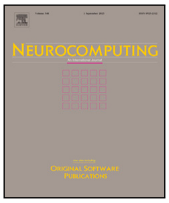
Neurocomputing 561 (2023) 126873

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



Direct side information learning for zero-shot regression

Miriam Fdez-Díaz [a,*], Elena Montañés [b], José Ramón Quevedo [b]

[a] *ArcelorMittal, Spain*
[b] *Artificial Intelligence Center. University of Oviedo at Gijón, 33204 Asturias, Spain* [1]

ARTICLE INFO

Communicated by Y. Cheng

Keywords:
Side information
Zero-shot regression
Kernel

ABSTRACT

Zero-shot learning provides models for targets for which instances are not available, commonly called unobserved targets. The availability of target side information becomes crucial in this context in order to properly induce models for these targets. The literature is plenty of strategies to cope with this scenario, but specifically designed on the basis of a zero-shot classification scenario, mostly in computer vision and image classification, but they are either not applicable or easily extensible for a zero-shot regression framework for which a continuous value is required to be predicted rather than a label. In fact, there is a considerable lack of methods for zero-shot regression in the literature. Two approaches for zero-shot regression that work in a two-phase procedure were recently proposed. They first learn the observed target models through a classical regression learning ignoring the target side information. Then, they aggregate those observed target models afterwards exploiting the target side information and the models for the unobserved targets are induced. Despite both have shown quite good performance because of the different treatment they grant to the common features and to the side information, they exploit features and side information separately, avoiding a global optimization for providing the unobserved target models. The proposal of this paper is a novel method that jointly takes features and side information in a one-phase learning process, but treating side information properly and in a more deserving way than as common features. A specific kernel that properly merges features and side information is proposed for this purpose resulting in a novel approach that exhibits better performance over both artificial and real datasets.

1. Introduction

Improving predictions of air pollutants in meteorological stations makes arise the research of this work, in particular some damaging pollutants (NO_2, PST, NO, SO_2, CO, O_3) collected in the Principality of Asturias, Spain. There are several factors that hardly condition the concentration of these pollutants. Both weather conditions (temperature, humidity, pressure...) and the activities around the meteorological stations (industry, leisure centers, residential areas, power plants, administrative buildings....) seem to be the most influential. However, there are from different nature. On the one hand, weather conditions depend on the climate and vary along the day, weeks, months and seasons. On the other hand, activities around the stations are constant for each station, hardly vary along the time and are known beforehand even if weather conditions have not been collected yet. Some studies ignore the information about activities in the surroundings [1]. Just some studies consider both kinds of factors, but they treat them separately and in a different manner. An advanced work considers activities in the surroundings to perform a previous split of the stations resulting in a set of models, one per station or group of stations, where just weather conditions are taken as features [2]. Also, taking pollutant values from the nearest neighbor station [3] is a preliminary work of including information about the stations. A prior classification of the stations taking into account roads, traffic flow and the area (urban, suburban or industrial) where the station is located [4] is another advanced strategy that considers the surroundings of the stations. However, this information, called side (or privileged) information [5–9], can go away along if it is exploited properly, for instance, including it in the learning process, in order to improve the pollutant predictions. Side information is neither features nor targets; it actually constitutes additional and, prior information about how targets (a pollutant measure in stations) are related to features (weather conditions). Also, treating surrounding information about stations as side information allows us to make predictions over potential future locations of meteorological stations, for which only side information is available. This perspective enables us to state the problem as a zero-shot regression learning task.

Zero-shot [10] is a kind of learning that tries to provide predictions for targets devoid of instances. These targets are commonly called

* Corresponding author.
E-mail addresses: miriam.fernandezdiaz@arcelormittal.com (M. Fdez-Díaz), montaneselena@uniovi.es (E. Montañés), quevedo@uniovi.es (J.R. Quevedo).
[1] http://www.aic.uniovi.es

https://doi.org/10.1016/j.neucom.2023.126873
Received 12 March 2023; Received in revised form 9 July 2023; Accepted 30 September 2023
Available online 7 October 2023

unobserved targets. The lack of instances for the unobservable targets really complicates the task of providing promising models to these targets, since only instances for observable targets are available. Then, one must draw on alternative resources to fill the gap. This is the point where side information is crucial and comes into play. Side information is commonly found in form of features [11,12], as it will be in our case. But depending on the context, it can be found in different formats, for instance in a hierarchy [13], in a response prediction [14] or in a structure [15]. Besides, sometimes it is multimodal and heterogeneous, as it happens in recommender systems [16]. In any case, all their special properties make side information worth exploiting in a way far from the way the features are exploited [17]. Its prominence is such in the zero-shot framework that great efforts have been made in the literature to collect it [12,18], or even, to extract or to learn it [19,20].

Lately, zero-shot learning has been a booming topic due to the increasing number of applications that demand predicting unobservable targets. This can be, for instance, the case of COVID-19 diagnosis classification, a recent topic that impacts the world notoriously [21]. Natural language processing [22–26], videos [27–32], mobile and wireless security [33], emoji predictions [34,35], human activity recognition [36], or neuroimaging data [37] are other applications. However, the great majority of applications of zero-shot learning concerns to image classification, facial recognition and computer vision [30,32,38–47]. This happens to such a high degree that researchers usually refer to just these applications as zero-shot [40,46], despite the emerging existence of some other classification applications [24,26,33,36]. Reviewing the state-of-the-art of the zero-shot approaches, one can find that they are mostly proposed for classification task (predicting a label) rather for regression task (predicting a continuous value), as it is our case of predicting pollutant concentrations. The main drawback of these approaches is that they adopt decisions and establish strategies that are specific of a classification task. Hence, extending or adapting them to a regression task is non-obvious, non-direct and even unfeasible [48]. The so-called regression-based zero-shot methods [12,26,39,40,46] have paved the way among the existing strategies to cope with zero-shot learning. Despite these approaches being labeled as regression-based methods, they do not solve regression zero-shot learning; otherwise, they cope with classification zero-shot. But they are referred as regression-based zero-shot methods because they commonly include regression learning to make projections to map instances in certain spaces depending on the classes of the classification, typically feature or side information spaces, as a previous step of performing the classification into classes. So far, very few approaches are found in the literature for zero-shot regression that would solve several other problems beyond the pollutant concentration prediction. For instance, in an industrial environment [49], predicting the performance of the parts of a machine given different system's states, where the parts are interchangeable and have certain specifications or characteristics known beforehand, what it can be extended to unknown and unobservable parts for unknown and unobservable system's states. Also, in agricultural production [50], static environment (soil moisture, position characteristics, weather...) and human defined factors (irrigation, fertilization, pest control). There are lots of research on predicting the agricultural production from the human defined factors, but these predictions are based on just certain static environment. Considering different static environments, or even, new locations for them, predicting the agricultural production becomes a zero-shot regression task. Another application could also be for monitoring, control and modeling of water treatment for human, industrial, or agricultural water consumption [51]. The ability to incorporate side information derived from the environment opens up a promising range of applications for water treatment prediction and control, such as pollution control, water quality prediction, and water quality prediction. The pioneers of providing zero-shot methods for regression are two preliminary works [48,52] not able to cope with a general-purpose regression zero-shot task [17]. The former [48] reduces the experiments to a single toy example based on a beta distribution with just two features for the side information and one common instance feature. In fact, if one tries to perform experiments over datasets with higher dimensions in features and side information, the available software reports the message "expected 1D vector for x". The latter [52] predicts the future position of a piece that is pushed by a robotic arm, given the present location through deep learning approach specifically built for this purpose, avoiding to be used as a general purpose zero-shot regressor. Quite recently, another work [17] overcomes this issue and has presented two novel approaches with appealing performance in a general-purpose zero-shot regression framework. Both approaches treat side information different from common features [53] and provide unobserved target models through a two-stage procedure. They coincide in providing observed target models in the first stage, whereas they differ in the way side information is taken in the second stage for providing unobserved target models. One of them is a simple relationship approach inspired by the inverse distance weighting. Particularly, predictions using the observed target models are weighted by the similarity of each observed target with the unobserved target in order to provide unobserved target predictions. The main disadvantage of this approach is that side information is just considered a posteriori in the testing stage of the second phase. In addition, it just interpolates predictions of the observed target models, whose values will be bounded in a certain range, then its generalization power is quite limited. The other method arose in an attempt of overcoming these drawbacks. The result was a correspondence method that takes the side information into the learning process of the second phase in order to increase the generalization power of the predictions for the unobserved targets. Particularly, the method learns the parameters of the unobserved target models from both the observed target models and the side information. This alternative actually improves the predictive performance of the unobserved targets. However, an assumption of a linear relationship between features and targets must be established. The main disadvantage of these two approaches is that they deal with side information (information of the surroundings of the meteorological stations) separately (in different phases) from the common features (weather conditions), avoiding a global optimization. At this point, one can notice that, on the one hand, taking side information as common features directly handles zero-shot regression in just one learning process obtaining a global optimization, but it does not provide promising performance [17] because the same treatment of both kinds of information is not a good practice. On the other hand, exploiting side information in a more strategic and specific way taking it separately from the common features, although, so far, into separate phases, has been shown to improve the predictive performance [17], but it does not provide a global optimization. This situation sheds the light of tackling zero-shot regression directly in just one learning process, but adequately handling features and side information, each one differently and according to their nature. The contribution of this paper goes in this direction and the proposal consists of a novel one-stage learning approach for zero-shot regression based on a kernel definition that properly integrates both features and side information in the same learning process. The proposed approach experimentally exhibits its superiority in performance with regard to other existing approaches for zero-shot regression, one of them consisting of treating side information as common features being the other the two the above-mentioned relationship method and correspondence method.

The rest of the paper is organized as follows. Section 2 describes some related work. Section 3 details the zero-shot regression statement and the state-of-the-art methods available in the literature. Then, the new proposal consisting of a one-stage learning process that jointly integrates side information and common features is detailed in Section 4. In Section 5 the description of the experiments and the discussion of the results are exposed. Finally, Section 6 draws some conclusions and proposes some lines of research for future work.

2. Related work

Zero-shot scenario opens the possibility of supposing an inductive rather than transductive learning paradigm with regard to targets [10]. Target inductive learning induces models for generic unknown and unobservable targets, whereas target transductive learning induces models for specific unknown and unobservable targets. Precisely, the own goal that characterizes the zero-shot scenario of inducing models for targets for which instances are not available (unobservable targets) makes possible contemplate both scenarios with regard to the targets. Notice that the side information of both observed and unobserved targets is supposed to be available. Hence, the difference between target transductive and target inductive learning is in fact reflected in whether the side information of the unobserved targets is including in the training phase of unobserved target models (transductive) or not (inductive). This situation does not happen in classical machine learning, for which models are required for just targets with available instances (observable targets). The majority of the works about zero-shot, including ours, assumes inductive learning for targets, but there are some that deal with target transductive learning [54].

The approaches to cope with zero-shot task can be split into instance-based or model-based methods [10,17] independently if they are for a classification task (predicting a label) or a regression task (predicting a continuous value). The former adopts diverse strategies such as extraction or learning in order to provide instances for the unobserved targets, from which unobserved target models are learned afterwards, whereas the latter the unobserved targets are directly learned from the information available. The model-based approaches, in which fall our proposal, can be split into relationship, correspondence and combination methods depending on how they manage to provide unobserved target models [10]. On one hand, both relationship and correspondence approaches learn observed target models. But they differ in the way they exploit the side information. Relationship assumes the existence of a relationship function between observed and unobserved targets. This function together with the observed target models are taken to obtain unobserved target models. However, correspondence methods learn the correspondence between the observed target models and the observed target side information. On the other hand, combination methods decompose the observed and unobserved targets into basic elements, learn a model per each basic element and finally they combine the basic element models via an inference process in order to obtain the unobserved target models.

As commented before, classification into a set of finite classes has extensively been the focus of researches in the zero-shot learning, leaving a lack of approaches to cope with zero-shot regression (predicting continuous values as they are the concentration of the pollutants). In general, classical classification methods has been adaptable and extensible to classical regression or at least it was possible to transfer the ideas established for the classical classification to the classical regression. However, in the context of zero-shot learning, the ideas taken for zero-shot classification are highly influenced by the classes of the classification and not easily extensible for zero-shot regression, even they are unfeasible to transfer. Great efforts have been made to provide a fan of approaches for zero-shot classification overcoming the lack of instances and exploiting the side information. Particularly, the so-called generalized zero-shot classification methods [55,56] have captured the attention of the research in the last years. They arise in an attempt of fill the existing gap of the traditional zero-shot classification methods, which have limited generalization power to the unobservable classes, since they combine semantic (side) information with common features of just the observable classes. Then, the models they induced tend to classify instances of the unobservable classes as belonging to one of the observable classes. Generalized zero-shot classification methods can be split into embedding-based methods, generative-based methods and common space methods. Embedding-based methods learn an embedding space to relate the common (visual in image classification) features of observable classes with their corresponding semantic information (side information) [57–59]. They learn a projection function able to recognize unobservable classes by measuring the similarity level between the semantic (side) information of the observable and unobservable classes in the embedding space. These methods are biased to the observable classes, so their generalization power to unobservable classes is limited. Generative-based methods [60–62] learn a model to generate instances (images) or common (visual) features for the unobservable classes based on the instances of observable classes and semantic (side) information of both kind of classes. By generating instances for unobservable classes, the task is converted into a classical classification task. In one sense, these methods overcome the bias problem of the embedding-based methods, since the models are learned to be able to classify instances from both observable and unobservable classes. Finally, common-space methods [63,64] learn a common representation space into which both common (visual) features and semantic (side) information are projected in order to get an effective knowledge transfer. Latent features are built, which are the ones that contains the whole information coming from the unobservable classes.

Similar frameworks to zero-shot learning are few-shot learning [43, 65,66] and one-shot learning [67,68]. In both scenarios, unobservable targets have some instances available, but a reduced and a limited number of them. The difference lies in that in the former few instances are available whereas in the latter just one instance is available.

Transfer learning [69,70] is a field closely related to zero-shot learning. In transfer learning, the source domain and source task respectively are the observable instances and targets. The counterpart of the unobservable instances and targets are called the target domain and target task. Hence, the aim of transfer learning consists of extracting knowledge from the source domain and task and transferring it to the target domain in order to cope with the target task. Transfer learning also includes inductive and transductive paradigms, but adds the unsupervised option [71]; all defined in terms of properties that the source and target domains and tasks satisfy. Under an inductive paradigm, source and target tasks differ, no matter if the respective domains coincide or not. Under a transductive setting, source and target domains differ, whereas the source and target tasks coincide. Finally, under an unsupervised scenario, the target and source tasks differ as it happens under an inductive paradigm, but the tasks fall into unsupervised learning. In the context of this paper, the source and target task differ, then leading to an inductive transfer learning paradigm. Besides, the source and target domains coincide, since both deal with the same features. More classical transfer learning required the availability of some instances in the target domain whatever inductive or transductive paradigms. Only more recent transfer learning approaches [72–74] cope with situations deprived of instances in the target domain, but unfortunately, they are designed exclusively for classification, which is not easily adaptable to regression. Besides, they are customized for what they were designed, for instance, for outlier detection [73], extracting specific image features [74], or obtaining synthesized dialogue instances [72]. Hence, they are not applicable to a general-purpose zero-shot regression task.

3. Zero-shot regression statement and state-of-the-art methods

This section formally states the zero-shot regression task and also formally discusses the strategies followed by the state-of-the-art methods. The formal statement of the zero-shot regression task will be defined in terms of inductive learning, since, as commented in Section 2, our assumption is that unobserved targets are supposed to be unknown and generic. Let $\mathcal{X}$, $\mathcal{S}$ and $\mathcal{Y}$ respectively denote the feature space of instances, the feature space of targets (side information) and the image space of the predictions. Hence,

- Let $\mathcal{T}^o = \{t_i^o \in \mathcal{S}\}_{i=1}^{m_o}$ be the set of observed targets, where m_o is the number of them. The notation $t_i^o \in \mathcal{S}$ represents the side information of the observed targets, that will be in form of features, as commented in Section 1. Let $t^u \in \mathcal{S}$ be the side information (feature representation) of a generic unobserved target such that $t^u \notin \mathcal{T}^o$.
- Let $\mathcal{D}^o = \{(x_j^o, y_j^o) \in \mathcal{X} \times \mathcal{Y}\}_{j=1}^{n_o}$ be the set of instances for the observed targets, where n_o is the number of them. Then, $\mathcal{D}^{o,t_i^o} \subset \mathcal{D}^o$ will denote the set of instances of the observed target $t_i^o \in \mathcal{T}^o$ for all $i = 1, \dots, m_o$. Let x^u be an instance of the unobserved target $t^u \notin \mathcal{T}^o$ whose prediction is $y^u \in \mathcal{Y}$ such that $(x^u, y^u) \notin \mathcal{D}^o$.

Therefore, the inductive zero-shot regression task consists of learning a function $f : \mathcal{X} \times \mathcal{S} \to \mathcal{Y}$ from $\mathcal{D}^o$ and $\mathcal{T}^o$ able to predict $y^u \in \mathcal{Y}$ for a generic instance $x^u \in \mathcal{X}$ of a generic unobserved target $t^u \in \mathcal{S}$ (in the transductive scenario the function to learn would be $f_{t^u} : \mathcal{X} \to \mathcal{Y}$ from $\mathcal{D}^o$, $\mathcal{T}^o$ and an specific unobserved target t^u able to predict $y^u \in \mathcal{Y}$ for a generic instance $x^u \in \mathcal{X}$, see [17] for more details). The next subsections formally and briefly detail the state-of-the-art methods for zero-shot regression.

3.1. Baseline method

The baseline method (BL) [17] was designed to be a point of reference in an attempt of avoiding ignoring side information. The side information is included as convectional features. Hence, the instances of the same target will have the same values in these features (see Fig. 1(a)). In the training phase, just one learning process takes place. The features are built taking $\mathcal{D}_{\mathcal{X}}^o = \{x_j^o \in \mathcal{X}\}_{j=1}^{n_o}$ (matrix X^o) and $\mathcal{T}^o = \{t_i^o \in \mathcal{S}\}_{i=1}^{m_o}$ (matrix S^o) and joining (concatenating) each target side information $t_i^o \in \mathcal{T}^o$ to their correspondent instances $\mathcal{D}_{\mathcal{X}}^{o,t_i^o} \subset \mathcal{D}_{\mathcal{X}}^o$. The goal consists of inducting a function $f : \mathcal{X} \times \mathcal{S} \to \mathcal{Y}$ taking the correspondent prediction values of $\mathcal{D}_{\mathcal{Y}}^o = \{y_j^o \in \mathcal{Y}\}_{j=1}^{n_o}$ (matrix Y^o). In the testing phase, the features of an unobserved instance x^u (vector X^u) of an unobserved target t^u are joined (concatenated) to the side information of the unobserved target t^u (vector S^u) to feed f and provide the prediction $f(x^u, t^u)$. This method works under the hypothesis of existing an equal kind of relationship between features and targets and between this relationship and the side information, since common features and side information are treated equally. However, this assumption is quite limited and tied, since it is not usually satisfied. In fact, the nature of side information is different from the common features.

3.2. Similarity relationship method

The similarity relationship method (SR) [17] is an appealing method to cope with zero-shot regression due to its simplicity. The main point of this method is the establishment of a relationship function $\delta^{o,u}$ defining a similarity between observed and unobserved targets in terms of the inverse of the distance, that is, $\delta^{o,u}(t^o, t^u) = 1/d(t^o, t^u)$.

The method works in two stages (see Fig. 1(b)). The first stage takes place in the training phase and consists of learning the observed target models $f^o = \{f^{t_i^o}\}_{i=1}^{m_o}$ from instance features $\mathcal{D}_{\mathcal{X}}^o = \{x_j^o \in \mathcal{X}\}_{j=1}^{n_o}$ (matrix X^o) and prediction values $\mathcal{D}_{\mathcal{Y}}^o = \{y_j^o \in \mathcal{Y}\}_{j=1}^{n_o}$ (matrix Y^o), then, ignoring side information $\mathcal{T}^o = \{t_i^o \in \mathcal{S}\}_{i=1}^{m_o}$ (matrix S^o). The second stage happens in the testing phase and it is the stage in which side information $\mathcal{T}^o = \{t_i^o \in \mathcal{S}\}_{i=1}^{m_o}$ (matrix S^o) comes into play. The, giving an unobserved instance x^u (vector X^u) of an unobserved target t^u (vector S^u), the set of prediction values $f^o(x^u) = \{f^{t_i^o}(x^u)\}_{i=1}^{m_o}$ are obtained and aggregated using a normalized weighting procedure induced by the correspondent similarity $\{\delta^{o,u}(t_i^o, t^u)\}_{i=1}^{m_o}$ in order to produce the prediction $f(x^u, t^u)$.

The similarity function based on the inverse of a distance allows guaranteeing that the models of the least similar observed targets to the unobserved target make less influence on the prediction of the unobserved instance than the models of the most similar observed targets. This method has shown to perform well in spite of its simplicity. However, the generalization power may be compromised because the method only interpolates values of the observed target models to provide predictions for the unobserved instances of unobserved targets. Besides, the side information is not included in the learning process; otherwise, it is exploited in the testing phase. Hence, features and side information are exploited separately in two different phases.

3.3. Model parameter learning correspondence method

Model parameter learning correspondence method (MPLC) [17] arose in an attempt of improving the generalization power of SR, including the side information in a learning process. Particularly, the approach tries to learn the correspondence between the observed targets and the correspondent observed targets model parameters, which may potentially increase the generalization power of the unobserved target models (see Fig. 1(c)).

The method also works in two stages, as SR. However, in this case, both stages take place in the training phase, unlike SR, whose second phase takes place in the testing phase. The first stage of MPLC coincides to the first stage of SR. However, the second stage of MPLC differs from the second stage of SR. In this case, another learning procedure takes place whose aim is to learn the parameters of the unobserved target models. The parameters $\Theta = \{(\theta_1^i, \dots, \theta_p^i)\}_{i=1}^{m_o}$ of the observed target models $f^o = \{f^{t_i^o}\}_{i=1}^{m_o}$ resume the relationship between features and targets for the observed targets. These parameters are taken together with the side information $\mathcal{T}^o = \{t_i^o \in \mathcal{S}\}_{i=1}^{m_o}$ (matrix S^o) of the observed targets in the learning process in order to induce the p models $g_\theta = \{g_{\theta_j}\}_{j=1}^{p}$ defined over $\mathcal{S}$.

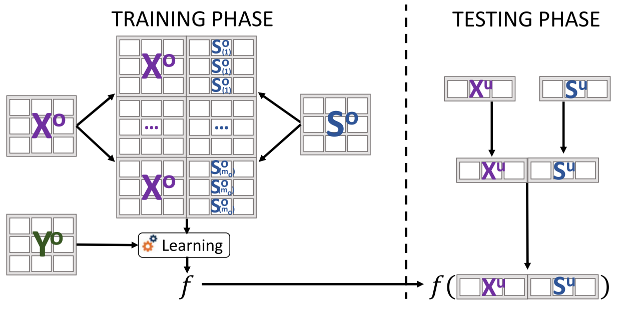
In the testing phase, the side information of an unobserved target t^u (matrix S^u) feeds all the $g_\theta = \{g_{\theta_j}\}_{j=1}^{p}$ to provide the parameters $\{\theta_j^u\}_{j=1}^{p}$ for the model of the unobserved target t^u. Finally, the function $f : \mathcal{X} \times \mathcal{S} \to \mathcal{Y}$ is configured from these parameter value predictions $g_\theta(t^u) = \{\theta_j^u = g_{\theta_j}(t^u)\}_{j=1}^{p}$ leading to a function $f^{g_\theta(t^u)} : \mathcal{X} \to \mathcal{Y}$ that gets ready to evaluate any unobserved instance x^u (matrix X^u) of the unobserved target t^u.

The main disadvantage of this method is that is necessary to assume a linear relationship (see [17] for details) in order to state the learning procedure of the second phase and this assumption do not hold in general.

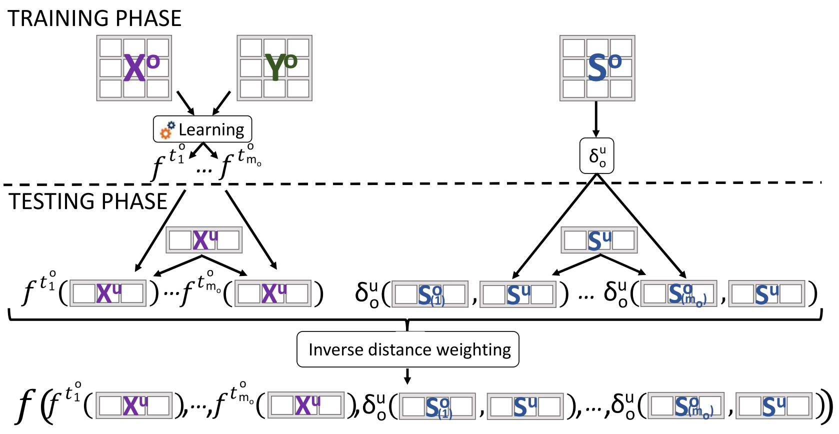
4. Direct side information learning for zero-shot regression

Despite MPLC includes the target side information into a learning process in order to provide more generalization power than SR, it does it in a separate stage different from the observed target model learning process. Hence, learning processes are locally optimized. The proposal of this paper unifies both instance and target (side information) features (all the information available) in a one-stage learning process, then, obtaining a globally optimized learning process, as BL method does it. However, and meanwhile BL method treats equally feature instance description and side information, our novel approach adequately integrates features and side information thought a kernel definition according to the nature of both kinds of information. Therefore, the function $f : \mathcal{X} \times \mathcal{S} \to \mathcal{Y}$ is directly learned from both instance features and side information. If a_x and a_s respectively are the instance feature and side information sizes, $x = (x_1, \dots, x_{a_x})$ is a feature instance description and $s = (s_1, \dots, s_{a_s})$ is a target side information description, let us restrict our proposal to the linear case,[2] (i) in the relationship between features and predictions and (ii) in the relationship between the side information and the relationship of (i) (a non-linear scenario

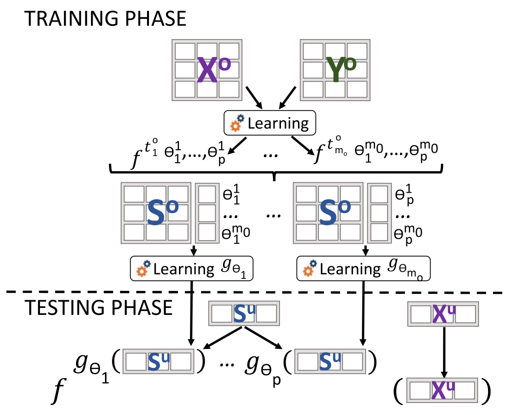
[2] This restriction is motivated by the fact that BL method works better with linear kernel than with quadratic kernel, as it can be seen later on in the experiments.



(a) BL



(b) SR



(c) MPLC

Fig. 1. Baseline, similarity relationship and model parameter learning correspondence methods for zero-shot regression.

will be proposed as future work). Then, $f(x,s)$ adopts the following form

$$\begin{aligned} f: \quad & \mathcal{X}\times\mathcal{S} && \rightarrow && \mathcal{Y}^u \\ & (x,s) && \rightarrow && f(x,s)=f_\beta(s)+\textstyle\sum_{i=1}^{a_x} f_i(s)\cdot x_i \end{aligned} \tag{1}$$

where $f_\beta(s)$ and each $f_i(s)$ are in turn linear functions of s, that is,

$$f_i(s)=\beta_i+\sum_{j=1}^{a_s}\alpha_{i,j}\cdot s_j \qquad i=1,\dots,a_x \tag{2}$$

$$f_\beta(s)=\beta_\beta+\sum_{j=1}^{a_s}\alpha_{\beta,j}\cdot s_j \tag{3}$$

where $\{\beta_i\}_{i=1}^{a_x}$, $\{\alpha_{i,j}\}_{i,j=1,1}^{a_x,a_s}$, β_β and $\{\alpha_{\beta,j}\}_{j=1}^{a_s}$ are the parameters of the linear functions. Then, including the expressions of the Eqs. (2) and (3) in Eq. (1), the expression of $f(x,s)$ will be

$$f(x,s)=\left(\beta_\beta+\sum_{j=1}^{a_s}\alpha_{\beta,j}\cdot s_j\right)+\sum_{i=1}^{a_x}\left(\beta_i+\sum_{j=1}^{a_s}\alpha_{i,j}\cdot s_j\right)\cdot x_i \tag{4}$$

or equivalently

$$f(x,s)=\beta_\beta+\sum_{j=1}^{a_s}\alpha_{\beta,j}\cdot s_j+\sum_{i=1}^{a_x}\beta_i\cdot x_i+\sum_{i=1}^{a_x}\sum_{j=1}^{a_s}\alpha_{i,j}\cdot s_j\cdot x_i \tag{5}$$

At this point, we will define a mapping function ϕ from $\mathcal{X}\times\mathcal{S}$ space into a Hilbert space $\mathcal{H}$ and a linear function g defined over the image space $\mathcal{H}$ of ϕ such that the function f will be expressed as

$$f(x,s)=g(\phi(x,s)) \tag{6}$$

For this purpose, the mapping function ϕ will be

$$\begin{array}{llll}\phi: & \mathcal{X}\times\mathcal{S} & \to & \mathcal{H}\\ & (x,s) & \to & \phi(x,s)=((1,x)^T\otimes_K(1,s)^T)^T\end{array} \tag{7}$$

where $\otimes_K$ denotes the Kronecker product of vectors. The Kronecker product of vectors, also called matrix direct product, means to vectorize the outer product of vectors, denoted by $\otimes_O$. Then, Eq. (7) can be expressed in the following way[3]

$$\phi(x,s)=\left(vec\left((1,x)^T\otimes_O(1,s)^T\right)\right)^T. \tag{8}$$

. Also, the outer product of vectors means to multiply each element of a vector by each element of the other vector. Then, Eq. (8) can be expressed as

$$\phi(x,s)=\left(vec\left((1,x)^T(1,s)\right)\right)^T. \tag{9}$$

Expanding Eq. (9) leads to the following expression of ϕ

$$\begin{aligned}\phi(x,s)&=\left(vec\left(\begin{bmatrix}1\\x_1\\\vdots\\x_{a_x}\end{bmatrix}\begin{bmatrix}1 & s_1 & \cdots & s_{a_s}\end{bmatrix}\right)\right)^T\\&=\left(vec\left(\begin{bmatrix}1 & s_1 & \cdots & s_{a_s}\\x_1 & x_1\cdot s_1 & \cdots & x_1\cdot s_{a_s}\\\vdots & \vdots & \ddots & \vdots\\x_{a_x} & x_{a_x}\cdot s_1 & \cdots & x_{a_x}\cdot s_{a_s}\end{bmatrix}\right)\right)^T\end{aligned} \tag{10}$$

and, finally, performing the vec function and the transpose operator afterwards in Eq. (10), ϕ will be

$$\begin{aligned}\phi(x,s)=\big(&1,\quad x_1,\quad\cdots,\quad x_{a_x},\\&s_1,\quad x_1\cdot s_1,\quad\cdots,\quad x_{a_x}\cdot s_1,\\&\ldots,\ldots,\ldots,\ldots\\&s_{a_s},\quad x_1\cdot s_{a_s},\quad\cdots,\quad x_{a_x}\cdot s_{a_s}\big)\end{aligned} \tag{11}$$

Let us notice that the components of ϕ are the unit and the components of x, that is, $(1,x_1,\ldots,x_{a_x})$, concatenated by these same components multiplied by the first component of s, that is, $(s_1,x_1\cdot s_1,\ldots,x_{a_x}\cdot s_1)$ and so on until the same components multiplied by the last component of s, that is, $(s_{a_s},x_1\cdot s_{a_s},\ldots,x_{a_x}\cdot s_{a_s})$. Then, the components of ϕ are all the monomials of a 2−degree polynomial, except two kinds of monomials, namely, (i) the squared ones of the kind x_i^2 and s_j^2 and (ii) the ones of the kind $x_i\cdot x_j$ and $s_i\cdot s_j$ with $i\neq j$.

Then, the linear function g will be defined as a linear combination of the monomial components of $\phi(x,s)$, that is,

$$g(\phi(x,s))=\beta_\beta\cdot 1+\sum_{i=1}^{a_x}\beta_i\cdot x_i+\sum_{j=1}^{a_s}\left(\alpha_{\beta,j}\cdot s_j+\sum_{i=1}^{a_x}\alpha_{i,j}\cdot x_i\cdot s_j\right) \tag{12}$$

Finally, let us notice that reordering the terms of Eq. (12), one can easily obtain the expression of $f(x,s)$ in Eq. (5). Therefore, $f(x,s)$ can be expressed as a linear function g defined over the image space of ϕ, as stated in Eq. (6). Now, let us propose three different ways of inducing the function f. The first one will consist of applying the mapping function ϕ and inducing the linear function g afterwards according to Eq. (6). The second and third ways define a kernel K that computes the inner product in the image space of ϕ. However, the second way explicitly applies the function mapping ϕ, whereas the third way straightly computes the inner product in the image space and avoids applying the mapping function. The result is a one-phase learning method that directly includes the side information in this learning process. This is the reason why the method is called Direct Side Information Learning (DSIL). Fig. 2 displays the training and testing phases for this new approach. The structure is analogous to that of the BL approach (see Fig. 1(a)), just changing the linear kernel by the new proposed kernels. Next Sections 4.1 and 4.2 deal with the second and third ways of inducing the function f, which both involve a kernel definition. Also, Section 4.3 describes a toy example in order to illustrate the way DSIL works. Finally, Section 4.4 analyses the three ways of inducing the function f in terms of computational cost.

[3] vec is the function that vectorizes a matrix, that is, it converts a matrix into a column vector by concatenating the columns.

4.1. A kernel definition using the mapping function ϕ

This section proposes a kernel associated with the ϕ mapping function in order to establish a gangway from linearity to non-linearity in terms of the inner dot product. The kernel definition allows performing the mapping and the inner product simultaneously. In this case, the kernel can be defined in terms of the linear kernel and ϕ using one of the closure properties of the kernels [75] as

$$K\left(\left(x^{(1)},s^{(1)}\right),\left(x^{(2)},s^{(2)}\right)\right)=\left\langle\phi\left(x^{(1)},s^{(1)}\right),\phi\left(x^{(2)},s^{(2)}\right)\right\rangle \tag{13}$$

Using the expression of the mapping function ϕ exposed in Eq. (8) and dispensing with the vec function, the kernel can be expressed in terms of the outer product of vectors and the Hadamard product (also called the element-wise product, entry wise product or Schur product) of a matrix, which is denoted by $\odot$ and consists of computing the product of the matrixes element-by-element.

$$\begin{aligned}&K\left(\left(x^{(1)},s^{(1)}\right),\left(x^{(2)},s^{(2)}\right)\right)=\sum\left(\left(1,x^{(1)}\right)^T\otimes_O\left(1,s^{(1)}\right)^T\right)\odot\\&\left(\left(1,x^{(2)}\right)^T\otimes_O\left(1,s^{(2)}\right)^T\right)\end{aligned} \tag{14}$$

Computing the outer and Hadamard products and the summation of the elements of the resultant matrix in Eq. (14), the expression of $K\left(\left(x^{(1)},s^{(1)}\right),\left(x^{(2)},s^{(2)}\right)\right)$ becomes

$$\begin{aligned}&K\left(\left(x^{(1)},s^{(1)}\right),\left(x^{(2)},s^{(2)}\right)\right)=1+\sum_{i=1}^{a_x}x_i^{(1)}x_i^{(2)}+\sum_{i=1}^{a_s}s_i^{(1)}s_i^{(2)}\\&+\sum_{i=1}^{a_s}\sum_{j=1}^{a_x}\left(x_j^{(1)}s_i^{(1)}\right)\left(x_j^{(2)}s_i^{(2)}\right)\end{aligned} \tag{15}$$

The main disadvantage of this kernel definition is precisely the necessity of computing the ϕ mapping, since the expression of Eq. (15) must be computed for all pairs (x,s) and in addition it is of quadratic order with regard to the features. Hence, an alternative will be to define the kernel thought an expression able to compute the inner product in the image space of ϕ but avoiding the use of the ϕ mapping and performing a linear order computation instead, which, in fact, it is the well-known interest and advantage of the use of kernels. Next subsection proposes an alternative at this respect.

4.2. A kernel definition using quadratic kernels instead of expanding the mapping function ϕ

This section proposes an alternative expression for the kernel of that of Eq. (15), which will avoid expanding the mapping function ϕ. Let us notice that the image space of ϕ is the set of monomials of degree between 0 and 2 formed with the components of x and s, but removing the monomials with more than one component of x or with more than one component of s, including the squared monomials (see Eq. (11)). This fact sheds light on defining the kernel in terms of the existing quadratic kernel, which adopts the form

$$K_{Q,c}(u,v)=(\langle u,v\rangle+c)^2=\left(\sum_{i=1}^{n}u_iv_i+c\right)^2 \tag{16}$$

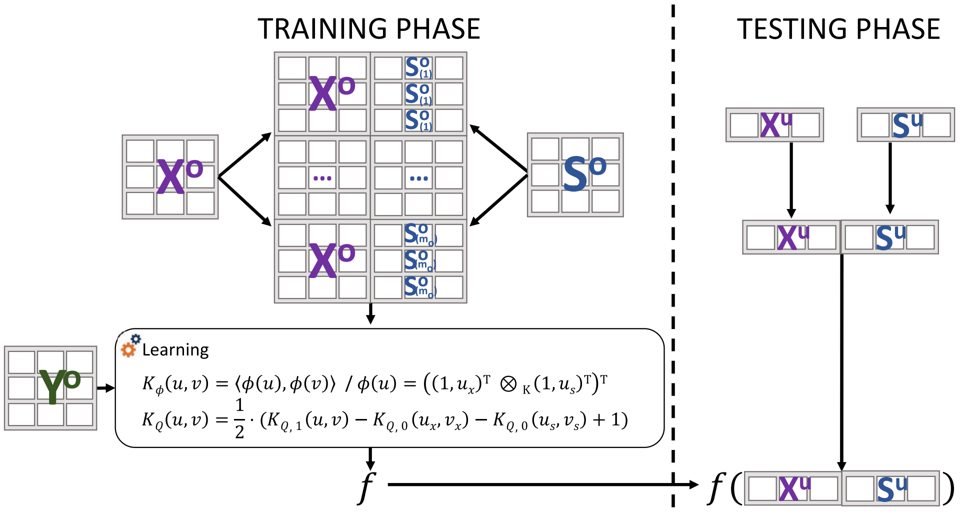

TRAINING PHASE
TESTING PHASE
X^o
S^o
$S^o_{(1)}$
$S^o_{(m_o)}$
Y^o
Learning
$K_\phi(u,v) = \langle \phi(u), \phi(v) \rangle \;/\; \phi(u) = ((1,u_x)^T \otimes_K (1,u_s)^T)^T$
$K_Q(u,v) = \frac{1}{2} \cdot (K_{Q,1}(u,v) - K_{Q,0}(u_x,v_x) - K_{Q,0}(u_s,v_s) + 1)$
f
X^u
S^u
$f(X^u \; S^u)$

Fig. 2. Training and testing phases for the DSIL method.

The quadratic kernel expressed in Eq. (16) computes the inner product in the original space, hence, it is of linear order with regard to the features. The idea is to express Eq. (15) in terms of a sum or a difference of several quadratic kernels, then, obtaining an expression of linear order with regard to the features. Particularly, the alternative kernel expression proposed is the following one

$$K\left(\left(x^{(1)}, s^{(1)}\right), \left(x^{(2)}, s^{(2)}\right)\right) = \frac{1}{2} \cdot \left(K_{Q,1}\left(\left(x^{(1)}, s^{(1)}\right), \left(x^{(2)}, s^{(2)}\right)\right) - K_{Q,0}\left(x^{(1)}, x^{(2)}\right) - K_{Q,0}\left(s^{(1)}, s^{(2)}\right) + 1\right) \tag{17}$$

Effectively, let us now demonstrate that the expression of Eq. (17) is equivalent to the expression of Eq. (15). For this purpose, let us first use the binomial theorem of elementary algebra to Eq. (16). Then, the expression of Eq. (16) becomes

$$K_{Q,c}(u,v) = \left(\sum_{i=1}^{n} u_i v_i\right)^2 + 2c\left(\sum_{i=1}^{n} u_i v_i\right) + c^2 \tag{18}$$

Now, let us apply the multinomial theorem to the first term of the expression of Eq. (18) and regrouping the terms properly. The result is the expression of the Eq. (19)

$$K_{Q,c}(u,v) = \sum_{i=1}^{n}(u_i^2)(v_i^2) + \sum_{i=2}^{n}\sum_{j=1}^{i-1}(\sqrt{2}u_i u_j)(\sqrt{2}v_i v_j) + \sum_{i=1}^{n}(\sqrt{2c}u_i)(\sqrt{2c}v_i) + c^2 \tag{19}$$

Next, and using the expression of Eq. (19), let us expand the terms of Eq. (17)

(i) $K_{Q,1}\left(\left(x^{(1)}, s^{(1)}\right), \left(x^{(2)}, s^{(2)}\right)\right)$, that is, for $c = 1$, $u = \left(x^{(1)}, s^{(1)}\right)$ and $v = \left(x^{(2)}, s^{(2)}\right)$
(ii) $K_{Q,0}\left(x^{(1)},\ x^{(2)}\right)$, that is, for $c = 0$, $u = x^{(1)}$ and $v = x^{(2)}$
(iii) $K_{Q,0}\left(s^{(1)}, s^{(2)}\right)$, that is, for $c = 0$, $u = s^{(1)}$ and $v = s^{(2)}$

Then, expanding these terms will respectively lead to the Eqs. (20), (21) and (22).

$$\begin{aligned} K_{Q,1}\left(\left(x^{(1)}, s^{(1)}\right), \left(x^{(2)}, s^{(2)}\right)\right) &= \left(\langle (x^{(1)}, s^{(1)}), (x^{(2)}, s^{(2)}) \rangle + 1\right)^2 \\ &= \sum_{i=1}^{a_x}\left(x_i^{(1)}\right)^2\left(x_i^{(2)}\right)^2 + \sum_{i=1}^{a_s}\left(s_i^{(1)}\right)^2\left(s_i^{(2)}\right)^2 \\ &+ \sum_{i=2}^{a_x}\sum_{j=1}^{i-1}\left(\sqrt{2}x_i^{(1)}x_j^{(1)}\right)\left(\sqrt{2}x_i^{(2)}x_j^{(2)}\right) \\ &+ \sum_{i=2}^{a_s}\sum_{j=1}^{i-1}\left(\sqrt{2}s_i^{(1)}s_j^{(1)}\right)\left(\sqrt{2}s_i^{(2)}s_j^{(2)}\right) \\ &+ \sum_{i=1}^{a_x}\sum_{j=1}^{a_s}\left(\sqrt{2}x_i^{(1)}s_j^{(1)}\right)\left(\sqrt{2}x_i^{(2)}s_j^{(2)}\right) \\ &+ \sum_{i=1}^{a_x}\left(\sqrt{2}x_i^{(1)}\right)\left(\sqrt{2}x_i^{(2)}\right) + \sum_{i=1}^{a_s}\left(\sqrt{2}s_i^{(1)}\right)\left(\sqrt{2}s_i^{(2)}\right) + 1 \end{aligned} \tag{20}$$

$$K_{Q,0}\left(x^{(1)}, x^{(2)}\right) = \left(\langle x^{(1)}, x^{(2)} \rangle + 0\right)^2 = \sum_{i=1}^{a_x}\left(x_i^{(1)}\right)^2\left(x_i^{(2)}\right)^2 + \sum_{i=2}^{a_x}\sum_{j=1}^{i-1}\left(\sqrt{2}x_i^{(1)}x_j^{(1)}\right)\left(\sqrt{2}x_i^{(2)}x_j^{(2)}\right) \tag{21}$$

$$K_{Q,0}\left(s^{(1)}, s^{(2)}\right) = \left(\langle s^{(1)}, s^{(2)} \rangle + 0\right)^2 = \sum_{i=1}^{a_s}\left(s_i^{(1)}\right)^2\left(s_i^{(2)}\right)^2 + \sum_{i=2}^{a_s}\sum_{j=1}^{i-1}\left(\sqrt{2}s_i^{(1)}s_j^{(1)}\right)\left(\sqrt{2}s_i^{(2)}s_j^{(2)}\right) \tag{22}$$

Therefore, replacing the terms of $K_{Q,1}\left(\left(x^{(1)}, s^{(1)}\right), \left(x^{(2)}, s^{(2)}\right)\right)$, $K_{Q,0}\left(x^{(1)},\ x^{(2)}\right)$ and $K_{Q,0}\left(s^{(1)}, s^{(2)}\right)$ in Eq. (17) by the expressions of the Eqs. (20), (21) and (22) and removing the terms that are annulled, $K\left(\left(x^{(1)}, s^{(1)}\right), \left(x^{(2)}, s^{(2)}\right)\right)$ adopts the following expression

$$K\left(\left(x^{(1)}, s^{(1)}\right), \left(x^{(2)}, s^{(2)}\right)\right) = \frac{1}{2} \cdot \left(\sum_{i=1}^{a_x}\sum_{j=1}^{a_s}\left(\sqrt{2}x_i^{(1)}s_j^{(1)}\right)\left(\sqrt{2}x_i^{(2)}s_j^{(2)}\right) + \sum_{i=1}^{a_x}\left(\sqrt{2}x_i^{(1)}\right)\left(\sqrt{2}x_i^{(2)}\right) + \sum_{i=1}^{a_s}\left(\sqrt{2}s_i^{(1)}\right)\left(\sqrt{2}s_i^{(2)}\right) + 2\right) = \tag{23}$$

Now, simplifying the constant, the expression becomes

$$K\left(\left(x^{(1)}, s^{(1)}\right), \left(x^{(2)}, s^{(2)}\right)\right) = \sum_{i=1}^{a_x}\sum_{j=1}^{a_s}\left(x_i^{(1)}s_j^{(1)}\right)\left(x_i^{(2)}s_j^{(2)}\right) + \sum_{i=1}^{a_x}\left(x_i^{(1)}\right)\left(x_i^{(2)}\right) + \sum_{i=1}^{a_s}\left(s_i^{(1)}\right)\left(s_i^{(2)}\right) + 1 \tag{24}$$

Finally, one can easily observe that the expressions (15) and (24) of $K\left(\left(x^{(1)}, s^{(1)}\right), \left(x^{(2)}, s^{(2)}\right)\right)$ are identical after adequately ordering their terms. Therefore, we have got defining a kernel using the expression of Eq. (16) through a set of quadratic kernels whose computation is of linear order with regard to the features instead of using either the expression of Eq. (15) or (24) which are of quadratic order with regard to the features.

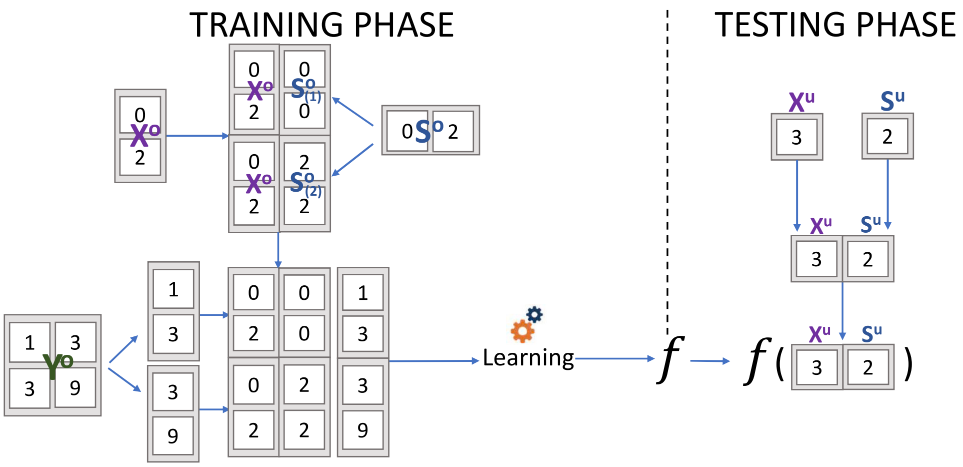


Fig. 3. A toy example that illustrates the direct side information learning method.

4.3. Toy example

This section presents a toy example that illustrates the way the method DSIL works. Fig. 3 shows how DSIL works through a toy example. Let the number of observed targets m_o be equal to 2, whose side information size a_s is equal to 1 (each target is represented by a one-dimensional vector). Let one of the observed targets be $t_1^o = 0$ and the other observed target be $t_2^o = 2$ (both represented in S^o in Fig. 3). Let us consider that each of these two targets has two instances (the same for simplicity), whose number of features a_x is equal to 1 (each instance is represented by a one-dimension vector). Let these two instances be $x_1^o = 0$ and $x_2^o = 2$ (both represented in X^o in Fig. 3). Finally, let us suppose a linear relationship between features x and predictions y, whose coefficients are in turn a linear function of the side information, that is, the linear function to be learned is $y = f(x,s) = \alpha(s) \cdot x + \beta(s)$, where $\alpha(s)$ and $\beta(s)$ are in turn linear functions. Let $\alpha(s) = s+1$ and $\beta(s) = s+1$ for simplicity. Then, the predictions will be 1 and 3 for the two instances of the observed target t_1^o and 3 and 9 for the two instances of the observed target t_2^o (represented in Y^o in Fig. 3). Then, the learning process to induce the function model f takes place. Notice that this schema also fits the method BL changing the kernel in the learning process. Once the function f is learned, for an unobserved instance $x^u = 3$ (represented by X^u in Fig. 3) and for an unobserved target $t^u = 2$ (represented by S^u in Fig. 3) the prediction using f is carried out. For this purpose, the feature description $x^u = 3$ and the side information $t^u = 2$ for an unobserved target is concatenated for feeding f and obtaining $f(x^u, s^u) = f(3,2)$.

Once the toy example is defined, let us compare the method BL with linear kernel (BL_L), the method DSIL and the method BL with quadratic kernel (BL_Q). Fig. 4 displays the learning process of the three methods. Firstly, the function mapping ϕ is applied according to the different kernels that the three methods use. Let us remind that $a_x = a_s = 1$ in the toy example. Then, both x and s are one-dimensioned. The mapping function $\phi(x,s)$ for the method BL_L is $\phi(x,s) = (x,s)$, then, only the monomials x and s are taken. In the case of the method DSIL, the mapping function $\phi(x,s)$ equals $(x, x\cdot s, s, 1)$, then, the monomials that this kernel considers are also x and s, but in addition, the monomials $x \cdot s$ and 1. Finally, the mapping function $\phi(x,s)$ for the kernel of the method BL_Q is $(x^2, x, x\cdot s, s, s^2, 1)$, that is, all possible combinations of monomials of degree 2, namely, x^2, s^2 in addition to x, s, $x \cdot s$ and 1. Once the expansion is carried out, a linear learning takes place in the image space of the mapping function ϕ. The learned models were respectively $f_{\mathrm{BL}_L}(x,s) = 2\cdot x + 2\cdot s$, $f_{\mathrm{DSIL}}(x,s) = x + s + x\cdot s + 1$ and $f_{\mathrm{BL}_Q}(x,s) = 0.2\cdot x + 0.2\cdot s + x\cdot s + 0.4\cdot x^2 + 0.4\cdot s^2 + 1$. The evaluations of these models over both the observed instances and the unobserved instance are shown on the bottom of Fig. 4. This toy example shows that the model induced by BL_L cannot predict neither the observed instances nor the unobserved instance properly. It also shows that the model induced by BL_Q could not have enough generalization power, since it correctly predicts the observed instances of the observed targets, but it fails in the prediction of the unobserved instance of the unobserved target (it overfits).

4.4. Computational complexity analysis

Three different implementations of DSIL have been proposed, namely, directly using the mapping function ϕ (DSIL_ϕ), defining a kernel using the mapping function ϕ (DSIL_{K_ϕ}) and defining a kernel that uses quadratic kernels instead of expanding the mapping function ϕ (DSIL_{K_Q}). All three have been shown to be equivalent, since they induce the same models for the function f. Hence, the performance in terms of accuracy is equal, but they highly differ in terms of computational cost. This section analyses this issue.

DSIL_{K_ϕ} and DSIL_{K_Q} need the same storage requirements than BL. However, DSIL_ϕ requires more storage than BL due to the increasing number of features. Particularly, the number of features changes from $a_x + a_s$ (linear) in the case of BL to $(a_x + 1)\cdot(a_s + 1)$ (quadratic) in the case of DSIL_ϕ due to the mapping function ϕ computation (let us remind that a_x and a_s respectively are the instance feature and side information sizes).

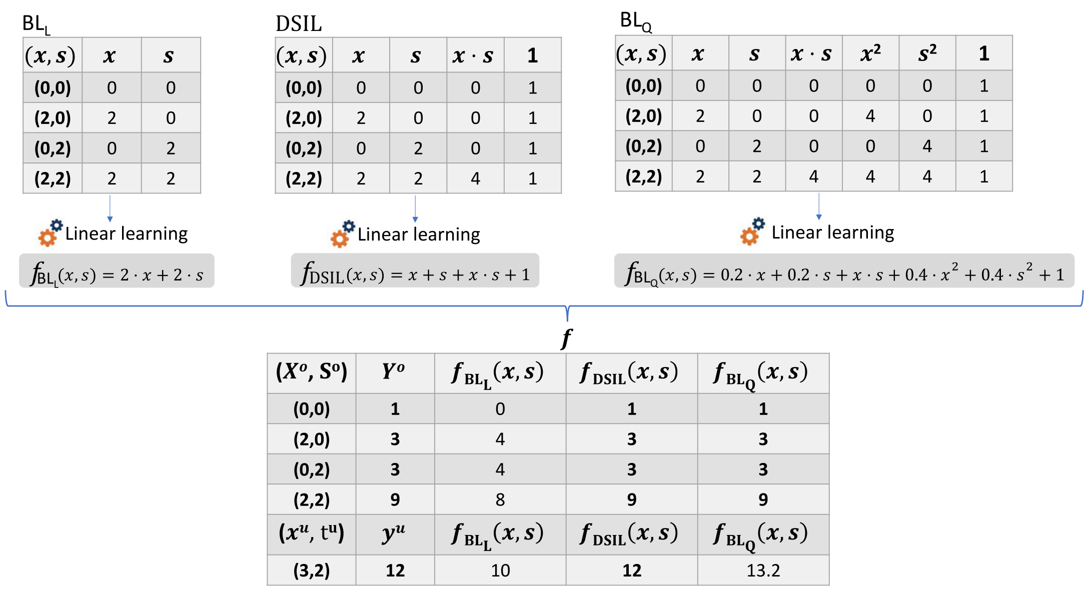
Regarding time complexity, the key of the differences is focused on the kernel evaluation. The kernel is evaluated per each pair of instances. Hence, the time complexity will be $\mathcal{O}(n_o^2)\cdot\mathcal{O}(K)$, where n_o is the number of instances and K is the kernel. Then, let us now analyze the different methods according to the kernel they use:

(i) The BL method when a kernel of linear time complexity is taken, like linear kernel (BL_L) or quadratic kernel (BL_Q), has the following time complexity

$$\mathcal{O}(\mathrm{BL}) = \mathcal{O}(n_o^2)\cdot\mathcal{O}(a_x + a_s) = \mathcal{O}(n_o^2\cdot(a_x + a_s))$$

(ii) DSIL_ϕ has in addition a preprocess apart from the kernel evaluations. This preprocess consists of expanding each instance using the mapping function ϕ obtaining an instance in the image space of ϕ of size $(a_x+1)\cdot(a_s+1)$. Then, the kernel evaluations will take place in the image space of ϕ. Hence, the time complexity will have two terms, one for the preprocess $\mathcal{O}(n_o\cdot((a_x+1)\cdot(a_s+1)))$ and the other for the kernel evaluations $\mathcal{O}(n_o^2)\cdot\mathcal{O}((a_x+1)\cdot(a_s+1))$. Therefore, the time complexity of DSIL_ϕ will be

$$\begin{aligned}\mathcal{O}(\mathrm{DSIL}_\phi) &= \mathcal{O}(n_o\cdot((a_x+1)\cdot(a_s+1))) + \mathcal{O}(n_o^2)\cdot\mathcal{O}((a_x+1)\cdot(a_s+1)) \\ &= \mathcal{O}(n_o\cdot a_x\cdot a_s) + \mathcal{O}(n_o^2\cdot a_x\cdot a_s) = \mathcal{O}(n_o^2\cdot a_x\cdot a_s)\end{aligned}$$



BL$_L$

(x,s)	x	s
(0,0)	0	0
(2,0)	2	0
(0,2)	0	2
(2,2)	2	2

Linear learning

$f_{\mathrm{BL_L}}(x,s) = 2\cdot x + 2\cdot s$

DSIL

(x,s)	x	s	$x\cdot s$	1
(0,0)	0	0	0	1
(2,0)	2	0	0	1
(0,2)	0	2	0	1
(2,2)	2	2	4	1

Linear learning

$f_{\mathrm{DSIL}}(x,s) = x + s + x\cdot s + 1$

BL$_Q$

(x,s)	x	s	$x\cdot s$	x^2	s^2	1
(0,0)	0	0	0	0	0	1
(2,0)	2	0	0	4	0	1
(0,2)	0	2	0	0	4	1
(2,2)	2	2	4	4	4	1

Linear learning

$f_{\mathrm{BL_Q}}(x,s) = 0.2\cdot x + 0.2\cdot s + x\cdot s + 0.4\cdot x^2 + 0.4\cdot s^2 + 1$

f

(X^o, S^o)	Y^o	$f_{\mathrm{BL_L}}(x,s)$	$f_{\mathrm{DSIL}}(x,s)$	$f_{\mathrm{BL_Q}}(x,s)$
(0,0)	1	0	1	1
(2,0)	3	4	3	3
(0,2)	3	4	3	3
(2,2)	9	8	9	9
(x^u, t^u)	y^u	$f_{\mathrm{BL_L}}(x,s)$	$f_{\mathrm{DSIL}}(x,s)$	$f_{\mathrm{BL_Q}}(x,s)$
(3,2)	12	10	12	13.2

Fig. 4. Differences of the methods BL$_L$, DSIL *y* BL$_Q$ through the toy example.

(iii) DSIL$_{K_\phi}$ expands each instance in the kernel evaluation. Hence, the time complexity of the kernel evaluation is $\mathcal{O}((a_x+1)\cdot(a_s+1))$. Then the time complexity of DSIL$_{K_\phi}$ will be

$$\mathcal{O}(\mathrm{DSIL}_{K_\phi}) = \mathcal{O}(n_o^2)\cdot\mathcal{O}((a_x+1)\cdot(a_s+1)) = \mathcal{O}(n_o^2\cdot a_x\cdot a_s)$$

(iv) DSIL$_{K_Q}$ applies the quadratic kernel, which is of the linear order with regard to the features. It evaluates this kernel three times, respectively taken $a_x + a_s$, a_x and a_s features. Then, the time complexity of the kernel evaluation will have three terms $\mathcal{O}(a_x + a_s)$, $\mathcal{O}(a_x)$ and $\mathcal{O}(a_s)$. Therefore, the time complexity of DSIL$_{K_Q}$ will be

$$\mathcal{O}(\mathrm{DSIL}_{K_Q}) = \mathcal{O}(n_o^2)\cdot(\mathcal{O}(a_x+a_s)+\mathcal{O}(a_x)+\mathcal{O}(a_s)) = \mathcal{O}(n_o^2\cdot(a_x+a_s))$$

(iv) SR performs m_o learning procedures in the first phase (there are any learning process in the second phase), one per each target, where an average of n_o/m_o instances are involved in each learning procedure. The kernel evaluation in this case is $\mathcal{O}(a_x)$. Hence, the complexity of the first phase is $m_o\cdot\mathcal{O}((n_o/m_o)^2)\cdot\mathcal{O}(a_x)$, that is,

$$\mathcal{O}(\mathrm{SR}) = \mathcal{O}(n_o^2/m_o\cdot a_x)$$

(v) The first phase of MPLC is identical to the one of SR, then, the complexity of the first phase will be $\mathcal{O}(n_o^2/m_o\cdot a_x)$. The second phase involves p learning process (one per each parameter), each one managing m_o instances (as many as number of targets) and a_s features. Then, the time complexity of the second phase will be $p\cdot\mathcal{O}(m_o^2\cdot a_s)$

$$\mathcal{O}(\mathrm{MPLC}) = \mathcal{O}(n_o^2/m_o\cdot a_x) + \mathcal{O}(p\cdot m_o^2\cdot a_s)$$

Table 1 displays a summary of the time complexity of the algorithms. On the one hand, the use of the mapping function ϕ in the kernel evaluation (DSIL$_\phi$ and DSIL$_{K_\phi}$) involves increasing the time complexity order from linear to quadratic with regard to the number of features a_x and a_s. The expansion that takes place in the preprocess of DSIL$_\phi$ is of linear order with regard to the number of instances. DSIL$_{K_\phi}$ avoids this preprocess of DSIL$_\phi$, but this affects the kernel evaluation of DSIL$_{K_\phi}$ in that DSIL$_{K_\phi}$ must perform the expansion per kernel evaluation, which is of quadratic order with regard to the number of instances. Hence, embedding the mapping function ϕ in the kernel evaluation does not help to improve the time complexity; otherwise, it worsens it, despite both DSIL$_\phi$ and DSIL$_{K_\phi}$ have the same time complexity order. On the other hand, DSIL$_{K_Q}$ is of the same order that BL$_Q$. Hence, it was possible to build a specific kernel for properly treating the side information without increasing the time complexity order. Later on, the experiments will show the improved performance that DSIL$_{K_Q}$ exhibits with regard to BL$_Q$. The complexity of SR and MPLC is lower because they carry out the learning processes only taking the instances of each target each time instead of taking the instances of all the targets together.

Table 1
Summary of time complexity of BL, DSIL$_\phi$, DSIL$_{K_\phi}$ and DSIL$_{K_Q}$ together with SR and MPLC.

Algorithm	Preprocess	Kernel size	Kernel evaluation	Time complexity
BL	–	$\mathcal{O}(n_o^2)$	$\mathcal{O}(a_x+a_s)$	$\mathcal{O}(n_o^2\cdot(a_x+a_s))$
DSIL$_\phi$	$\mathcal{O}(n_o\cdot a_x\cdot a_s)$	$\mathcal{O}(n_o^2)$	$\mathcal{O}(a_x\cdot a_s)$	$\mathcal{O}(n_o^2\cdot a_x\cdot a_s)$
DSIL$_{K_\phi}$	–	$\mathcal{O}(n_o^2)$	$\mathcal{O}(a_x\cdot a_s)$	$\mathcal{O}(n_o^2\cdot a_x\cdot a_s)$
DSIL$_{K_Q}$	–	$\mathcal{O}(n_o^2)$	$\mathcal{O}(a_x+a_s)$	$\mathcal{O}(n_o^2\cdot(a_x+a_s))$
SR	–	$\mathcal{O}(n_o^2/m_o)$	$\mathcal{O}(a_x)$	$\mathcal{O}(n_o^2/m_o\cdot a_x)$
MPLC	–	$\mathcal{O}(n_o^2/m_o)$ $\mathcal{O}(m_o^2)$	$\mathcal{O}(a_x)$ $\mathcal{O}(p\cdot a_s)$	$\mathcal{O}(n_o^2/m_o\cdot a_x)+\mathcal{O}(p\cdot m_o^2\cdot a_s)$

5. Experiments

This section describes the experiments that were carried out in order to compare the approaches. Particularly, the BL, SR, MPLC and DSIL approaches were compared. Besides, both implementations of DSIL were also compared in terms of computational time. Section 5.1 detailed the datasets taken for the experiments. In Section 5.2 the parameter settings are established. Finally, Section 5.3 analyses and discusses the results.

5.1. Description of datasets

There hardly are available datasets with side information in the literature for zero-shot regression. Despite several applications can fit this scenario, as it was illustrated in Section 1, the problem is that the benchmark datasets do not include side information because it could be ignored or even it has not been collected. Hence, and before applying the methods to the air pollution datasets from the Principality of Asturias of Spain, some artificial datasets were designed in order

Table 2
Number of targets and side information size of the artificial datasets. The $R^{k,l}$ datasets reproduce a linear dependence of the side information, whereas $S^{k,l}$ datasets simulate a similarity measure, where k and l respectively are the number of targets and the side information size.

Dataset	m_o	a_s	Dataset	m_o	a_s	Dataset	m_o	a_s
$\{R, S\}^{5,5}$	5	5	$\{R, S\}^{5,15}$	5	15	$\{R, S\}^{5,25}$	5	25
$\{R, S\}^{10,5}$	10	5	$\{R, S\}^{10,15}$	10	15	$\{R, S\}^{10,25}$	10	25
$\{R, S\}^{50,5}$	50	5	$\{R, S\}^{50,15}$	50	15	$\{R, S\}^{50,25}$	50	25
$\{R, S\}^{100,5}$	100	5	$\{R, S\}^{100,15}$	100	15	$\{R, S\}^{100,25}$	100	25

to exhaustively analyze the performance of the methods. The number of targets and the side information size were varied to build different artificial datasets in order to study the effect of the methods under different values of these parameters.

5.1.1. Artificial datasets

The artificial datasets are built following the same guidelines of [17]. They only differ in the number of instances in order to make the BL_Q computationally feasible, but the conclusions reported in [17] remains. A uniform distribution in the range of $(-2;-1] \cup [+1;+2)$ was taken to provide the feature values of the instances x and the side information s (feature values of the target description) due to avoid zero values or values near zero. Then, given the features of an instance $x = (x_i, \dots, x_{a_x})$ and the side information of a target $s = (s_i, \dots, s_{a_s})$, prediction value y for this instance is built as a linear function of x, whose coefficients $\{\alpha_i(s)\}_{i=1}^{a_x}$ are in turn functions of s, that is,

$$y = \sum_{i=1}^{a_x} (\alpha_i(s) \cdot x_i) + \beta$$

Two different ways of obtaining $\{\alpha_i(s)\}_{i=1}^{a_x}$ coefficients have been addressed in order to cover the domains that the methods SR, MPLC and DSIL are able to encompass.[4] On the one hand, datasets will simulate the most general structure that provides a linear dependence of side information. On the other hand, the target will be generated in terms of a similarity measure δ applied to the side information s of the target and a set of other side information descriptions $\{\mu^k\}_{k=1}^{d}$ (see [17] for more details). Formally,

$$\alpha_i(s) = \sum_{j=1}^{a_s} (\gamma_{i,j} \cdot s_j) + \beta_i \qquad \alpha_i(s) = \frac{\sum_{k=1}^{d} \left(\tau_{i,k} \cdot \delta(s, \mu^k)\right)}{\sum_{k=1}^{d} \delta(s, \mu^k)}$$

The same uniform distribution in the range of $(-2;-1] \cup [+1;+2)$ as for the feature values of x and s was taken for obtaining β, $\{\beta_i\}_{i=1}^{a_s}$, $\{\gamma_{i,j}\}_{i=1,j=1}^{a_x,a_s}$, $\{\tau_{i,k}\}_{i=1,k=1}^{a_x,d}$ and $\{\mu^k\}_{k=1}^{d}$ coefficients. The similarity function δ has been randomly chosen to be either the Manhattan (L1 norm) or the Euclidean (L2 norm) in equal shares in order to avoid bias.

The number of instances n_o and features a_x were respectively fixed to 500 (5000 in [17]) and 50 (the same value as in [17]) in both ways of obtaining $\{\alpha_i(s)\}_{i=1}^{a_x}$. The number of targets m_o considered were $5, 10, 50$ and 100, and the side information size a_s were $5, 15$ and 25, in order to cover a range both below and above the real datasets (the same values as in [17]). Table 2 shows the artificial datasets taken. The $R^{k,l}$ datasets reproduce a linear dependence of the side information, whereas $S^{k,l}$ datasets simulate a similarity measure, where k and l respectively are the number of targets m_o and the side information size a_s.

Other artificial datasets were generated in order to compare the computational time of the different implementations of DSIL. Let us make vary the number of instances (n_o) and features (a_x) together with the number of targets (m_o) and side information size (a_s). Let us remind that the number of features that feed the DSIL approach is the number of features (a_x) plus the side information size (a_s), that is, $a_x + a_s$. Besides, the number of instances that feed the DSIL approach is the number of instances (n_o) multiplied by the number of targets (m_o), that is $n_o \cdot m_o$ (assuming that the same instances are shared by all the targets, for simplicity). Hence, a range of representative values for each a_x, a_s, n_o and m_o were taken to perform the comparison. Particularly, both a_x and a_s took values 10, 100, 250 and 500, whereas the values for n_o were 10, 20, 30 and 40 and the values for m_o were 5, 10, 15 and 20. Then, the values for the number of features that feed the DSIL approach ($a_x + a_s$) were 20, 200, 500 and 1000, whereas the values for the number of instances ($n_o \cdot m_o$) were 50, 200, 450 and 800.

[4] This process has not been done for the BL method, since it would mean that the side information would not reproduce the relationship between the instances and targets; otherwise it would lead to a general-purpose regression task and not a general-purpose zero-shot regression task.

Table 3
Properties of the air pollution datasets.

Dataset	n_o	a_x	m_o	a_s
NO_2, PST, NO, SO_2, CO, O_3	41 325	12	5	16

5.1.2. Air pollution datasets

A total of 11 pollutants are collected every 15 min (and hourly averaged) between 2010 and 2018 in 18 pollution and meteorological stations located in the Principality of Asturias, Spain. However, just 6 pollutants (NO_2, PST, NO, SO_2, CO and O_3) and 5 stations ($m_o = 5$) are taken to get the largest common set of pollutants and stations, since not all stations collect all the pollutants. The instance features are a total of 12 ($a_x = 12$) hourly averaged weather conditions, such as wind direction according to Cartesian axes, the season or the precipitation. The goal is to predict the hourly pollutant concentration. The stations are the targets for which surrounding characteristics were collected to form the side information. The fact is that environmental experts [3,4] argue that pollutant concentration not only depends on the weather conditions; otherwise it is highly influenced by the environment where it is gathered. Particularly, the surrounding information collected consisted of establishing if an urban center or highway or factory or sea (or river) is nearby the station in the North, South, East and/or West directions. Then, a total of 16 features (4 cardinal points multiplied by 4 possible kinds of surroundings) have formed the side information ($a_s = 16$). Hence, the goal is to predict pollutant concentration from certain weather conditions taken in a station for which weather conditions have not been collected (unobserved station). Table 3 displays the properties of the 6 pollutants (NO_2, PST, NO, SO_2, CO and O_3) datasets.

5.1.3. Communities and crime dataset

The Communities and Crime dataset of the UCI Machine Learning Repository[5] includes socio-economic data from the 1990 US Census, law enforcement data from the 1990 US LEMAS survey and crime data from the 1995 FBI UCR. The instances are counties of the USA and the target to predict is the violent (murder, rape, robbery and assault) crime ratio in these counties. The original version of this dataset was adapted to be suitable for zero-shot regression. Particularly, the instances are counties whose features are socio-economic characteristics. The targets are the states that are divided into counties. The side information for the states is also socio-economic features. Hence, the goal is to predict the crime ratio for a county that belongs to an unobserved state. A total of 1420 counties (n_o) described by 101 features (a_x) and belonging to 13 states (m_o) described by 15 features (a_s) conform this dataset for predicting the crime ratio.

[5] https://archive.ics.uci.edu/ml/index.php

5.2. Parameter settings

All the approaches were implemented in Python using *Scikit Learn Library*.[6] All the code is available in *GitHub repository*.[7] Support Vector Regression (SVR)[8] is required in order to make possible the use of a quadratic kernel in the BL approach and also to implement the versions of DSIL that involve kernels. Hence, SVR is taken for all the approaches in order to get a fair comparison. BL_L and BL_Q are the BL approach respectively using linear and quadratic kernels. Only these kernels are considered, since DSIL is in between both, that is, DSIL considers the monomials of order 1, as the linear kernel, but it also includes some monomials of order 2, but not all possible monomials of order 2 that the quadratic kernel includes. SR_E and SR_M are the SR approach respectively using the typical Euclidean (L2 norm) and Manhattan (L1 norm) distances to define the relationship between observed and unobserved targets. MPLC has just one version. Finally, DSIL admits three different implementations, namely DSIL_ϕ, DSIL_{K_ϕ} and DSIL_{K_Q}, which respectively are the implementation that computes ϕ mapping directly, the implementation that computes the kernel using ϕ mapping and the implementation that computes the kernel though a linear combination of quadratic kernels. However, the three implementations report the same performance scores since, in fact, it is the same method. Then, they just differ in the computational time that will be also compared. The hyperparameter c of SVR was optimized through a grid search procedure taking the values $\{10^{-3}, 10^{-2}, \dots, 10^{2}, 10^{3}\}$ for c and using a 3-fold cross validation for estimating the mean squared error. The score for comparing the approaches was the relative mean squared error computed following again a 3-fold cross validation. The cross validation performed under the existence of side information is slightly different from the common cross validation. Under this paradigm, not only instances are taken into account; otherwise, both instances and targets must be considered. Besides, some prediction values for the testing fold must be discarded for consistency. These values are (i) for the observed (training) targets and unobserved (testing) instances and (ii) for the unobserved (testing) targets and observed (training) instances (see [17] for more information). None cross-validation was performed for comparing the computational time of the DSIL approaches (DSIL_ϕ, DSIL_{K_ϕ} and DSIL_{K_Q}) in order to avoid the little differences in number of instances and targets that may be in the different folds. A training-test procedure repeated 3 times was carried out instead. A Friedman-Nemenyi test [76] was performed. The Friedman test is a non-parametric hypothesis test that ranks all algorithms for each data set separately. If the null-hypothesis (all ranks are not significantly different) is rejected, the Nemenyi test is adopted as the post-hoc test. According to the Nemenyi test, the performance of two algorithms is considered significantly different if the corresponding average ranks differ by at least the so-called critical difference.

5.3. Result analysis and discussion

This section analyses and discusses the results of the experiments.

5.3.1. Artificial datasets

Table 4 displays the mean relative square error and Friedman ranks for artificial datasets using BL_L, BL_Q and DSIL. The interest of this comparison lies in finding out to what extent the DSIL approach reproduces a linear relationship between features and targets with regard to a linear approach (BL_L) and a quadratic approach (BL_Q). Clearly, DSIL outperforms both BL_L and BL_Q. Besides, the differences are statistically significant at the level of 95% in the case of the BL_Q for both kinds of artificial datasets and in the case of BL_L for $R^{k,l}$ datasets, since the critical difference of the Nemenyi test at this level is 0.95. The exception is with regard to BL_L for $S^{k,l}$ datasets, for which DSIL and BL_L perform highly similar. In this case, DSIL and BL_L split up the best performance. DSIL outperforms BL_L when there are many targets, whereas BL_L gets the best performance when the number of targets is

Table 4
Mean relative square error and Friedman ranks for artificial datasets using BL_L, BL_Q and DSIL.

Dataset	BL_L	BL_Q	DSIL
$R^{5,5}$	112.03(2)	113.85(3)	**58.55**(1)
$R^{10,5}$	131.46(3)	0.04(2)	**3.32E−12**(1)
$R^{50,5}$	108.35(3)	4.70E−05(2)	**8.61E−15**(1)
$R^{100,5}$	111.40(3)	2.58E−05(2)	**2.54E−15**(1)
$R^{5,15}$	115.68(3)	89.13(2)	**78.73**(1)
$R^{10,15}$	184.42(3)	72.66(2)	**61.71**(1)
$R^{50,15}$	100.68(3)	2.02E−05(2)	**2.00E−14**(1)
$R^{100,15}$	117.02(3)	1.30E−05(2)	**3.21E−15**(1)
$R^{5,25}$	**104.87**(1)	107.74(3)	105.50(2)
$R^{10,25}$	194.14(3)	80.82(2)	**65.25**(1)
$R^{50,25}$	102.47(3)	1.11E−04(2)	**6.47E−13**(1)
$R^{100,25}$	108.13(3)	1.34E−05(2)	**4.89E−15**(1)
Avg. Rank	(2.75)	(2.17)	(**1.08**)
Dataset	BL_L	BL_Q	DSIL
$S^{5,5}$	**6.07**(1)	114.58(2)	136.56(3)
$S^{10,5}$	2.97(2)	76.97(3)	**2.45**(1)
$S^{50,5}$	1.74(2)	85.13(3)	**0.41**(1)
$S^{100,5}$	1.42(2)	73.45(3)	**0.20**(1)
$S^{5,15}$	**0.20**(1)	107.15(3)	63.40(2)
$S^{10,15}$	**0.33**(1)	87.87(3)	70.75(2)
$S^{50,15}$	0.31(2)	89.43(3)	**0.04**(1)
$S^{100,15}$	0.38(2)	93.31(3)	**0.03**(1)
$S^{5,25}$	**0.63**(1)	69.63(3)	54.71(2)
$S^{10,25}$	**1.96**(1)	71.69(3)	54.55(2)
$S^{50,25}$	1.70(2)	83.00(3)	**0.50**(1)
$S^{100,25}$	1.21(2)	89.55(3)	**0.16**(1)
Avg. Rank	(1.58)	(2.92)	(**1.50**)
Mean rank	(2.17)	(2.55)	(**1.29**)

Table 5
Mean relative square error and Friedman ranks for artificial datasets using SR_E, SR_M, MPLC, and DSIL.

Dataset	SR_E	SR_M	MPLC	DSIL
$R^{5,5}$	105.76(3)	102.90(2)	111.74(4)	**58.55**(1)
$R^{10,5}$	115.20(4)	105.75(3)	3.61E−08(2)	**3.32E−12**(1)
$R^{50,5}$	72.04(4)	67.05(3)	1.84E−10(2)	**8.61E−15**(1)
$R^{100,5}$	78.90(4)	72.72(3)	9.79E−11(2)	**2.54E−15**(1)
$R^{5,15}$	105.26(4)	104.94(3)	88.97(2)	**78.73**(1)
$R^{10,15}$	197.03(3)	197.18(4)	85.73(2)	**61.71**(1)
$R^{50,15}$	93.10(4)	89.17(3)	2.97E−10(2)	**2.00E−14**(1)
$R^{100,15}$	104.29(4)	101.45(3)	1.06E−10(2)	**3.21E−15**(1)
$R^{5,25}$	**101.56**(1)	101.60(2)	107.88(4)	105.50(3)
$R^{10,25}$	157.36(4)	156.97(3)	87.09(2)	**65.25(1)**
$R^{50,25}$	100.03(4)	97.81(3)	6.03E−09(2)	**6.47E−13**(1)
$R^{100,25}$	102.22(4)	100.21(3)	1.22E−10(2)	**4.89E−15**(1)
Avg. Rank	(3.58)	(2.92)	(2.33)	(**1.17**)
Dataset	SR_E	SR_M	MPLC	DSIL
$S^{5,5}$	**6.73**(1)	6.84(2)	136.56(3.5)	136.56(3.5)
$S^{10,5}$	**2.30**(1)	2.33(2)	6.79(4)	2.45(3)
$S^{50,5}$	1.33(4)	1.30(3)	**0.28**(1)	0.41(2)
$S^{100,5}$	1.10(4)	1.08(3)	**0.20**(1.5)	**0.20**(1.5)
$S^{5,15}$	**0.19**(1)	0.20(2)	109.21(4)	63.40(3)
$S^{10,15}$	**0.30**(1.5)	**0.30**(1.5)	70.73(3)	70.75(4)
$S^{50,15}$	0.29(3.5)	0.29(3.5)	**0.04**(1.5)	**0.04**(1.5)
$S^{100,15}$	0.35(4)	0.34(3)	**0.03**(1.5)	**0.03**(1.5)
$S^{5,25}$	**0.65**(1)	0.66(2)	54.71(3.5)	54.71(3.5)
$S^{10,25}$	**1.90**(1.5)	**1.90**(1.5)	54.55(3.5)	54.55(3.5)
$S^{50,25}$	1.52(4)	1.51(3)	**0.50**(1.5)	**0.50**(1.5)
$S^{100,25}$	1.16(4)	1.15(3)	**0.16**(1.5)	**0.16**(1.5)
Avg. Rank	(2.54)	(**2.46**)	(2.50)	(2.50)
Mean rank	(3.06)	(2.69)	(2.42)	(**1.84**)

[6] https://scikit-learn.org/stable/.
[7] https://github.com/UO231492/DSILZSR.
[8] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html.

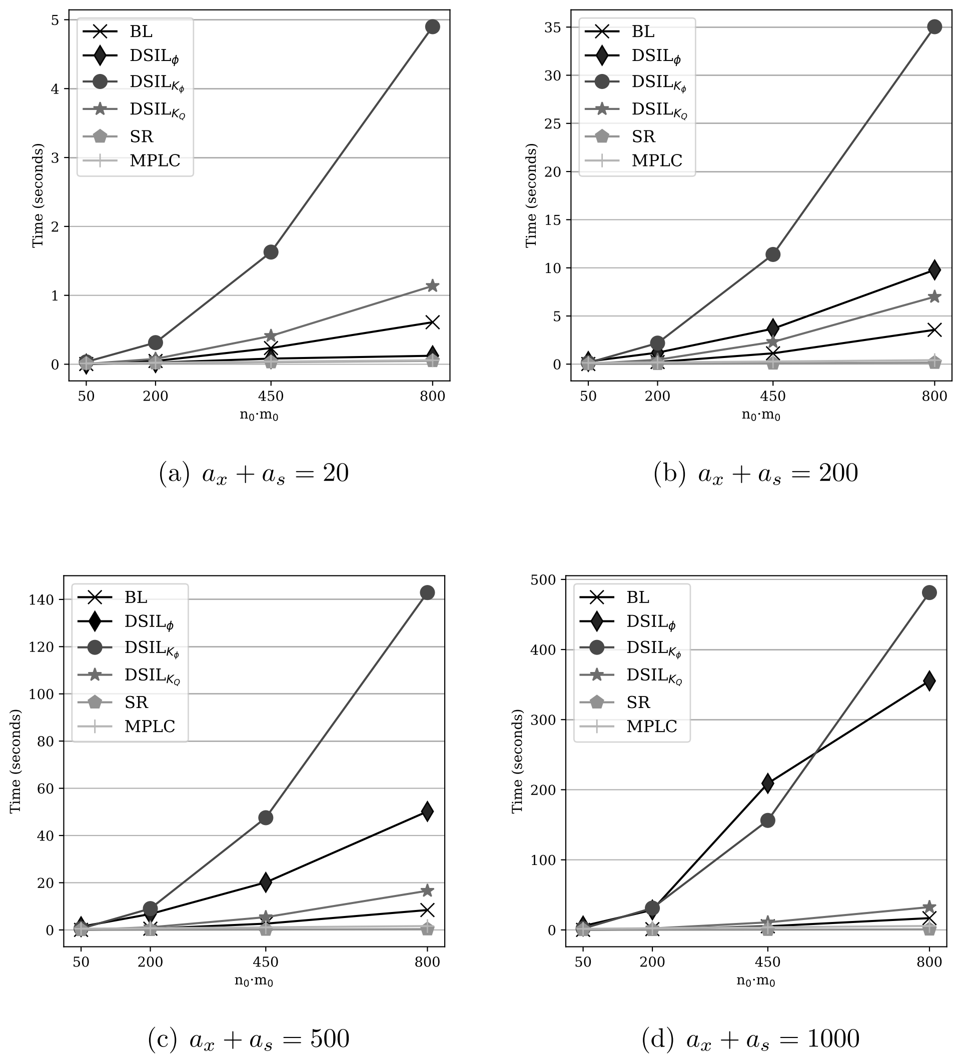


Fig. 5. Time (in seconds) of BL$_Q$, DSIL$_\phi$, DSIL$_{K_\phi}$ and DSIL$_{K_Q}$ together with SR and MPLC when the number of instances that feed the approaches varies and for different values of $a_x + a_s$.

low. Globally, DSIL is also statistically better than both BL$_L$ and BL$_Q$ at the significant level of 95% with a critical difference of the Nemenyi test of 0.67.

Table 5 displays the mean relative square error and Friedman ranks for artificial datasets using SR$_E$, SR$_M$, MPLC, and DSIL. DSIL exhibits the best performance for $R^{l,k}$ datasets. Besides, DSIL is significantly better than both SR$_E$ and SR$_M$ at the significant level of 95%, since the critical difference is 1.35. However, all the approaches perform highly similar for $S^{l,k}$ datasets, where SR$_M$ is slightly better than the rest. In certain sense, this result is expected because of the way the artificial datasets were built. In any case, DSIL continues providing the best global performance. However, the differences in global performance are only significant in the case of SR$_E$ at the level of 95%, for which the critical difference is 0.95. Reducing the significant level to 90%, DSIL provides significant differences with regard to both SR$_E$ and SR$_M$ because the critical difference in this case is 0.85. Finally, there are no significant differences between DSIL and MPLC.

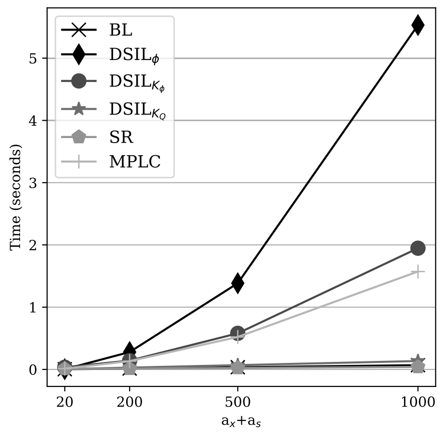
5.3.2. Real datasets

Table 6 exhibits the mean relative square error and Friedman ranks for air pollution datasets and the Communities and Crime dataset of the UCI Machine Learning repository using BL$_L$, BL$_Q$, SR$_E$, SR$_M$, MPLC and DSIL. Clearly, DSIL provides the best performance for most of the real datasets. The exceptions are the SO_2 and O_3 pollutant dataset, for which SR$_M$ slightly outperforms DSIL. The critical differences at the level of 99%, 95% and 90% respectively are 3.93, 3.37 and 3.06. Hence, DSIL is significantly better than BL$_Q$ at the level of 90% and 95% even though BL$_Q$ is slighly better than DSIL for the Communities and Crime dataset. Despite DSIL outperforms all the methods, SR$_M$ and MPLC also reach good performance with regard to BL$_Q$, BL$_L$ and SR$_E$. These results confirm the hypothesis that properly treating side information will lead to an improvement in performance. However, as it can be seen, it varies from one pollutant to another, probably because the relationship between features and targets and between this relationship and the side information varies from one pollutant to another.

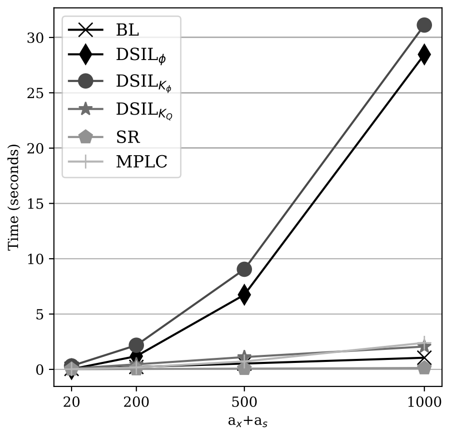
5.3.3. Computational time analysis of DSIL$_\phi$, DSIL$_{K_\phi}$ and DSIL$_{K_Q}$

Figs. 5 and 6 show the computational time (in seconds) employed by the three implementations of DSIL (DSIL$_\phi$, DSIL$_{K_\phi}$ and DSIL$_{K_Q}$) over the specific datasets built in order to carry out this analysis. It also shows the computational time of BL, SR and MPLC. Particularly, Fig. 5 shows the computational time varying the number of instances ($n_o \cdot m_o$) for the different values of the number of features ($a_x + a_s$). Conversely, Fig. 6 displays the computational time varying the number of features ($a_x + a_s$) for the different values of the number of instances ($n_o \cdot m_o$).

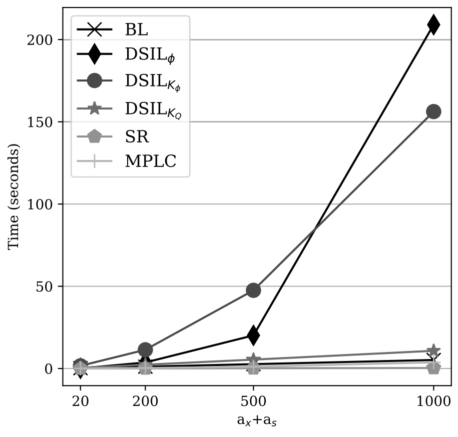
On the one hand, DSIL$_{K_\phi}$ clearly reaches the worst computational time as the number of instances increases (see Fig. 5). In fact, this behavior was expected because, as commented before, the interest in



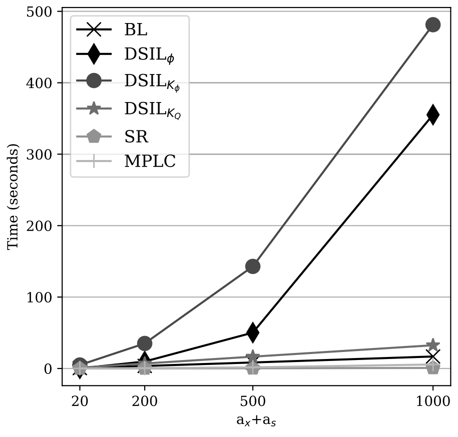
(a) $n_o \cdot m_o = 50$



(b) $n_o \cdot m_o = 200$



(c) $n_o \cdot m_o = 450$



(d) $n_o \cdot m_o = 800$

Fig. 6. Time (in seconds) of BL$_Q$, DSIL$_\phi$, DSIL$_{K_\phi}$ and DSIL$_{K_Q}$ together with SR and MPLC when the number of features that feed the approaches varies and for different values of $n_o \cdot m_o$.

Table 6
Mean relative square error and Friedman ranks for air pollution datasets using BL$_L$, BL$_Q$, SR$_E$, SR$_M$, MPLC and DSIL.

Dataset	BL$_L$	BL$_Q$	SR$_E$	SR$_M$	MPLC	DSIL
NO2	86.11(5)	117.49(6)	81.07(4)	76.52(3)	76.43(2)	**71.70**(1)
NO	90.86(4)	100.42(6)	91.17(5)	88.95(3)	87.15(2)	**86.64**(1)
PST	96.20(5)	98.71(6)	93.84(4)	90.28(2)	91.68(3)	**89.45**(1)
SO2	95.01(4)	100.95(6)	95.53(5)	**92.19**(1)	93.49(3)	92.87(2)
CO	90.08(2)	100.11(6)	99.27(5)	98.48(4)	90.22(3)	**85.36**(1)
O3	68.91(2)	100.50(6)	69.83(3)	**68.10**(1)	90.50(5)	76.79(4)
Crime	33.36(4)	**24.39**(1)	33.43(5)	35.86(6)	25.40(3)	24.57(2)
Avg. Rank	(3.71)	(5.29)	(4.43)	(2.86)	(3.00)	(**1.71**)

using kernels is precisely to avoid computing the kernel via ϕ mapping. On the other hand, dispensing with kernels (DSIL$_\phi$) is the best option for low values of features, but soon, the computational time starts to rocket as the number of features increases (see the evolution of the computational time of DSIL$_\phi$ from Fig. 5(a) to Fig. 5(d)). Indeed, the graph of DSIL$_\phi$ is quite closer to DSIL$_{K_\phi}$ for the highest value of the number of features ($a_x + a_s = 1000$). Finally, DSIL$_{K_Q}$ is the steadier option whatever the number of features is.

Varying the number of features (see Fig. 6), the implementations of DSIL that apply the ϕ mapping (DSIL$_\phi$ and DSIL$_{K_\phi}$) have a quadratic order computational time, whereas the computational time is of linear order in case of the implementation defined though a linear combination of quadratic kernels (DSIL$_{K_Q}$). Also, DSIL$_{K_\phi}$ is highly affected by whatever value of the number of instances (see the evolution of the computational time of DSIL$_{K_\phi}$ from Fig. 6(a) to Fig. 6(d) and especially from $n_o \cdot m_o = 50$ to $n_o \cdot m_o = 200$).

SR and MPLC spent the lowest time, since, as commented before, they do not learn taken the instances of all the targets together, but they perform a learning procedure per target only taking instances of this target.

6. Conclusions and future work

This paper proposes a one-phase method for zero-shot regression, a task whose goal is to induce models for predicting continuous values for unobserved targets, a kind of targets for which instances have not been collected. Under this lack of instances, zero-shot regression takes advantage of the existence of side information, which is neither features nor targets, but provides information about the relationship between features and targets. The straightforward way of exploiting such information consists of taking them as common features in a unique process.

However, some recent approaches have empirically shown that side information deserves to be mined in a different way than the way common instance features are treated. These approaches firstly learn models for the observed targets just taking into account the instance description features and they secondly integrate side information with these models, leading to a two-phase process. The main disadvantage of these approaches is precisely the existence of two different phases, which leads to locally optimize the exploitation of the information instead of as a whole. This paper proposes an alternative in between, that is, it deals with zero-shot regression in a one-phase procedure integrating instance description features and side information simultaneously in order to get a global optimal process, but treating side information properly according to its nature and in any case in a different way that instance features are treated. The method is firstly defined in terms of the existing but unknown relationship between features and targets and between this relationship and side information. Then, a mapping function is deduced from this definition. However, the high storage requirements of this definition make necessary look for an alternative. For this purpose, the method is then built on the basis of a kernel designed from that mapping function. The main drawback of this kernel is its own definition in terms of the mapping function, which incurs in a high computational cost. In fact, the aim of the kernels is precisely to avoid explicitly applying the mapping function and to perform the inner product in the image space of the mapping function instead. Hence, another alternative definition is provided for the kernel in terms of the existing quadratic kernel, providing an implementation of linear instead of quadratic order with regard to the number of features.

Several experiments over both artificial and real datasets exhibit the superiority of the novel approach with regard to other recent existing approaches, being statistically significant with regard to some of them. Additional experiments were accomplished to compare the computational time of the different implementations of the new method (the one consisting of directly mapping the instances using and assuming a linear relationship in the image space of the mapping afterwards, the one consisting of defining a kernel via the mapping function and the one consisting of defining the kernel via the existing quadratic kernel). The conclusion is that the computational time is much steadier if the kernel is defined in terms of the existing quadratic kernel both varying the number of instances or features.

In future work, the plan is to extend the kernel design beyond the linear scenario. This means to cope with non-linearity both in the relation between targets and features and between the side information and the observed target model parameters. Adapting the method for classification or even ordinal regression is neither a trivial nor straightforward task, since it would means to provide predictions for new unobserved classes, something that a simple change of the kernel in the method never provides. On the other hand, a zero-shot multi-regression scenario can be contemplated as a task for future research, where more than one continuous value is predicted simultaneously. Extending the method to this scenario is quite easy but entails studying how the side information influences the relationship between the multiple outputs. Another context to extend the method may be in a preference learning framework, where instances of an unobserved target can be ordered from the orders provided for the instances of the observed targets. In this context, the model that provides the order changes from one target to another, but, unlike in classification or ordinal regression, it does not require predictions for new unobserved classes, since in preference learning there are not class values.

CRediT authorship contribution statement

Miriam Fdez-Díaz: Conceptualization, Methodology, Software, Validation. **Elena Montañés:** Investigation, Supervision, Writing – review & editing. **José Ramón Quevedo:** Conceptualization, Methodology, Investigation, Software, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

A link for the code and data is available in the manuscript.

Acknowledgments

This research has been partially supported by the Spanish Ministry of Science and Innovation through the grant PID2019-110742RB-I00.

References

[1] D. Kumar, et al., Evolving Differential evolution method with random forest for prediction of Air Pollution, Procedia Comput. Sci. 132 (2018) 824–833.
[2] S. Masmoudi, H. Elghazel, D. Taieb, O. Yazar, A. Kallel, A machine-learning framework for predicting multiple air pollutants' concentrations via multi-target regression and feature selection, Sci. Total Environ. 715 (2020) 136991.
[3] M.R. Delavar, A. Gholami, G.R. Shiran, Y. Rashidi, G.R. Nakhaeizadeh, K. Fedra, S. Hatefi Afshar, A novel method for improving air pollution prediction based on machine learning approaches: a case study applied to the capital city of Tehran, ISPRS Int. J. Geo-Inf. 8 (2) (2019) 99.
[4] J. Murillo-Escobar, J. Sepulveda-Suescun, M. Correa, D. Orrego-Metaute, Forecasting concentrations of air pollutants using support vector regression improved with particle swarm optimization: Case study in Aburrá Valley, Colombia, Urban Clim. 29 (2019) 100473.
[5] Y. Guo, G. Ding, J. Han, Y. Gao, Synthesizing samples for zero-shot learning, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI '17, AAAI Press, 2017, pp. 1774–1780.
[6] E. Kodirov, T. Xiang, Z. Fu, S. Gong, Unsupervised domain adaptation for zero-shot learning, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 2452–2460.
[7] C.H. Lampert, H. Nickisch, S. Harmeling, Attribute-based classification for zero-shot visual object categorization, IEEE Trans. Pattern Anal. Mach. Intell. 36 (3) (2013) 453–465.
[8] B. Romera-Paredes, P. Torr, An embarrassingly simple approach to zero-shot learning, in: International Conference on Machine Learning, 2015, pp. 2152–2161.
[9] W. Waegeman, K. Dembczyński, E. Hüllermeier, Multi-target prediction: a unifying view on problems and methods, Data Min. Knowl. Discov. 33 (2) (2019) 293–324.
[10] W. Wang, V.W. Zheng, H. Yu, C. Miao, A survey of zero-shot learning: Settings, methods, and applications, ACM Trans. Intell. Syst. Technol. 10 (2) (2019).
[11] D. Kang, D. Dhar, A. Chan, Incorporating side information by adaptive convolution, in: Advances in Neural Information Processing Systems, 2017, pp. 3867–3877.
[12] M. Palatucci, D. Pomerleau, G. Hinton, T.M. Mitchell, Zero-shot learning with semantic output codes, in: Proceedings of the 22nd International Conference on Neural Information Processing Systems, NIPS '09, Curran Associates Inc., Red Hook, NY, USA, 2009, pp. 1410–1418.
[13] S. Hirschmeier, D. Schoder, Combining word embeddings with taxonomy information for multi-label document classification, in: Proceedings of the ACM Symposium on Document Engineering 2019, DocEng '19, Association for Computing Machinery, New York, NY, USA, 2019.
[14] A.K. Menon, K.-P. Chitrapura, S. Garg, D. Agarwal, N. Kota, Response prediction using collaborative filtering with hierarchies and side-information, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '11, Association for Computing Machinery, New York, NY, USA, 2011, pp. 141–149.
[15] L. Jacob, J.-P. Vert, Protein-ligand interaction prediction: An improved chemogenomics approach, Bioinform. (Oxford, England) 24 (2008) 2149–2156.
[16] T. Liu, Z. Wang, J. Tang, S. Yang, G.Y. Huang, Z. Liu, Recommender systems with heterogeneous side information, in: The World Wide Web Conference, WWW '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 3027–3033.
[17] M. Fdez-Díaz, J.R. Quevedo, E. Montañés, Target inductive methods for zero-shot regression, Inform. Sci. 599 (2022) 44–63.
[18] R. Qiao, L. Liu, C. Shen, A. Van Den Hengel, Less is more: Zero-shot learning from online textual documents with noise suppression, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2249–2257.
[19] V.F. Farias, A.A. Li, Learning preferences with side information, Manage. Sci. 65 (7) (2019) 3131–3149.

[20] Q. Wang, K. Chen, Alternative semantic representations for zero-shot human action recognition, in: M. Ceci, J. Hollmén, L. Todorovski, C. Vens, S. Dzeroski (Eds.), Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2017, Skopje, Macedonia, September 18-22, 2017, Proceedings, Part I, in: Lecture Notes in Computer Science, 10534, Springer, 2017, pp. 87–102.

[21] M. Rezaei, M. Shahidi, Zero-shot learning and its applications from autonomous vehicles to covid-19 diagnosis: A review, Intell.-Based Med. (2020) 100005.

[22] J. Gu, Y. Wang, K. Cho, V.O. Li, Improved zero-shot neural machine translation via ignoring spurious correlations, 2019, arXiv preprint arXiv:1906.01181.

[23] M. Johnson, M. Schuster, Q.V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado, et al., Google's multilingual neural machine translation system: Enabling zero-shot translation, Trans. Assoc. Comput. Linguist. 5 (2017) 339–351.

[24] O. Levy, M. Seo, E. Choi, L. Zettlemoyer, Zero-shot relation extraction via reading comprehension, 2017, arXiv preprint arXiv:1706.04115.

[25] Y. Ma, E. Cambria, S. Gao, Label embedding for zero-shot fine-grained named entity typing, in: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, 2016, pp. 171–180.

[26] Y. Shigeto, I. Suzuki, K. Hara, M. Shimbo, Y. Matsumoto, Ridge regression, hubness, and zero-shot learning, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2015, pp. 135–151.

[27] M. Elhoseiny, J. Liu, H. Cheng, H. Sawhney, A. Elgammal, Zero-shot event detection by multimodal distributional semantic embedding of videos, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 30, 2016.

[28] C. Gan, C. Sun, R. Nevatia, Deck: Discovering event composition knowledge from web images for zero-shot event detection and recounting in videos, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 31, 2017.

[29] J. Gao, T. Zhang, C. Xu, I know the relationships: Zero-shot action recognition via two-stream graph convolutional networks and knowledge graphs, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 8303–8311.

[30] P. Mettes, C.G. Snoek, Spatial-aware object embeddings for zero-shot localization and classification of actions, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 4443–4452.

[31] B. Xu, Y. Fu, Y.-G. Jiang, B. Li, L. Sigal, Video emotion recognition with transferred deep feature encodings, in: Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval, 2016, pp. 15–22.

[32] X. Xu, T. Hospedales, S. Gong, Transductive zero-shot action recognition by word-vector embedding, Int. J. Comput. Vis. 123 (3) (2017) 309–333.

[33] P. Robyns, E. Marin, W. Lamotte, P. Quax, D. Singelée, B. Preneel, Physical-layer fingerprinting of LoRa devices using supervised and zero-shot learning, in: Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks, 2017, pp. 58–63.

[34] S. Cappallo, T. Mensink, C.G. Snoek, Image2emoji: Zero-shot emoji prediction for visual media, in: Proceedings of the 23rd ACM International Conference on Multimedia, 2015, pp. 1311–1314.

[35] C. Zhan, D. She, S. Zhao, M.-M. Cheng, J. Yang, Zero-shot emotion recognition via affective structural embedding, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1151–1160.

[36] W. Wang, C. Miao, S. Hao, Zero-shot human activity recognition via nonlinear compatibility based method, in: Proceedings of the International Conference on Web Intelligence, 2017, pp. 322–330.

[37] C.A. Caceres, M.J. Roos, K.M. Rupp, G. Milsap, N.E. Crone, M.E. Wolmetz, C.R. Ratto, Feature selection methods for zero-shot learning of neural activity, Front. Neuroinformatics 11 (2017) 41.

[38] A. Kuznetsova, S.J. Hwang, B. Rosenhahn, L. Sigal, Exploiting view-specific appearance similarities across classes for zero-shot pose prediction: A metric learning approach, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 30, 2016.

[39] C. Liao, L. Su, W. Zhang, Q. Huang, Semantic manifold alignment in visual feature space for zero-shot learning, in: 2018 IEEE International Conference on Multimedia and Expo (ICME), IEEE, 2018, pp. 1–6.

[40] C. Luo, Z. Li, K. Huang, J. Feng, M. Wang, Zero-shot learning via attribute regression and class prototype rectification, IEEE Trans. Image Process. 27 (2) (2017) 637–648.

[41] S. Naha, Y. Wang, Object figure-ground segmentation using zero-shot learning, in: 2016 23rd International Conference on Pattern Recognition (ICPR), IEEE, 2016, pp. 2842–2847.

[42] X. Shu, G.-J. Qi, J. Tang, J. Wang, Weakly-shared deep transfer networks for heterogeneous-domain knowledge propagation, in: Proceedings of the 23rd ACM International Conference on Multimedia, MM '15, Association for Computing Machinery, New York, NY, USA, 2015, pp. 35–44.

[43] H. Song, B. Deng, M. Pound, E. Özcan, I. Triguero, A fusion spatial attention approach for few-shot learning, Inf. Fusion 81 (2022) 187–202.

[44] J. Tang, X. Shu, Z. Li, G.-J. Qi, J. Wang, Generalized deep transfer networks for knowledge propagation in heterogeneous domains, ACM Trans. Multimedia Comput. Commun. Appl. 12 (4s) (2016).

[45] Z. Wang, R. Hu, C. Liang, Y. Yu, J. Jiang, M. Ye, J. Chen, Q. Leng, Zero-shot person re-identification via cross-view consistency, IEEE Trans. Multimed. 18 (2) (2015) 260–272.

[46] Y. Yang, T. Hospedales, Zero-shot domain adaptation via kernel regression on the grassmannian, in: H. Drira, S. Kurtek, P. Turaga (Eds.), Proceedings of the 1st International Workshop on DIFFerential Geometry in Computer Vision for Analysis of Shapes, Images and Trajectories (DIFF-CV 2015), BMVA Press, 2015, pp. 1.1–1.12.

[47] Z. Zhang, V. Saligrama, Zero-shot learning via joint latent similarity embedding, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 6034–6042.

[48] J. Reis, G. Gonçalves, Hyper-process model: A zero-shot learning algorithm for regression problems based on shape analysis, 2018, ArXiv abs/1810.10330.

[49] C. Morariu, O. Morariu, S. Răileanu, T. Borangiu, Machine learning for predictive scheduling and resource allocation in large scale manufacturing systems, Comput. Ind. 120 (2020) 103244.

[50] E. Belcore, S. Angeli, E. Colucci, M.A. Musci, I. Aicardi, Precision agriculture workflow, from data collection to data management using FOSS tools: An application in northern Italy vineyard, ISPRS Int. J. Geo-Inf. 10 (4) (2021) 236.

[51] X. Hu, M. Dai, J. Sun, E. Sunderland, The utility of machine learning models for predicting chemical contaminants in drinking water: Promise, challenges, and opportunities, in: Current Environmental Health Reports, Vol. 10, 2023.

[52] W. Zhang, S. Seto, D.K. Jha, CAZSL: Zero-shot regression for pushing models by generalizing through context, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020, pp. 7131–7138.

[53] A. Mollaysa, A. Kalousis, E. Bruno, M. Diephuis, Learning to augment with feature side-information, in: W.S. Lee, T. Suzuki (Eds.), Proceedings of the Eleventh Asian Conference on Machine Learning, in: Proceedings of Machine Learning Research, 101, Nagoya, Japan, 2019, pp. 173–187.

[54] S. Rahman, S. Khan, N. Barnes, Transductive learning for zero-shot object detection, in: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 6081–6090.

[55] H. Li, F. Wang, J. Liu, J. Huang, T. Zhang, S. Yang, Micro-knowledge embedding for zero-shot classification, Comput. Electr. Eng. 101 (2022) 108068.

[56] F. Pourpanah, M. Abdar, Y. Luo, X. Zhou, R. Wang, C.P. Lim, X.-Z. Wang, Q.M.J. Wu, A review of generalized zero-shot learning methods, IEEE Trans. Pattern Anal. Mach. Intell. (2022) 1–20.

[57] S. Chen, Z. Hong, Y. Liu, G.-S. Xie, B. Sun, H. Li, Q. Peng, K. Lu, X. You, TransZero: Attribute-guided transformer for zero-shot learning, Proc. AAAI Conf. Artif. Intell. 36 (1) (2022) 330–338.

[58] S. Chen, Z. Hong, G. Xie, W. Yang, Q. Peng, K. Wang, J. Zhao, X. You, MSDN: Mutually semantic distillation network for zero-shot learning, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, la, USA, June 18-24, 2022, IEEE, 2022, pp. 7602–7611.

[59] D. Huynh, E. Elhamifar, Fine-grained generalized zero-shot learning via dense attribute-based attention, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 4482–4492.

[60] S. Chen, W. Wang, B. Xia, Q. Peng, X. You, F. Zheng, L. Shao, FREE: Feature refinement for generalized zero-shot learning, in: 2021 IEEE/CVF International Conference on Computer Vision (ICCV), 2021, pp. 122–131.

[61] S. Narayan, A. Gupta, F.S. Khan, C.G.M. Snoek, L. Shao, Latent embedding feedback and discriminative features for zero-shot classification, in: A. Vedaldi, H. Bischof, T. Brox, J.-M. Frahm (Eds.), Computer Vision – ECCV 2020, Springer International Publishing, Cham, 2020, pp. 479–495.

[62] Y. Xian, S. Sharma, B. Schiele, Z. Akata, F-VAEGAN-D2: A feature generating framework for any-shot learning, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 10267–10276.

[63] S. Chen, G.-S. Xie, Q. Peng, Y. Liu, B. Sun, H. Li, X. You, L. Shao, HSVA: Hierarchical semantic-visual adaptation for zero-shot learning, in: 35th Conference on Neural Information Processing Systems (NeurIPS), 2021.

[64] E. Schonfeld, S. Ebrahimi, S. Sinha, T. Darrell, Z. Akata, Generalized zero- and few-shot learning via aligned variational autoencoders, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, Los Alamitos, CA, USA, 2019, pp. 8239–8247.

[65] B. Kang, Z. Liu, X. Wang, F. Yu, J. Feng, T. Darrell, Few-shot object detection via feature reweighting, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 8420–8429.

[66] Z. Peng, Z. Li, J. Zhang, Y. Li, G. Qi, J. Tang, Few-shot image recognition with knowledge transfer, in: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 441–449.

[67] G. Koch, R. Zemel, R. Salakhutdinov, Siamese neural networks for one-shot image recognition, in: ICML Deep Learning Workshop, Vol. 2, Lille, 2015.

[68] Li Fei-Fei, R. Fergus, P. Perona, One-shot learning of object categories, IEEE Trans. Pattern Anal. Mach. Intell. 28 (4) (2006) 594–611.

[69] A. Farahani, B. Pourshojae, K. Rasheed, H.R. Arabnia, A concise review of transfer learning, in: 2020 International Conference on Computational Science and Computational Intelligence (CSCI), IEEE, 2020, pp. 344–351.

[70] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, Q. He, A comprehensive survey on transfer learning, Proc. IEEE 109 (1) (2020) 43–76.

[71] S.J. Pan, Q. Yang, A survey on transfer learning, IEEE Trans. Knowl. Data Eng. 22 (10) (2010) 1345–1359.

[72] G. Campagna, A. Foryciarz, M. Moradshahi, M.S. Lam, Zero-shot transfer learning with synthesized data for multi-domain dialogue state tracking, 2020, arXiv preprint arXiv:2005.00891.

[73] R. Socher, M. Ganjoo, H. Bastani, O. Bastani, C. Manning, A. Ng, Zero-shot learning through cross-modal transfer, Adv. Neural Inf. Process. Syst. (2013).
[74] G. Yang, J. Xu, Zero-shot transfer learning based on visual and textual resemblance, in: International Conference on Neural Information Processing, Springer, 2019, pp. 353–362.
[75] J. Shawe-Taylor, N. Cristianini, et al., Kernel Methods for Pattern Analysis, Cambridge University Press, 2004.
[76] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.

Miriam Fdez-Díaz received her degree in Computer Engineering in Information Technologies from the University of Oviedo, Spain, in 2016 and her masters in Artificial Intelligent from AEPIA (Spanish Association for Artificial Intelligence) and UIMP (Menéndez Pelayo International University), Spain, in 2017. She is a model engineer in ArcelorMittal Company and a Ph.D. in computer science (intelligent systems) from the University of Oviedo.

Elena Montañés received her M.Sc. degree in Mathematics in 1998 and her Ph.D. in Computer Science from the University of Oviedo, Spain, in 2003. She is currently a Senior Lecturer and a member of the Artificial Intelligence Center, Gijón. Her research interests are focused on several machine learning problems. She has authored many papers in peer-reviewed journals and conferences, including IJCAI, ECML, Machine Learning, Information Systems, Information Science and Pattern Recognition.

José Ramón Quevedo received his M.Sc. and Ph.D. degrees in Computer Science from the University of Oviedo, Spain, in 1997 and 2000, respectively. He is currently a Senior Lecturer and a member of the Artificial Intelligence Center, Gijón. His current research is focused on applying machine learning methods to quantification and to domains where the data distribution varies. He has published many papers in peer-reviewed journals placed at top of the Journal Citation Report. Also, he has authored papers in conferences of recognized prestige.