Multi-task additive models with shared transfer functions based on dictionary learning

Alhussein Fawzi ALHUSSEIN.FAWZI@EPFL.CH
Signal Processing Laboratory (LTS4), EPFL

Mathieu Sinn MATHSINN@IE.IBM.COM
IBM Research, Ireland

Pascal Frossard PASCAL.FROSSARD@EPFL.CH
Signal Processing Laboratory (LTS4), EPFL

Abstract

Additive models are a widely popular class of regression models which represent the relation between covariates and response variables as the sum of low-dimensional *transfer functions*. Besides flexibility and accuracy, a key selling point is their *interpretability* as the transfer functions provide visual means for inspecting the models and identifying domain-specific relations between inputs and outputs. In this paper, we introduce a novel multi-task learning approach which provides a corpus of accurate and interpretable additive models for a large number (e.g., thousands) of related forecasting tasks. Our key idea is to *share transfer functions across models* in order to reduce the complexity and ease the exploration of the corpus. We establish a connection with sparse dictionary learning and propose an efficient fitting algorithm which alternates between sparse coding and transfer function updates. The former step is solved via an extension of Orthogonal Matching Pursuit, and the latter step is solved using a traditional dictionary update step. Experiments on electricity data demonstrate that our approach compares favorably to baseline methods while yielding an interpretable corpus of models, revealing structure among the individual tasks and being more robust when training data are scarce.

Proceedings of the 31^{st} *International Conference on Machine Learning*, Lille, France, 2015. JMLR: W&CP volume 37. Copyright 2015 by the author(s).

1. Introduction

Additive models are a widely popular class of nonparametric regression models which have been extensively studied theoretically and successfully applied to a wide range of practical problems (Hastie et al., 2009; Wood, 2006; Hastie & Tibshirani, 1990). The key ingredient of additive models are *transfer functions* that explain the effect of covariates on the response variable in an additive manner. Besides being flexible (e.g., allowing for the modeling of nonlinear effects for both continuous and categorical covariates) and yielding good predictive performance, an important selling point of additive models is their interpretability. In particular, the transfer functions provide intuitive visual means for domain experts to understand the models and explore the relationship between inputs and outputs of the system under study.

In many real-world data modeling settings, one faces the problem of forecasting a large number (e.g., several thousands) of related tasks. Learning additive models independently for each task has several disadvantages: firstly, the number of models would be too large for a domain expert to visually inspect all the transfer functions, hence - in essence - the corpus of models loses its interpretability from a human point of view; secondly, independently learning the models ignores *structure* and *commonality* among the tasks; thirdly, when training data is scarce - relative to the number of tasks - learning the models independently is inefficient and prone to overfitting the data.

To overcome these challenges, we introduce a novel *multi-task learning* framework for additive models in this paper. Intuitively, the key idea is to *share transfer functions across tasks* that exhibit commonality in their relationships between input and output variables. More specifically, each individual task is modeled as a weighted sum of transfer functions chosen from a candidate set which is common to all tasks, and the cardinality of which is small relative to the total number of tasks. In the context of multi-task learning (Caruana, 1997), many approaches have been considered previously. In (Evgeniou et al., 2005), the authors impose the linear weight vectors of different tasks to be close to each other. The work in (Evgeniou & Pontil, 2007) constrains the weight vectors to live in a low-dimensional subspace. Still in the context of linear models, the authors of (Jacob et al., 2009) assume that the tasks are clustered into groups, and that tasks within a group have similar weight vectors. In (Liu et al., 2009), the authors consider additive models where the selected covariates are enforced to be the same across tasks, but the transfer functions can be different. While this results in an improved predictive performance for problems with a large number of covariates, the number of transfer functions may still be too large for inspection by human experts; moreover, this approach only leverages commonality with respect to *which* covariates affect the dependent variable, but not *how* they affect it.

Our algorithm for solving the multi-task additive model learning problem uses an intrinsic connection with *sparse dictionary learning* (Aharon et al., 2006; Tosic & Frossard, 2011; Kreutz-Delgado et al., 2003). More specifically, we reformulate the fitting problem as a special form of dictionary learning with additional constraints; leveraging recent advances in the field, we propose a fitting approach that alternates between updates of the transfer functions and the scaling weights. We introduce a novel algorithm for updating the coefficients that scale the transfer functions, called Block Constrainted Orthogonal Matching Pursuit (BC-OMP), which extends conventional Orthogonal Matching Pursuit (Pati et al., 1993; Mallat & Zhang, 1993).

In the experimental part of our paper, we apply the proposed algorithm to real-world electricity demand data. We show that, on 4,066 smart meter time series from Ireland, our approach yields predictive performance similar to baseline methods while only using a small number of candidate functions; interestingly, the discovered commonality of tasks corresponds to classes of residential and different types of enterprise customers. When using only a small fraction of the training data, our approach yields more robust results than independent learning and hence inherits the benefits of traditional multi-task learning.

The paper is structured as follow: Sec. 2 introduces notation and provides a review of additive models. In Sec. 3 we formulate the multi-task additive model learning problem and establish the connection with sparse dictionary learning. The algorithm for solving the multi-task problem is explained in Sec. 4. In Sec. 5 we describe our experiments on real-world electricity demand data; conclusions and an outlook on future research are given in Sec. 6.

Finally, we refer the reader to our technical report in (Fawzi et al., 2015) for theoretical guarantees on the algorithm, as well as additional experimental results that we omitted here due to space constraints.

2. Preliminaries

2.1. Notations

We use **boldface** notation for vectors and matrices. Moreover, we use $[n]$ to refer to the set $\{1, \dots, n\}$. Given a vector $\mathbf{z}$, we denote by $\|\mathbf{z}\|_0$ the ℓ_0 "norm", that counts the number of nonzero elements in $\mathbf{z}$. Also, we denote by $\otimes$ the Kronecker product operation. If $\mathbf{Z} \in \mathbb{R}^{n_1 \times n_2}$ is a matrix, $\text{vec}(\mathbf{Z}) \in \mathbb{R}^{n_1 n_2}$ denotes the vectorization of $\mathbf{Z}$, obtained by stacking the columns of $\mathbf{Z}$, and $\mathbf{Z}^\dagger$ denotes the Moore-Penrose pseudo-inverse. Moreover, we use the notation $\mathbf{z} \in \mathbf{Z}$ to denote that $\mathbf{z}$ is one of the columns of $\mathbf{Z}$. Finally, $\mathbf{Z} \geq \mathbf{0}$ denotes the entry-wise non-negativity constraint.

2.2. Additive models review

We first briefly review additive models. Let $\{x_{ij}, i \in [n], j \in [p]\}$ and $\{y_i, i \in [n]\}$ denote respectively the observed covariates and response variable. Here, n is the number of observations and p the number of covariates. Additive models have the form:

$$y_i = \mu + \sum_{j=1}^{p} f_j(x_{ij}) + \epsilon_i,$$

where μ is the intercept and ϵ_i is assumed to be a white noise process. The *transfer functions* f_j represent the effect of a covariate on the response variable. To ensure unique identification of the f_j's, we assume that transfer functions are centered: $\sum_{i=1}^{n} f_j(x_{ij}) = 0$ for all $j \in [p]$. Nonlinear transfer functions of continuous covariates are commonly modeled as smoothing splines (Wood, 2006; Hastie et al., 2009), i.e.,

$$f_j(z) = \sum_{t=1}^{T_j} \beta_{jt} \phi_{jt}(z), \tag{1}$$

where β_{jt} denotes the spline coefficients, ϕ_{jt} the B-spline basis functions, and T_j the number of basis splines. Using this representation, estimating the transfer functions therefore amounts to the estimation of the spline coefficients β_{jt} and the intercept μ. We consider the following fitting problem with centering constraints:

$$\min_{\mu, \{\beta_{jt}\}} \sum_{i=1}^{n} \left(y_i - \mu - \sum_{j=1}^{p} \sum_{t=1}^{T_j} \beta_{jt} \phi_{jt}(x_{ij}) \right)^2$$

$$\text{subject to } \sum_{i=1}^{n} \sum_{t=1}^{T_j} \beta_{jt} \phi_{jt}(x_{ij}) = 0 \text{ for all } j \in [p].$$

One can convert the above problem to an unconstrained optimization problem by centering the response and basis functions. Specifically, let $\bar{\phi}_{jt} = \frac{1}{n}\sum_{i=1}^{n}\phi_{jt}(x_{ij})$, $\mathbf{s}_j(z) = [\phi_{j1}(z) - \bar{\phi}_{j1}, \ldots, \phi_{jT_j}(z) - \bar{\phi}_{jT_j}]^T$ and

$$\mathbf{S} = \begin{bmatrix} \mathbf{s}_1(x_{11})^T & \ldots & \mathbf{s}_p(x_{1p})^T \\ & \vdots & \\ \mathbf{s}_1(x_{n1})^T & \ldots & \mathbf{s}_p(x_{np})^T \end{bmatrix}. \tag{2}$$

We define the vectorized spline coefficients $\boldsymbol{\beta} = \begin{bmatrix}\boldsymbol{\beta}_1^T & \ldots & \boldsymbol{\beta}_p^T\end{bmatrix}^T$ with $\boldsymbol{\beta}_j = \begin{bmatrix}\beta_{j1} & \ldots & \beta_{jT_j}\end{bmatrix}^T$. The above constrained fitting problem is then equivalent to the following unconstrained least squares problem:

$$\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{S}\boldsymbol{\beta}\|_2^2,$$

where $\mathbf{y}$ denotes the centered response variables $\mathbf{y} = [y_1 - \bar{y}, \ldots, y_n - \bar{y}]^T$, with $\bar{y} = 1/n\sum_{i=1}^{n} y_i$. In order to avoid overfitting, a quadratic penalizer is commonly added, leading to the problem:

$$\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{S}\boldsymbol{\beta}\|_2^2 + \boldsymbol{\beta}^T\boldsymbol{\Sigma}\boldsymbol{\beta},$$

with a regularization matrix $\boldsymbol{\Sigma}$. The penalized minimization problem has the closed form solution:

$$\hat{\boldsymbol{\beta}} = (\mathbf{S}^T\mathbf{S} + \boldsymbol{\Sigma})^{-1}\mathbf{S}^T\mathbf{y},$$

provided that $\mathbf{S}^T\mathbf{S} + \boldsymbol{\Sigma}$ is non-singular.

3. Multi-task additive model with shared transfer functions

We now introduce our new multi-task additive model with shared transfer functions. We assume a N-task regression problem where $\{x_{ij}^{(m)}, i \in [n], j \in [p], m \in [N]\}$ are the covariates, and $\{y_i^{(m)}, i \in [n], m \in [N]\}$ denotes the response variables, where the superscript (m) is the task index. We further assume without loss of generality that the response variables have zero mean. Our multi-task model is given as follows:

$$y_i^{(m)} = \sum_{j=1}^{p}\sum_{l=1}^{L_j} \lambda_{jl}^{(m)} f_{jl}\left(x_{ij}^{(m)}\right) + \epsilon_i^{(m)} \tag{3}$$
$$\text{with } \|\boldsymbol{\lambda}_j^{(m)}\|_0 \le 1 \text{ and } \boldsymbol{\lambda}_j^{(m)} \ge \mathbf{0} \text{ for all } j \in [p], m \in [N].$$

Note that, in our new model, the response variables are *weighted* linear combinations of p transfer functions, each of which is selected from the set $\mathcal{F}_j \triangleq \{f_{jl}, l \in [L_j]\}$ which contains L_j candidate transfer functions that model the effects of the covariate j. The ℓ_0 norm constraint on the weights $\boldsymbol{\lambda}_j^{(m)}$ prevents two transfer functions from the set $\mathcal{F}_j$ to be active for the same task. Hence, only one transfer function captures the effect of a covariate in a response variable. This constraint is crucial, as it disallows the creation of "new" transfer functions from the candidate ones by linearly combining them. While the transfer functions f_{jl} are *common* to all the tasks, the non-negative weights $\lambda_{jl}^{(m)}$ are *task-specific* and permit to scale the transfer functions specifically for each task. This offers extra flexibility as a wide range of tasks can be modeled using the model in Eq. (3) while keeping the number of (standardized) candidate transfer functions small. As we will see in Sec. 5, the non-negativity constraint in Eq. (3) facilitates the interpretation of the activation of the same transfer functions across different tasks as commonality; without this constraint, the same transfer functions could represent exactly opposite effects, e.g., higher temperatures leading to higher electricity demand for one task, and leading to lower demand for another one.

Similarly to what is done with single-task additive models (Sec. 2.2), we model transfer functions using smoothing splines. Specifically, we write:

$$f_{jl}(z) = \mathbf{s}_j(z)^T\boldsymbol{\beta}_{jl}, \tag{4}$$

where $\mathbf{s}_j$ and $\boldsymbol{\beta}_{jl}$ denote the centered spline basis functions and coefficients, respectively. Using this representation, we rewrite the model in Eq. (3) in the following vector form:

$$\forall m \in [N], \quad \mathbf{y}^{(m)} = \sum_{j=1}^{p} \mathbf{S}_j^{(m)}\mathbf{B}_j\boldsymbol{\lambda}_j^{(m)} + \boldsymbol{\epsilon}^{(m)},$$

where $\mathbf{S}_j^{(m)} = \left[\mathbf{s}_j\left(x_{1j}^{(m)}\right) \ \ldots \ \mathbf{s}_j\left(x_{nj}^{(m)}\right)\right]^T$, $\mathbf{B}_j = \begin{bmatrix}\boldsymbol{\beta}_{j1} & \ldots & \boldsymbol{\beta}_{jL_j}\end{bmatrix}$, and $\boldsymbol{\epsilon}^{(m)}$ is a Gaussian iid random vector with zero mean. The model fitting then consists in finding admissible $\{\mathbf{B}_j\}_{j=1}^{p}$ and $\{\boldsymbol{\lambda}_j^{(m)}\}_{j\in[p],m\in[N]}$ that minimize the sum of squared residuals, while avoiding overfitting. We therefore write the problem as follows:

$$\text{(P):} \min_{\mathbf{B}_j, \boldsymbol{\lambda}_j^{(m)}} \sum_{m=1}^{N}\left\|\mathbf{y}^{(m)} - \sum_{j=1}^{p}\mathbf{S}_j^{(m)}\mathbf{B}_j\boldsymbol{\lambda}_j^{(m)}\right\|_2^2 + \Omega(\{\mathbf{B}_j\}_{j=1}^{p}),$$
$$\text{subject to } \|\boldsymbol{\lambda}_j^{(m)}\|_0 \le 1 \text{ and } \boldsymbol{\lambda}_j^{(m)} \ge \mathbf{0} \text{ for all } j, m,$$

where Ω is a regularization term that prevents model overfitting. Note that, unlike traditional additive model learning, the above problem has two types of unknowns, that is, weights and transfer functions. In this paper, we use the following regularization function

$$\Omega(\{\mathbf{B}_j\}_{j=1}^{p}) = \mathbf{b}^T\boldsymbol{\Sigma}\mathbf{b}, \tag{5}$$

with the regularization matrix $\boldsymbol{\Sigma} = \nu\mathbf{I}$ and $\nu > 0$, and $\mathbf{b}$ is the vector formed by concatenating $\text{vec}(\mathbf{B}_j), 1 \le j \le$

p. Note that this regularizer penalizes large coefficients of smoothing splines with the strength of this effect tuned by ν.

The fitting problem (P) is inherently related to *sparse dictionary learning* (Tosic & Frossard, 2011) where the goal is to find the dictionary $\mathbf{D}$ and sparse codes $\mathbf{C}$ that minimize

$$\|\mathbf{Y} - \mathbf{DC}\|_F^2 \text{ subject to } \|\mathbf{c}\|_0 \le p \text{ for all } \mathbf{c} \in \mathbf{C},$$

with $\mathbf{Y} = [\mathbf{y}^{(1)} \dots \mathbf{y}^{(N)}] \in \mathbb{R}^{n \times N}$, and p is the desired level of sparsity. To simplify the exposition of the analogy, let us consider the multi-response scenario where covariates are equal across tasks ($x_{ij}^{(m)} = x_{ij}$ for all m). In this case, we have $\mathbf{S}_j^{(m)} = \mathbf{S}_j$ for all $m \in [N]$. We define the subdictionaries (or *blocks*) $\mathbf{D}_j \triangleq \mathbf{S}_j \mathbf{B}_j$ and the global dictionary $\mathbf{D} \triangleq \begin{bmatrix}\mathbf{D}_1 & \dots & \mathbf{D}_p\end{bmatrix}$. The problem (P) can be rewritten as follows:

$$\|\mathbf{Y} - \mathbf{D}\mathbf{\Lambda}\|_F^2 + \Omega(\{\mathbf{B}_j\}_{j=1}^p)$$
$$\text{subject to } \|\boldsymbol{\lambda}_j^{(m)}\|_0 \le 1 \text{ for all } j, m \text{ and } \mathbf{\Lambda} \ge \mathbf{0},$$

with

$$\mathbf{\Lambda} = \begin{bmatrix} \boldsymbol{\lambda}_1^{(1)} & \dots & \boldsymbol{\lambda}_1^{(N)} \\ \vdots & \vdots & \vdots \\ \boldsymbol{\lambda}_p^{(1)} & \dots & \boldsymbol{\lambda}_p^{(N)}. \end{bmatrix}$$

Hence, the difference between sparse dictionary learning and problem (P) essentially lies in the underlying *sparsity constraints*: while in the former one the only constraint is that sparse codes have no more than p nonzero entries, in the latter they are further constrained *to have at most one nonzero entry for each subdictionary*. Based on this analogy, we introduce in the next section a novel algorithm for efficiently approximating the solution of problem (P).

4. Learning algorithm

The problem of dictionary learning has proved challenging. In fact, even if the dictionary is known, it can be NP-hard to represent a vector as a linear combination of the columns in the dictionary (Davis et al., 1997). Problem (P) inherits the difficulty of dictionary learning, and we therefore propose an approximate algorithm that solves successively for the weights $\{\boldsymbol{\lambda}_j^{(m)}\}$ and spline coefficients $\{\mathbf{B}_j\}$. The proposed algorithm can be seen as an extension of the popular MOD algorithm (Engan et al., 1999), which alternates between sparse coding and dictionary updates.

4.1. Weights update

We assume that the spline coefficients matrices $\{\mathbf{B}_j\}$ are given, and we define $\mathbf{D}_j^{(m)} = \mathbf{S}_j^{(m)} \mathbf{B}_j \in \mathbb{R}^{n \times L_j}$. We define the columns of each subdictionary $\mathbf{D}_j^{(m)} = \begin{bmatrix}\mathbf{d}_{j,1}^{(m)} & \dots & \mathbf{d}_{j,L_j}^{(m)}\end{bmatrix}$ to be the *atoms* of $\mathbf{D}_j^{(m)}$. Hence, an atom $\mathbf{d}_{j,l}^{(m)}$ is obtained by applying the transfer function f_{jl} to all the observations of the jth covariate: $\mathbf{d}_{j,l}^{(m)} = \begin{bmatrix}f_{jl}(x_{1j}^{(m)}) & \dots & f_{jl}(x_{nj}^{(m)})\end{bmatrix}^T$. The weight estimation problem is given by

$$\min_{\{\boldsymbol{\lambda}_j^{(m)}\}_{j,m}} \sum_{m=1}^{N} \left\| \mathbf{y}^{(m)} - \sum_{j=1}^{p} \mathbf{D}_j^{(m)} \boldsymbol{\lambda}_j^{(m)} \right\|_2^2$$
$$\text{subject to } \|\boldsymbol{\lambda}_j^{(m)}\|_0 \le 1 \text{ and } \boldsymbol{\lambda}_j^{(m)} \ge \mathbf{0} \text{ for all } j, m.$$

The above problem can be seen as computing the best non-negative p-sparse approximations of the signals $\mathbf{y}^{(m)}$ in the dictionary $\mathbf{D}^{(m)} = [\mathbf{D}_1^{(m)} | \dots | \mathbf{D}_p^{(m)}]$, provided that no two active dictionary atoms belong to the same subdictionary. We first note that this problem is separable and therefore can be solved independently for each task. Next, we simplify the problem and drop the non-negativity constraints on $\boldsymbol{\lambda}_j^{(m)}$. Following the approach used in (Ekanadham et al., 2011; Fawzi & Frossard, 2013), non-negative coefficients can then be obtained in a post-processing step by including the negative of each atom in the dictionary[1], as we have:

$$\sum_{j=1}^{p} \mathbf{D}_j^{(m)} \boldsymbol{\lambda}_j^{(m)} = \sum_{j=1}^{p} \begin{bmatrix}\mathbf{D}_j^{(m)} & -\mathbf{D}_j^{(m)}\end{bmatrix} \begin{bmatrix}\max(0, \boldsymbol{\lambda}_j^{(m)}) \\ \max(0, -\boldsymbol{\lambda}_j^{(m)})\end{bmatrix}.$$

For a single task, our weight estimation problem is written:

$$\min_{\{\boldsymbol{\lambda}_j\}_{j=1}^p} \left\| \mathbf{y} - \sum_{j=1}^{p} \mathbf{D}_j \boldsymbol{\lambda}_j \right\|_2^2$$
$$\text{subject to } \|\boldsymbol{\lambda}_j\|_0 \le 1 \text{ for all } j \in [p].$$

To solve this problem, we propose the iterative algorithm Block Constrained Orthogonal Matching Pursuit (BC-OMP). It is an extension of the popular Orthogonal Matching Pursuit algorithm (Pati et al., 1993; Mallat & Zhang, 1993) which is an efficient greedy method for solving sparse coding problems. At each iteration of the algorithm, we select the dictionary atom which has the strongest correlation with the residual, provided it belongs to an available subdictionary whose index is listed in $\mathcal{A}_{j-1}$.

[1]This post-processing step results in doubling the number of candidate transfer functions. That is, for covariate j, we have $2L_j$ candidate transfer functions, as we consider the positive and negative versions of every (original) transfer function. Note also that this post-processing approach is an *approximation* and is therefore not guaranteed to give the exact solution to the original (constrained) problem, with $2L_j$ transfer functions for each covariate j.

Algorithm 1 BC-OMP

```
Input: Subdictionaries $\mathbf{D}_1, \dots, \mathbf{D}_p$, signal $\mathbf{y}$.
Output: Weight vectors $\boldsymbol{\lambda}_1, \dots, \boldsymbol{\lambda}_p$.

  Initialization:
  Available covariates: $\mathcal{A}_0 \leftarrow \{1, \dots, p\}$, residual: $\mathbf{r}_0 \leftarrow \mathbf{y}$, selected atoms: $\mathbf{U}_0 \leftarrow \emptyset$, weight vectors: $\boldsymbol{\lambda}_j \leftarrow \mathbf{0}$ for all $j \in [p]$.
  for all $j = 1, \dots, p$ do
    Selection step:
      $\{k_j, l_j\} \leftarrow \underset{k \in \mathcal{A}_{j-1}}{\text{argmax}} \underset{l \in \{1,\dots,L_k\}}{\text{argmax}} \frac{|\langle \mathbf{r}_{j-1}, \mathbf{d}_{k,l} \rangle|}{\|\mathbf{d}_{k,l}\|_2}.$
    Update step:
      $\mathcal{A}_j \leftarrow \mathcal{A}_{j-1} \backslash \{k_j\}, \mathbf{U}_j \leftarrow \begin{bmatrix} \mathbf{U}_{j-1} & \{\mathbf{d}_{k_j,l_j}\} \end{bmatrix}$
      $\mathbf{c}_j \leftarrow \mathbf{U}_j^{\dagger} \mathbf{y}, \mathbf{r}_j \leftarrow \mathbf{y} - \mathbf{U}_j \mathbf{c}_j.$
  end for
  for all $j = 1, \dots, p$ do
    Set $\boldsymbol{\lambda}_{k_j}[l_j] \leftarrow \mathbf{c}_p[j]$.
  end for
```

The residual is then updated using an orthogonal projection onto the selected atoms. The availability set $\mathcal{A}_j$ is in turn updated to prevent selecting two atoms from the same subdictionary. Details of our approach are presented in Algorithm 1. We refer the reader to our technical report (Fawzi et al., 2015) for a detailed analysis of the recovery conditions of BC-OMP.

4.2. Spline coefficients update

We now solve the problem of learning the spline coefficients $\mathbf{B}_j$ given the fixed weights $\boldsymbol{\lambda}_j^{(m)}$. We note that:

$$\begin{aligned}\sum_{j=1}^{p} \mathbf{S}_j^{(m)} \mathbf{B}_j \boldsymbol{\lambda}_j^{(m)} &= \sum_{j=1}^{p} ((\boldsymbol{\lambda}_j^{(m)})^T \otimes \mathbf{S}_j^{(m)}) \text{vec}(\mathbf{B}_j) \\ &= \begin{bmatrix} (\boldsymbol{\lambda}_1^{(m)})^T \otimes \mathbf{S}_1^{(m)} & \dots & (\boldsymbol{\lambda}_p^{(m)})^T \otimes \mathbf{S}_p^{(m)} \end{bmatrix} \begin{bmatrix} \text{vec}(\mathbf{B}_1) \\ \vdots \\ \text{vec}(\mathbf{B}_p) \end{bmatrix} \\ &\triangleq \mathbf{Z}^{(m)} \mathbf{b}.\end{aligned}$$

Thus, the objective function becomes:

$$\sum_{m=1}^{N} \|\mathbf{y}^{(m)} - \mathbf{Z}^{(m)} \mathbf{b}\|_2^2 + \mathbf{b}^T \boldsymbol{\Sigma} \mathbf{b} = \|\text{vec}(\mathbf{Y}) - \mathbf{Z}\mathbf{b}\|_2^2 + \mathbf{b}^T \boldsymbol{\Sigma} \mathbf{b},$$

with $\text{vec}(\mathbf{Y}) = \begin{bmatrix} \mathbf{y}^{(1)} \\ \vdots \\ \mathbf{y}^{(N)} \end{bmatrix}$ and $\mathbf{Z} = \begin{bmatrix} \mathbf{Z}^{(1)} \\ \vdots \\ \mathbf{Z}^{(N)} \end{bmatrix}$. The minimum of the above least-squares program with respect to $\mathbf{b}$ is given by:

$$\hat{\mathbf{b}} = (\mathbf{Z}^{\mathbf{T}} \mathbf{Z} + \boldsymbol{\Sigma})^{-1} \mathbf{Z}^{\mathbf{T}} \text{vec}(\mathbf{Y}). \tag{6}$$

Given the spline basis coefficients $\mathbf{B}_j$, the transfer functions can then be obtained by multiplying the obtained coefficients with the spline basis vectors (Eq. (4)).

4.3. Complete learning algorithm

The complete algorithm is shown in Algorithm 2. Using a random initialization of the spline coefficients, we iterate through the weights update and spline coefficients update steps, until a termination criterion is met. In this paper, we terminate the algorithm after a fixed number of iterations. In a final step, the matrices $\mathbf{B}_j$ and $\boldsymbol{\lambda}_j^{(m)}$ are modified to ensure the non-negativity of the weights, as discussed in Section 4.1.

Algorithm 2 Multi-task additive model algorithm

```
Input: Covariates $\{x_{ij}^{(m)}\}_{i,j,m}$, response variables $\{y_i^{(m)}\}_{i,m}$, parameters $L_1, \dots, L_p$ and $\nu$.
Output: Spline coefficients $\{\mathbf{B}_j\}_j$, scaling weights $\{\boldsymbol{\lambda}_j^{(m)}\}_{j,m}$.

  Initialize $\mathbf{B}_1, \dots, \mathbf{B}_p$ with random entries from $\mathcal{N}(0,1)$.
  while not converged do
    Weights update: Use BC-OMP for each response variable $\mathbf{y}^{(m)}$ to estimate $\{\boldsymbol{\lambda}_j^{(m)}\}_{j \in [p], m \in [N]}$.
    Spline coefficients update: Use Eq. (6) to update the spline coefficients.
  end while
  Ensure the non-negativity of the weights:
      $\mathbf{B}_j \leftarrow \begin{bmatrix} \mathbf{B}_j & -\mathbf{B}_j \end{bmatrix},$
      $\boldsymbol{\lambda}_j^{(m)} \leftarrow \begin{bmatrix} \max(0, \boldsymbol{\lambda}_j^{(m)}) \\ \max(0, -\boldsymbol{\lambda}_j^{(m)}) \end{bmatrix},$
  for all $j, m$.
```

5. Experimental results

5.1. Experimental setup

We compare the proposed multi-task learning approach to the following baseline regression methods:

1. **Linear Regression (LR):** A linear regressor is learned independently using ϵ-SVR with a linear kernel for each task. The penalty parameter C is set using a cross-validation procedure for each task.

2. **Support Vector Regression with Radial Basis Function kernel (SVR-RBF):** We learn an ϵ-SVR-

RBF regressor independently for each task where the penalty parameter C and kernel bandwidth σ are determined using cross-validation.

3. **Independent Additive Models (IAM):** An additive model is fitted for each task independently using the `mgcv` package in `R` (Wood, 2006).

4. **K-Means and Additive Model (KAM):** This is a two-step approach, where in the first step we use the K-means algorithm to group the set of tasks into different clusters. In the second step, one additive model is learned independently for each cluster centroid. The prediction of a signal is given by the prediction for the centroid of the cluster it belongs to.

5.2. Modeling of smart meter data

We use data from a smart metering trial of the Irish Commission for Energy Regulation (CER) (cer, 2011). The data set contains half-hourly electricity consumption data from July 14, 2009 to December 31, 2010 for approx. $5,000$ residential (RES) and small-to-medium enterprise (SME) customers. It comes with survey information about different demographic and socio-economic indicator, e.g., number of people living in the household, type of appliances, and business opening times. In our experiment, we only use customers that do not have any missing consumption data, leaving us with a total of $N = 4,066$ meters out of which $3,639$ are residential and 427 SME customers. Since half-hourly smart meter data is very volatile due to the stochastic nature of electricity consumption at the individual household level, we aggregate each signal over 6 time points to obtain one measurement every 3 hours. We split the data into 12 months of training and 6 months of test data. We consider a simple additive model with "Hour Of Day", "Time Of Year" and "Day of Week" as covariates.

Table 1 shows the average RMSE on the training and test data over all $N = 4,066$ meters. Here, we use $L = L_1 = \cdots = L_p = 5$ for the number of candidate transfer functions per covariate in our approach, and the number of clusters in the KAM method. In terms of predictive performance, our proposed approach clearly outperforms LR and KAM; it performs only slight worse than IAM albeit using only $L = 5$ different transfer functions per covariate while IAM learns independently one additive model per signal. Note that the methods based on additive models are competitive with SVR-RBF, while the latter approach is computationally expensive at training and test time, which makes it only moderately suitable for large-scale problems, besides leading to models that are unfortunately difficult to interpret.

In an attempt to quantitatively compare the interpretability of the different methods, Table 1 shows the *model complexity* of the different methods learned on the CER data, defined as the number of scalar variables needed to store the model[2]. For easier numerical comparison we divide the numbers by Np, i.e., the number of tasks times the number of covariates. As it can be seen, the SVR-RBF is the most complex model, since it depends on the number of observations n. IAM also has a high complexity as it fits one additive model per task. On the other hand, our proposed approach only needs pTL variables for the transfer functions, and $2Np$ values to store the weights $\{\boldsymbol{\lambda}_j^{(m)}\}$. In the experiment on the CER data, this results in a complexity of 2.01, where 0.01 results from the representation of the transfer functions, and 2 from storing the weight coefficients for each task. We conclude that our proposed approach finds a good trade-off between predictive performance and model complexity.

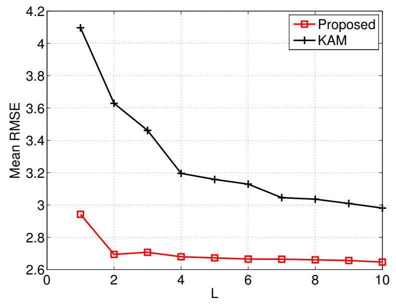


Figure 1. Average RMSE on the CER data set vs. number of candidate functions (per covariate) and clusters L, respectively, for the proposed and the KAM method.

Fig. 1 shows the predictive performance of the proposed algorithm and the KAM method for different values of L. One can see that, already for values of $L \geq 2$, our method reaches an accuracy that is close to the performance of independently learned additive models. The performance of KAM is consistently worse, regardless of the number of clusters.

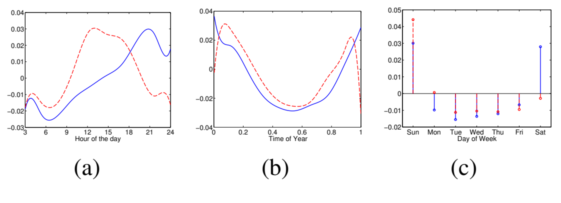


Figure 2. Transfer functions obtained with $L = 2$ (a): Hour of Day, (b): Time of Year (0: January, 1st, 1: December, 31st), (c): Day of Week.

Fig. 2 (a)-(c) display the transfer functions obtained by our

[2]For the SVR-RBF method, it is assumed for simplicity that the number of support vectors is equal to the number of training points n.

Table 1. Average RMSE over all $4,066$ tasks in the CER data set, and model complexity. For the proposed and KAM method, the results with $L = 5$ are shown. For easier comparison, the fourth column shows the *normalized* model complexity, i.e., the model complexity divided by Np where N is the number of tasks and p the number of covariates. T is the number of elements in the spline basis (equal for all covariates, for simplicity).

Method	RMSE		Model complexity	
	Training	Testing	Theoretical	Numerical example
Proposed	2.6	2.7	$p(TL + 2N)$	2.01
LR	3.1	3.1	Np	1
SVR-RBF	**2.2**	**2.5**	$Nn(p + 1)$	≥ 3800
IAM	2.3	2.6	pTN	12
KAM	3.1	3.2	pTL	**0.01**

method for $L = 2$. Let us consider the interpretability of the transfer functions learned using our method, and study correspondances with the customer survey information in the CER data set. Table 2 relates the activation of the "Hour of Day" transfer functions to the customer type (residential vs. SME). In this experiment we chose $L = 2$, resulting in 4 "final" transfer functions due to the non-negativity trick discussed in Sec. 4.1. One can see that, for residential customers, overwhelmingly the first transfer function is activated, while the majority of SME signals is modeled using the second one. Looking at the shape of the transfer functions, this intuitively makes sense: the consumption of residential customers typically peaks in the evening, while SMEs consume most electricity during the day. Similarly, Table 3 shows the correspondence between the activation of the "Day of Week" transfer function, and the SME business days (which is available from the CER survey information). Again, there is an intuitive and easy-to-interpret correspondance between the learned models and available ground truth information.

Finally, we evaluate the performance of our method in a setting where training data is scarce. For this purpose, we consider now a training set of n samples that are randomly selected from the CER data, and consider the remaining data for testing. Fig. 3 illustrates the testing RMSE of the proposed method (with $L = 2$) and the other competing methods with respect to n. It can be seen that when the training data is scarce, the proposed method outperforms IAM, and our method inherits the advantages of traditional multi-task learning by sharing information across tasks, and hence avoids overfitting. Note that for larger n, the gap between the two methods decreases, and IAM slightly outperforms our approach (with $L = 2$), as it provides a much more flexible model, which however suffers from lack of interpretability. Note finally that our approach consistently outperforms all other competing methods (LR, SVR-RBF, KAM) in Fig. 3.

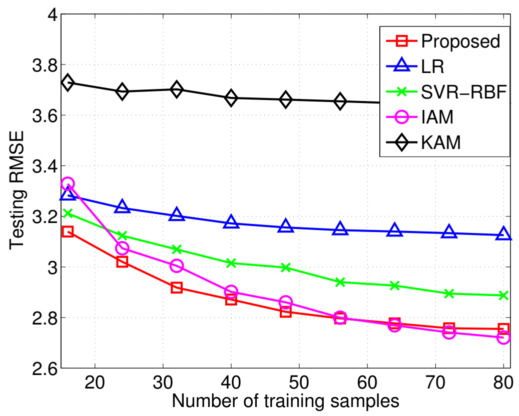


Figure 3. Testing RMSE versus number of training samples for the CER experiment.

6. Conclusion

In this paper, we introduced a novel multi-task learning framework for additive models with the key idea to *share transfer functions across the different tasks*. We established a connection between the proposed model and sparse dictionary learning and leveraged it to derive an efficient fitting algorithm. In experiments with real-world electricity demand data, we demonstrated that our proposed multi-task approach achieves competitive performance with baseline methods that learn models independently for each task, while providing models that are *more interpretable*, extracting *inherent structure* in the tasks (e.g., clustering of

Table 2. Percentage of the activation of "Hour of Day" transfer functions for residential and SME customers.

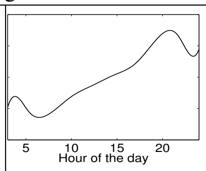
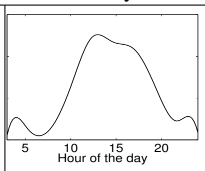
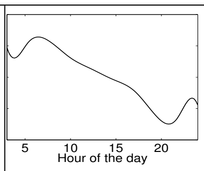
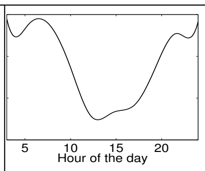
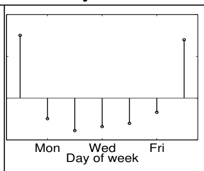
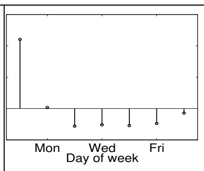
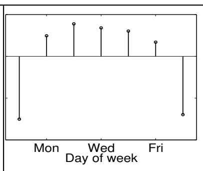
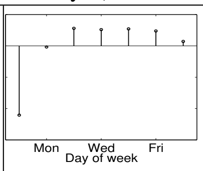
				
Residential	**89%**	9%	0%	2%
SME	19%	**68%**	8 %	5 %

Table 3. Percentage of the activation of "Day of Week" transfer functions for SMEs with different business days (see the left column).

				
Week days only	1%	2%	**93%**	4%
Week days + Saturday	0%	0%	37%	**63%**
All days	36%	5%	19%	**40%**

tasks corresponding to different customer types), and being *more robust* in settings where training data are scarce. In future work, we plan to improve the scalability of the method to apply it to domains that potentially involve millions of tasks.

References

Smart metering information paper 4: Results of electricity cost-benefit analysis, customer behavior trials and technology trials. The Commission for Energy Regulation (CER), 2011.

Aharon, M, Elad, M, and Bruckstein, A. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Trans. on Sig. Proc.*, 54(11):4311–4322, 2006.

Caruana, R. Multitask learning. *Machine Learning*, 28: 41–75, 1997.

Davis, G, Mallat, S, and Avellaneda, M. Adaptive greedy approximations. *Constructive approximation*, 13(1):57–98, 1997.

Ekanadham, C, Tranchina, D, and Simoncelli, E. Sparse decomposition of transformation-invariant signals with continuous basis pursuit. In *IEEE ICASSP*, pp. 4060–4063, 2011.

Engan, K, Aase, S, and H, Hakon. Method of optimal directions for frame design. In *IEEE ICASSP*, volume 5, pp. 2443–2446, 1999.

Evgeniou, A and Pontil, M. Multi-task feature learning. *NIPS*, 19:41–48, 2007.

Evgeniou, T, Micchelli, C, and Pontil, M. Learning multiple tasks with kernel methods. In *JMLR*, pp. 615–637, 2005.

Fawzi, A and Frossard, P. Classification of unions of subspaces with sparse representations. In *Asilomar Conference on Signals, Systems and Computers*, pp. 1368–1372, 2013.

Fawzi, A, Sinn, M, and Frossard, P. Multi-task additive models with shared transfer functions based on dictionary learning. `http://infoscience.epfl.ch/record/207540/files/double.pdf`, 2015.

Hastie, T and Tibshirani, R. *Generalized additive models*, volume 43. CRC Press, 1990.

Hastie, T, Tibshirani, R, and Friedman, J. *The elements of statistical learning*. Springer, 2009.

Jacob, L, Vert, J-P, and Bach, F. Clustered multi-task learning: A convex formulation. In *NIPS*, pp. 745–752, 2009.

Kreutz-Delgado, K, Murray, J, Rao, B, Engan, K, Lee, TW, and Sejnowski, T. Dictionary learning algorithms for sparse representation. *Neural computation*, 15(2):349–396, 2003.

Liu, H, Wasserman, L, and Lafferty, J. Nonparametric regression and classification with joint sparsity constraints. In *NIPS*, pp. 969–976, 2009.

Mallat, S and Zhang, Z. Matching pursuits with time-frequency dictionaries. *IEEE Trans. on Sig. Proc.*, 41: 3397–3415, Dec. 1993.

Pati, YC, Rezaiifar, R, and Krishnaprasad, PS. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Asilomar Conference on Signals, Systems and Computers*, 1993.

Tosic, I and Frossard, P. Dictionary learning. *IEEE Sig. Proc. Mag.*, 28(2):27–38, 2011.

Wood, S. *Generalized additive models: an introduction with R*. CRC press, 2006.