning phase: mean ± std		
Method/task	AWA	Caltech101
m-SVM	1633.51 ± 2.31	1267.87 ± 1.67
MLP	6192.21 ± 1.17	4312.42 ± 2.53
Sem-IOKR	62.13 ± 1.82	42.92 ± 0.64
Sem-KRR	18.32 ± 2.11	13.56 ± 1.22
OFER-GMM	19.13 ± 0.51	13.91 ± 1.28

Interestingly, it requires significantly less time in test phase than a structured output prediction methods such as IOKR based on a more involved pre-image problem to solve. Similarly, training times of OFER are comparable to those of m-KRR except in the Time series prediction problem where estimating a State-Space Model involves more time. Note that MLP and m-SVM are not competitive in terms of training or test time. When m-RF is eligible in terms of performance (all cases except AWA and CalTech101), it provides the best computation time in the test phase. To conclude, OFER allows to improve prediction accuracy especially

Table 20 Computation time in test phase on AWA (9143 test data) and Caltech101 (5472 test data) for all tested methods

Computation time (s) in test phase: mean ± std		
Method/task	AWA	Caltech101
m-SVM	101.77 ± 1.02	60.10 ± 0.79
MLP	60.57 ± 0.99	41.04 ± 1.12
Sem-IOKR	1783 ± 1.63	1067 ± 1.31
Sem-KRR	7.01 ± 0.81	4.18 ± 1.67
OFER-GMM	7.12 ± 2.10	4.19 ± 1.41

when learning in the small data regime at a cost largely smaller than the cost of a structured output method such as IOKR.

References

Akata, Z., Perronnin, F., Harchaoui, Z., & Schmid, C. (2016). Label-embedding for image classification. *IEEE Transactions on Pattern Analysis Machine Intelligence*, *38*(7), 1425–1438.

Álvarez, M. A., Rosasco, L., & Lawrence, N. D. (2012). Kernels for vector-valued functions: A review. *Foundations and Trends in Machine Learning*, *4*(3), 195–266.

Amari, S.-I. (1998). Natural gradient works efficiently in learning. *Neural Computation*, *10*(2), 251–276.

Bakir, G., Hofmann, T., Schölkopf, B., Smola, A., Taskar, B., & Vishwanathan, S. (2007). *Predicting structured data*. Cambridge: MIT Press.

Brouard, C., d'Alché-Buc, F., & Szafranski, M. (2011). Semi-supervised penalized output kernel regression for link prediction. In *International conference on machine learning (ICML)* (pp. 593–600).

Brouard, C., d'Alché Buc, F., & Szafranski, M. (2016). Input output kernel regression. *Journal of Machine Learning Research*, *17*(176), 1–48.

Chen, L., Schwing, A. G., Yuille, A. L., & Urtasun, R. (2015). Learning deep structured models. In *Proceedings of the 32nd international conference on machine learning, ICML 2015* (pp. 1785–1794).

Ciliberto, C., Rosasco, L., & Rudi, A. (2016). A consistent regularization approach for structured prediction. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems 29* (pp. 4412–4420). New York: Curran Associates Inc.

Cortes, C., Mohri, M., & Weston. J. (2005). A general regression technique for learning transductions. In *International conference on machine learning (ICML)* (pp. 153–160).

Cuturi, M., Vert, J., Birkenes, Ø., & Matsui, T. (2007). A kernel for time series based on global alignments. In *IEEE international conference on acoustics, speech and signal processing (ICASSP), 2007*.

Fei-Fei, L., Fergus, R., & Perona, P. (2006). One-shot learning of object categories. *IEEE Transactions on Pattern Analysis Machine Intelligence*, *28*, 594–611.

Geurts, P., Wehenkel, L., & d'Alché-Buc, F. (2006). Kernelizing the output of tree-based methods. In *International conference on machine learning (ICML)* (pp. 345–352).

Geurts, P., Wehenkel, L., & d'Alché-Buc, F. (2007). Gradient boosting for kernelized output spaces. In *Machine learning, proceedings of the twenty-fourth international conference (ICML 2007), Corvallis, Oregon, USA, June 20–24, 2007* (pp. 289–296).

Hofmann, T. (2000). Learning the similarity of documents: An information-geometric approach to document retrieval and categorization. In S. A. Solla, T. K. Leen, & K. Müller (Eds.), *Advances in neural information processing systems 12* (pp. 914–920). MIT Press.

Honeine, P., & Richard, C. (2011). Preimage problem in kernel-based machine learning. *IEEE Signal Processing Magazine*, *28*(2), 77–88.

Hou, Y., Hsu, W., Lee, M. L., & Bystroff, C. (2003). Efficient remote homology detection using local structure. *Bioinformatics*, *19*(17), 2294.

Jaakkola, T., & Haussler, D. (1998). Exploiting generative models in discriminative classifiers. In M. J. Kearns, S. A. Solla, & D. A. Cohn (Eds.), *In advances in neural information processing systems 11* (pp. 487–493). Cambridge: MIT Press.

John, O. P., & Srivastava, S. (1999). The big five trait taxonomy: History, measurement, and theoretical perspectives. *Handbook of Personality: Theory and Research*, *2*(1999), 102–138.

Kadri, H., Ghavamzadeh, M., & Preux, P. (2013). A generalized kernel approach to structured output learning. In *International conference on machine learning (ICML)* (pp. 471–479).

Kocev, D., Vens, C., Struyf, J., & Dzeroski, S. (2013). Tree ensembles for predicting structured outputs. *Pattern Recognition*, *46*(3), 817–833.

Kogan, S., Levin, D., Routledge, B. R., Sagi, J. S., & Smith, N. A. (2009). Predicting risk from financial reports with regression. In *Proceedings of the human language technologies: NAACL'09*.

Lafferty, J. D., McCallum, A., & Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the international conference on machine learning (ICML)*.

Lebret, R., Pinheiro, P. H. O., & Collobert, R. (2015). Phrase-based image captioning. In *International conference on machine learning (ICML)* (pp. 2085–2094).

LeCun, Y., & Huang, F. (2005). Loss functions for discriminative training of energy-based models. In *Proceedings of the 10-th international workshop on artificial intelligence and statistics (AIStats'05)*.

Micchelli, C. A., & Pontil, M. A. (2005). On learning vector-valued functions. *Neural Computation*, *17*, 177–204.

Nowozin, S., & Lampert, C. H. (2011). Structured learning and prediction in computer vision. *Foundations and Trends Computer Graphics and Vision*, *6*(3:8211;4), 185–365.

Oneata, D., Verbeek, J., & Schmid, C. (2013). Action and event recognition with Fisher vectors on a compact feature set. In *Proceedings of the IEEE international conference on computer vision* (pp. 1817–1824).

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. In *Empirical methods in natural language processing (EMNLP)* (pp. 1532–1543).

Perronnin, F., Sánchez, J., & Mensink, T. (2010). Improving the Fisher kernel for large-scale image classification. In *Proceedings of the 11th european conference on computer vision: Part IV, ECCV'10* (pp. 143–156). Berlin, Heidelberg: Springer.

Ponce-López, V., Chen, B., Oliu, M., Corneanu, C., Clapés, A., Guyon, I., Baró, X., Escalante, H. J., & Escalera, S. (2016). Chalearn lap 2016: First round challenge on first impressions-dataset and results. In *Computer vision–ECCV 2016 workshops* (pp. 400–418). Berlin: Springer.

Pugelj, M., & Džeroski, S. (2011). Predicting structured outputs k-nearest neighbours method. In *Proceedings of the 14th international conference on discovery science, DS'11* (pp. 262–276). Berlin, Heidelberg: Springer.

Salvador, S., & Chan, P. (2004). Fastdtw: Toward accurate dynamic time warping in linear time and space. In *KDD workshop on mining temporal and sequential data*. Citeseer.

Siolas, G., & d'Alché-Buc, F. (2002). Mixtures of probabilistic pcas and Fisher kernels for word and document modeling. In *ICANN 2002* (pp. 769–776).

Sohn, K., Lee, H., & Yan, X. (2015). Learning structured output representation using deep conditional generative models. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett, (Eds.), *Advances in neural information processing systems 28* (pp. 3483–3491). Curran Associates, Inc.

Su, H., Heinonen, M., & Rousu, J. (2010). Structured output prediction of anti-cancer drug activity. In *International conference on pattern recognition in bioinformatics (PRIB)* (pp. 38–49). Berlin: Springer.

Sydorov, V., Sakurada, M., & Lampert, C. H. (2014). Deep fisher kernels—End to end learning of the Fisher kernel GMM parameters. In *2014 IEEE conference on computer vision and pattern recognition, CVPR* (pp. 1402–1409).

Tsochantaridis, I., Joachims, T., Hofmann, T., & Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, *6*, 1453–1484.

Vezhnevets, A., Ferrari, V., & Buhmann, J. M. (2012). Weakly supervised structured output learning for semantic segmentation. In *2012 IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 845–852). IEEE.

Vinyals, O., Blundell, C., Lillicrap, T. P., Kavukcuoglu, K., & Wierstra, D. (2016). Matching networks for one shot learning. *CoRR*. arXiv:1606.04080.

Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). Show and tell: A neural image caption generator. In *The IEEE conference on computer vision and pattern recognition (CVPR), June 2015*.

Yu, C. J., & Joachims, T. (2009). Learning structural svms with latent variables. In A. P. Danyluk, L. Bottou, and M. L. Littman (Eds.) *Proceedings of the 26th annual international conference on machine learning, ICML 2009, Montreal, Quebec, Canada, June 14–18, 2009* (Vol. 382, pp. 1169–1176). ACM.

Zhu, L., Chen, Y., Yuille, A. L., & Freeman, W. T. (2010). Latent hierarchical structural learning for object detection. In *The twenty-third IEEE conference on computer vision and pattern recognition, CVPR 2010* (pp. 1062–1069).