Output Feature Augmented Lasso

Conference Paper · December 2014

DOI: 10.1109/ICDM.2014.103

4 authors:

Changqing Zhang
Tianjin University
25 PUBLICATIONS **41** CITATIONS
SEE PROFILE

Yahong Han
Tianjin University
60 PUBLICATIONS **436** CITATIONS
SEE PROFILE

Xiaojie Guo
Tianjin University
25 PUBLICATIONS **164** CITATIONS
SEE PROFILE

Xiaochun Cao
Chinese Academy of Sciences
150 PUBLICATIONS **875** CITATIONS
SEE PROFILE

2014 IEEE International Conference on Data Mining

Output Feature Augmented Lasso

Changqing Zhang*, Yahong Han*†, Xiaojie Guo‡, and Xiaochun Cao*‡
*School of Computer Science and Technology, Tianjin University, Tianjin 300072
Email: {zhangchangqing,yahong}@tju.edu.cn
†Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin University, Tianjin 300072
‡State Key Laboratory of Information Security, Institute of Information Engineering
Chinese Academy of Sciences, Beijing 100093
Email: xj.max.guo@gmail.com; caoxiaochun@iie.ac.cn

***Abstract*—Lasso simultaneously conducts variable selection and supervised regression. In this paper, we extend Lasso to multiple output prediction, which belongs to the categories of structured learning. Though structured learning makes use of both input and output simultaneously, the joint feature mapping in current framework of structured learning is usually application-specific. As a result, ad hoc heuristics have to be employed to design different joint feature mapping functions for different applications, which results in the lackness of generalization ability for multiple output prediction. To address this limitation, in this paper, we propose to augment Lasso with output by decoupling the joint feature mapping function of traditional structured learning. The contribution of this paper is three-fold: 1) The augmented Lasso conducts regression and variable selection on both the input and output features, and thus the learned model could fit an output with both the selected input variables and the other correlated outputs. 2) To be more general, we set up nonlinear dependencies among output variables by generalized Lasso. 3) Moreover, the Augmented Lagrangian Method (ALM) with Alternating Direction Minimizing (ADM) strategy is used to find the optimal model parameters. The extensive experimental results demonstrate the effectiveness of the proposed method.**

Keywords*—Lasso; multiple output prediction; structured learning; alternating direction minimizing

I. Introduction

Sparse model [1] is very important in many high dimensional data analysis tasks. The well studied model, Lasso [2], is a popular strategy to simultaneously select features and build regression models. Lasso deals with the problem that one input is associated with a univariate output. However, learning with multiple dimensional output is also a general and important problem. For example, in econometrics [3], we consider predicting the stock prices of several companies based on previous values; in computational biology [4], we aim to predict the gene-expression levels of multiple genes based on a set of single nucleotide polymorphism (SNPs); and so on. All these applications suggest a common underlying problem: predicting multivariate output.

Generally, the outputs (output variables) themselves are often related to each other via some underlying structure. Therefore, constructing models independently for each output variable tends to fail to capture the correlations among different outputs. These correlations are usually critical for many applications. The early work called curds and whey method [5] uses shrinkage techniques in output space to reduce prediction error. The methods in [6], [7] model the correlations among the output space but without sparse assumption. The landmark selection method [8] is a sparse model, which assumes that there exists a small set of landmark variables in the high dimensional output space, and solves this problem in a two-step manner. Note that, most of the previous methods assume that the structure over the output is known as an a priori [4], [9], [10], [11], which limits the generalization ability of these approaches. In this paper, we model the correlations of outputs by decoupling the joint feature mapping under the structured learning framework. More importantly, our method does not need the explicit structure of the output variables, which makes it more general for multiple output prediction.

To address more complex correlations, some nonlinear sparse methods [12], [13], [14], [15] have been proposed. The instance-wise nonlinear Lasso proposed in [12] transforms the original instance by a nonlinear function to handle the nonlinearity. There are also some methods concentrating on the interaction effects [14], [16], unlike the traditional work only considering the main effects. In our work, we use Generalized Lasso [12] to explore the nonlinear dependencies among output variables. Moreover, several optimization techniques [17], [18] have been proposed for sparse learning models. The Augmented Lagrange Multiplier (ALM) based algorithms [19] are faster than the previous state-of-the-art APG (Accelerated Proximal Gradient) based methods [20], [21] and simpler to analyze or implement. Therefore, we make use of the augmented Lagrangian method with Alternating Direction Minimizing (ADM) strategy as our solver.

The key contributions of this paper are highlighted as follows. First, we propose a novel method which extends Lasso for multiple output prediction. Our method decouples the joint feature mapping in the framework of structured learning, resulting in the Output Feature Augmented Lasso (*OFA-Lasso*). Second, unlike the methods in [8], [22], we model the nonlinear dependencies instead of the linear ones among outputs via Generalized Lasso. Thus, the more general correlations can be explored. Finally, we propose to optimize our model via the effective ALM method. We have conducted extensive experiments on both synthetic and real-world datasets which demonstrate the effectiveness of the proposed method.

II. Approach

In this section, we first briefly introduce Lasso in subsection II-A. Then, we review how the nonlinear dependencies between the input and output are addressed in subsection II-B. Afterwards, the output feature augmented Lasso is proposed and the proof of the equivalence between Lasso and the linear output feature augmented Lasso is given in subsection II-C. In subsection II-D, we use the efficient L-BFGS [23] optimization

1550-4786/14 $31.00 © 2014 IEEE
DOI 10.1109/ICDM.2014.103

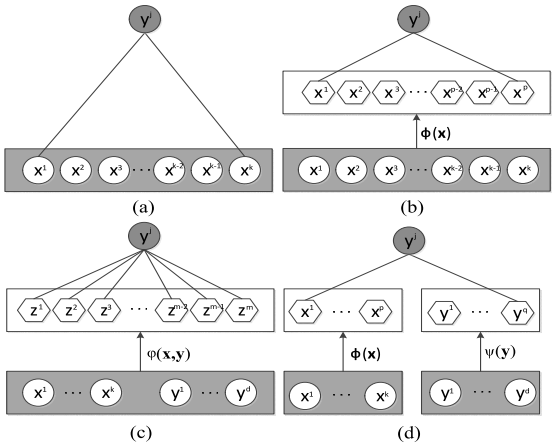


Fig. 1. Illustration of different multiple-output regression models. (a) Lasso. (b) Generalized Lasso. (c) Structured Regression. (d) OFA-Lasso. The directed edges denote the feature mapping from the original feature space (dark gray rectangle) to the mapped feature space (white rectangle). The undirected edges indicate the dependence between input and output variables. The gray node stands for the current output variable to be predicted.

approach to obtain prediction in the testing stage.

A. Preliminary

Generally, given a set of training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, the task of supervised learning is to estimate a function $y = f(\mathbf{x})$ by utilizing the training set. Considering the linear prediction mode $f(\mathbf{x}) = \mathbf{w}^\mathsf{T}\mathbf{x}$, when the data dimension is much larger than the number of data, say $k \gg n$, an unconstrained empirical risk minimization is inadequate because the solution is very likely to overfit the training data. Hence, regularization by imposing a penalty to the parameter $\mathbf{w}$ is employed to reduce the effect of overfitting. Lasso is a well-known sparse regression method which regularizes parameter $\mathbf{w}$ by sparse assumption, as shown in Fig. 1(a). The model parameter $\mathbf{w}$ of Lasso is obtained via the following formula,

$$\hat{\mathbf{w}}(\lambda) = \underset{\mathbf{w}}{\operatorname{argmin}} ||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \lambda||\mathbf{w}||_1 \ , \tag{1}$$

where $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}^\mathsf{T} \in \mathbb{R}^{n\times k}$ and $\mathbf{y} = \{y_1, ..., y_n\}^\mathsf{T} \in \mathbb{R}^n$ are the input and output of training data, respectively. The model parameter $\mathbf{w} \in \mathbb{R}^k$ is sparse due to the ℓ_1-norm, while nonzero elements in $\mathbf{w}$ correspond to the selected features.

B. Regression with Generalized Lasso

To extend Lasso to nonlinear cases, Generalized Lasso is proposed in [12], which transforms the original instance by a nonlinear function $\phi(\cdot) : \mathbb{R}^k \to \mathbb{R}^p$, where k and p are the dimensionalities of the original and the mapped feature space respectively, as shown in Fig. 1(b). As a result, the formulation of generalized linear model turns out to be

$$f(\mathbf{x}) = \boldsymbol{\beta}^\mathsf{T}\phi(\mathbf{x}) \ . \tag{2}$$

Thus, the Generalized Lasso optimization problem can be expressed as the minimization of the following formula,

$$\hat{\boldsymbol{\beta}}(\lambda) = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} ||\mathbf{y} - \mathbf{K}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_1 \ , \tag{3}$$

where $\mathbf{K}$ is a $n \times n$ kernel matrix with $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\mathsf{T}\phi(\mathbf{x}_j)$, $\boldsymbol{\beta} = (\beta_1, ..., \beta_n)^\mathsf{T}$ is the regression coefficient vector. Specifically, $k(\cdot)$ is a kernel function and β_j is the coefficient of the j^{th} basis of $\mathbf{K}$. The trade-off factor $\lambda > 0$ is used to balance the empirical error and the sparsity of model parameters. In this case, the kernels themselves are interpreted as entries of feature vectors within a generalized linear model,

$$\phi(\mathbf{x}) = [k(\mathbf{x}, \mathbf{x}_1), ..., k(\mathbf{x}, \mathbf{x}_n), 1]^\mathsf{T} \ . \tag{4}$$

Therefore, by introducing the kernel function into Lasso, more general correlations between the input and output can be explored. Furthermore, in our OFA-Lasso, the correlations among the output variables themselves are also explored in this manner.

C. Output Feature Augmented Lasso

Lasso and Generalized Lasso are promising techniques for univariate output regression. For the multiple output case, a set of training sample pairs $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ are given, where n is the size of training set and $\mathbf{y}_i \in \mathbb{R}^d$ is an output with the dimensionality of d. The task is to learn a map function from input $\mathbf{x}$ to output $\mathbf{y}$. A intuitive way to tackle such a problem is to process each output variable independently. In this case, one can learn d functions $\{f^j\}_{j=1}^d$ from training data,

$$\hat{\mathbf{w}}^j = \underset{\mathbf{w}^j}{\operatorname{argmin}} \sum_{i=1}^{n} \{L(y_i^j, \hat{f}^j(\mathbf{x}_i, \mathbf{w}^j, b^j)) + \lambda^j R(\mathbf{w}^j)\}$$
$$\text{s.t. } \hat{f}^j(\mathbf{x}) = (\mathbf{w}^j)^\mathsf{T}\mathbf{x} + b^j, \ j = 1, ..., d \ , \tag{5}$$

where $\mathbf{w}^j$ and $\mathbf{b}^j$ stand for the model parameter and bias of the j^{th} model, respectively. $L(\cdot)$ is the loss function and $R(\cdot)$ represents the regularization term. It is observed that from Eq. (5), each output is independently predicted without considering the dependencies among different outputs. However, outputs themselves often have certain correlations. Therefore, for the multiple output setting, taking into account both the input and output associatively is of great importance. Based on this observation, we introduce the structured learning [24], [25] fashion to model the input-output associative model which often performs in a search-based way [26],

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}}\{\mathbf{w}^\mathsf{T} \cdot \varphi(\mathbf{x}, \mathbf{y})\} \ , \tag{6}$$

where φ is the joint feature mapping. Generally, exactly solving the problem is intractable except in limited cases such as the dependency structure among outputs is explicitly known. Accordingly, the structured regression model (see Fig. 1(c)) can be induced as follow,

$$\hat{f}^j = (\mathbf{w}^j)^\mathsf{T} \cdot \varphi(\mathbf{x}, \mathbf{y}) + b^j \ , j = 1, ..., d. \tag{7}$$

Although the model takes advantage of the output correlations into account, it is difficult to decide the joint feature mapping φ, which is often application-specific [27], [28]. To address this difficulty, we substitute the joint feature mapping $\varphi(\mathbf{x}, \mathbf{y})$ with two independent mapping functions $\phi(\mathbf{x})$ and $\psi(\mathbf{y})$, then an input-output associative regression model (see Fig. 1(d)) is obtained,

$$\hat{f}^j = (\mathbf{u}^j)^\mathsf{T} \cdot \phi(\mathbf{x}) + (\mathbf{v}^j)^\mathsf{T} \cdot \psi(\mathbf{y}^{-j}) + b^j, \ j = 1, ..., d \ , \tag{8}$$

where $\mathbf{y}^{-j}$ is the output vector except the j^{th} dimension, which means that the current variable to be predicted depends on both the input variables and the other output variables. $\mathbf{u}^j$ and $\mathbf{v}^j$ are model parameters. For training stage, we can obtain the model parameters by minimizing the following regularized empirical risk,

$$\{\mathbf{U}, \mathbf{V}, \mathbf{b}\} = \underset{\mathbf{u}^j, \mathbf{v}^j, b^j}{\operatorname{argmin}} \sum_{j=1}^{d} \sum_{i=1}^{n} \{L(y_i^j, \hat{f}^j(\mathbf{x}_i, \mathbf{y}_i, \mathbf{u}^j, \mathbf{v}^j, b^j)) + R(\mathbf{u}^j, \mathbf{v}^j)\},$$

with

$$R(\mathbf{u}^j, \mathbf{v}^j) = \lambda^j \parallel \mathbf{u}^j \parallel_1 + \beta^j \parallel \mathbf{v}^j \parallel_1 , \, j = 1, ..., d \, , \quad (9)$$

We rewrite it in the following more compact form,

$$\underset{\mathbf{u}^j, \mathbf{v}^j}{\operatorname{argmin}} \sum_{j=1}^{d} \|\mathbf{y}^j - \mathbf{K}\mathbf{u}^j - \mathbf{G}^j\mathbf{v}^j\|_2^2 + \lambda^j\|\mathbf{u}^j\|_1 + \beta^j\|\mathbf{v}^j\|_1. \quad (10)$$

where similar to the kernel matrix $\mathbf{K}$ in Eq. (3), $\mathbf{G}^j$ is the kernel matrix with $\mathbf{G}^j_{p,q} = k(\mathbf{y}_p^{-j}, \mathbf{y}_q^{-j}) = \psi(\mathbf{y}_p^{-j})^T\psi(\mathbf{y}_q^{-j})$. Let $\mathbf{u}^j = (u_1^j, ..., u_n^j)^\mathsf{T}$ and $\mathbf{v}^j = (v_1^j, ..., v_n^j)^\mathsf{T}$ denote the regression coefficient vectors, with u_p^j and v_p^j being the coefficients of the p^{th} basis of $\mathbf{K}^j$ and $\mathbf{G}^j$, respectively. $\lambda^j > 0$ and $\beta^j > 0$ are trade-off factors. For simplicity, the trade-off factors λ^j and β^j corresponding to different output variables can be set as the same value. The model incorporates the Output variables as Augmented Features into Lasso, therefore we call it *OFA-Lasso* in this paper.

Note that, the training of the d regression models is decoupled, which makes the parameters of different models can be learned independently. Compared to Lasso, the proposed method learns the regression parameters using both the input and output features in a seamless model. Therefore, OFA-Lasso can explore the correlations among output variables which makes it more suitable for multiple output prediction. In addition, the kernel mapping technique is introduced to our model which makes our model more general and can be used to address the equivalence problem (see proposition II.1).

Proposition II.1. *Given the linear mapping function ϕ_l and ψ_l, the output augmented regression model, i.e., $y^j = (\mathbf{u}^j)^\mathsf{T} \cdot \phi_l(\mathbf{x}_i) + (\mathbf{v}^j)^\mathsf{T} \cdot \psi_l(\mathbf{y}_i^{-j}) + b^j$, is equal to the regression model $y^j = (\mathbf{u}^j)^\mathsf{T} \cdot \mathbf{x} + b^j$.*

Proof II.1. Denote $\boldsymbol{\vartheta}^j = (v_1^j, ..., v_{j-1}^j, -1, v_{j+1}^j, ..., v_d^j)^\mathsf{T}$ where $j = 1, ..., d$, the more compact representation of the output augmented regression model can be rewritten as follow,

$$(\boldsymbol{\vartheta}^j)^\mathsf{T} \cdot \mathbf{y} + (\mathbf{u}^j)^\mathsf{T} \cdot \mathbf{x} + b^j = 0 \, .$$

Using Gaussian elimination on the linear equation system, we can get the following equation:

$$\mathbf{D}\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{b} \, ,$$

where the matrix $\mathbf{D} \in \mathbb{R}^{d \times d}$ is a diagonal matrix after appropriate linear transformation. The matrix $\mathbf{C} \in \mathbb{R}^{d \times k}$ and vector $\mathbf{b}$ are the coefficient matrix of input $\mathbf{x}$ and bias vector, respectively. Obviously, each output variable y^j is a linear combination of input variables, without any dependency on the output variables. □

The above proposition clearly show the equivalence between Lasso and OFA-Lasso when using linear kernels and implies the necessity for setting up nonlinear dependency among output variables by Generalized Lasso.

D. Prediction

After obtaining the model parameters with the ALM (Augmented Lagrangian Method) which will be detailed in the next section, we utilize the L-BFGS [23] optimization method to obtain the prediction, which is a limited-memory quasi-Newton method for unconstrained optimization. Our objective function is

$$\begin{aligned} \hat{\mathbf{y}} &= \underset{\mathbf{y}}{\operatorname{argmin}} \sum_{j=1}^{d} L(y_i^j, f^j(\mathbf{x}_i, \mathbf{y}_i, \mathbf{u}^j, \mathbf{v}^j, b^j)) \\ &= \underset{\mathbf{y}}{\operatorname{argmin}} \sum_{j=1}^{d} (y^j - (\mathbf{u}^j)^\mathsf{T} \cdot \phi(\mathbf{x}) - (\mathbf{v}^j)^\mathsf{T} \cdot \psi(\mathbf{y}^{-j}))^2 \, . \end{aligned} \quad (11)$$

We initialize the L-BFGS optimizer with the prediction result given by independent Lasso regression. In our experiments, the initialization improves the accuracy and speeds up the prediction simultaneously, compared to random initialization. The step size is set by linear search satisfying the Wolfe conditions [29] which is used to ensure the fast convergence.

III. Optimization Method

To compute the model parameters, we now focus on solving the following objective functions by decoupling the formula (10) into d independent objective functions , as follow,

$$\underset{\mathbf{u}^j, \mathbf{v}^j}{\operatorname{argmin}} \|\mathbf{y}^j - \mathbf{K}\mathbf{u}^j - \mathbf{G}^j\mathbf{v}^j\|_2^2 + \lambda^j\|\mathbf{u}^j\|_1 + \beta^j\|\mathbf{v}^j\|_1, \quad j = 1, ..., d. \quad (12)$$

This is a least squares loss with ℓ_1-norm regularization optimization problem, more specially, an unconstrained formulation. Plenty of methods have been developed to address this problem. For the reasons of effectiveness and efficiency, we employ ALM as our solver. However, ALM is an efficient algorithm when the augmented Lagrangian function is easy to be minimized with respect to variables. Since there are two variables $\mathbf{u}^j$ and $\mathbf{v}^j$ in our formulation, the objective function is difficult to be minimized. Directly using ALM for our problem is time consuming.

To address above problem and inspired by the Alternating Direction Minimizing (ADM) strategy [30], we approximately solve it in the manner of minimizing the variables once at a time, which makes the ALM tractable for our objective function. The efficiency of our method is similar as that of the ALM. To adopt the ADM strategy to our problem, we need to make our objective function separable. We thus introduce two auxiliary variables $\mathbf{p}^j$ and $\mathbf{q}^j$ to replace $\mathbf{u}^j$ and $\mathbf{v}^j$ in the second and third terms of the objective function (12), respectively. Accordingly, $\mathbf{p}^j = \mathbf{u}^j$ and $\mathbf{q}^j = \mathbf{v}^j$ are two additional constraints. As a result, the formulation of the problem (12) turns out to be,

$$\begin{aligned} &\underset{\mathbf{u}^j, \mathbf{v}^j}{\operatorname{argmin}} \|\mathbf{y}^j - \mathbf{K}\mathbf{u}^j - \mathbf{G}^j\mathbf{v}^j\|_2^2 + \lambda^j\|\mathbf{p}^j\|_1 + \beta^j\|\mathbf{q}^j\|_1 \\ &\text{s. t.} \quad \mathbf{p}^j = \mathbf{u}^j, \quad \mathbf{q}^j = \mathbf{v}^j. \end{aligned} \quad (13)$$

Therefore, the corresponding augmented Lagrangian function is given as follow,

$$\begin{aligned}&\mathcal{L}_\mu(\mathbf{u}^j,\mathbf{v}^j,\mathbf{p}^j,\mathbf{q}^j,\mathbf{Z}^j,\mathbf{Q}^j)\\&=\|\mathbf{y}^j-\mathbf{K}\mathbf{u}^j-\mathbf{G}^j\mathbf{v}^j\|_2^2+\lambda^j\|\mathbf{p}^j\|_1+\beta^j\|\mathbf{q}^j\|_1\\&+\frac{\mu}{2}\|\mathbf{p}^j-\mathbf{u}^j\|_F^2+<\mathbf{Z}^j,\mathbf{p}^j-\mathbf{u}^j>\\&+\frac{\mu}{2}\|\mathbf{q}^j-\mathbf{v}^j\|_F^2+<\mathbf{Q}^j,\mathbf{q}^j-\mathbf{v}^j>,\end{aligned}\tag{14}$$

where $\mathbf{Z}^j$ and $\mathbf{Q}^j$ are Lagrange multipliers, μ is a positive penalty factor, $<\cdot,\cdot>$ denotes the matrix inner product, and $\|\cdot\|_F$ represents the Frobenius norm.

The ADM updates all the variables as follows:

$$\begin{aligned}\mathbf{u}_{k+1}^j&=\operatorname{argmin}\mathcal{L}_{\mu_k}(\mathbf{u}^j,\mathbf{v}_k^j,\mathbf{p}_k^j,\mathbf{q}_k^j,\mathbf{Z}_k^j,\mathbf{Q}_k^j)\\&=\operatorname{argmin}\|\mathbf{y}^j-\mathbf{K}\mathbf{u}^j-\mathbf{G}^j\mathbf{v}_k^j\|_2^2\\&+\frac{\mu_k}{2}\|\mathbf{p}_k^j-\mathbf{u}^j\|_F^2+<\mathbf{Z}_k^j,\mathbf{p}_k^j-\mathbf{u}^j>,\end{aligned}\tag{15}$$

$$\begin{aligned}\mathbf{v}_{k+1}^j&=\operatorname{argmin}\mathcal{L}_{\mu_k}(\mathbf{u}_{k+1}^j,\mathbf{v}^j,\mathbf{p}_k^j,\mathbf{q}_k^j,\mathbf{Z}_k^j,\mathbf{Q}_k^j)\\&=\operatorname{argmin}\|\mathbf{y}^j-\mathbf{K}\mathbf{u}_{k+1}^j-\mathbf{G}^j\mathbf{v}^j\|_2^2\\&+\frac{\mu_k}{2}\|\mathbf{q}_k^j-\mathbf{v}^j\|_F^2+<\mathbf{Q}_k^j,\mathbf{q}_k^j-\mathbf{v}^j>,\end{aligned}\tag{16}$$

$$\begin{aligned}\mathbf{p}_{k+1}^j&=\operatorname{argmin}\mathcal{L}_{\mu_k}(\mathbf{u}_{k+1}^j,\mathbf{v}_{k+1}^j,\mathbf{p}^j,\mathbf{q}_k^j,\mathbf{Z}_k^j,\mathbf{Q}_k^j)\\&=\operatorname{argmin}\lambda^j\|\mathbf{p}^j\|_1+\frac{\mu_k}{2}\|\mathbf{p}^j-\mathbf{u}_{k+1}^j\|_F^2\\&+<\mathbf{Z}_k^j,\mathbf{p}^j-\mathbf{u}_{k+1}^j>,\end{aligned}\tag{17}$$

$$\begin{aligned}\mathbf{q}_{k+1}^j&=\operatorname{argmin}\mathcal{L}_{\mu_k}(\mathbf{u}_{k+1}^j,\mathbf{v}_{k+1}^j,\mathbf{p}_{k+1}^j,\mathbf{q}^j,\mathbf{Z}_k^j,\mathbf{Q}_k^j)\\&=\operatorname{argmin}\beta^j\|\mathbf{q}^j\|_1+\frac{\mu_k}{2}\|\mathbf{q}^j-\mathbf{v}_{k+1}^j\|_F^2\\&+<\mathbf{Q}_k^j,\mathbf{q}^j-\mathbf{v}_{k+1}^j>,\end{aligned}\tag{18}$$

$$\begin{aligned}\mathbf{Z}_{k+1}^j&=\mathbf{Z}_k^j+\mu_k(\mathbf{p}_{k+1}^j-\mathbf{u}_{k+1}^j),\\\mathbf{Q}_{k+1}^j&=\mathbf{Q}_k^j+\mu_k(\mathbf{q}_{k+1}^j-\mathbf{v}_{k+1}^j),\\\mu_{k+1}&=\rho\mu_k,\end{aligned}\tag{19}$$

where $\rho>1$ is the amplification coefficient of penalty factor μ. Fortunately, for the subproblems (15)-(18), each has a simple closed form solution, and hence can be solved efficiently. The solutions are shown in Algorithm 1, respectively. $\mathbf{I}$ stands for the identity matrix with an appropriate size. $\mathcal{S}_\varepsilon[\cdot]$ is the shrinkage operator, the definition of which on scalars is: $\mathcal{S}_\varepsilon[x]\doteq\operatorname{sgn}(x)\max(|x|-\varepsilon,0)$. The extension of the shrinkage operator to vectors and matrices is applied in element-wise manner. The procedure of solving (12) has been summarized as Algorithm 1. The algorithm is stopped when $\|\mathbf{p}_{k+1}^j-\mathbf{u}_{k+1}^j\|_F^2\le\delta\max(\|\mathbf{p}_{k+1}^j\|_F^2,\|\mathbf{u}_{k+1}^j\|_F^2)$ and $\|\mathbf{q}_{k+1}^j-\mathbf{v}_{k+1}^j\|_F^2\le\delta\max(\|\mathbf{q}_{k+1}^j\|_F^2,\|\mathbf{v}_{k+1}^j\|_F^2)$ with $\delta=10^{-7}$ or the maximal number of iterations is reached.

Algorithm 1: The optimization algorithm of (12)

Input: $\mathbf{y}$; $\mathbf{K}$; $\mathbf{G}^j$; $\lambda>0$; $\beta>0$;
Initialization: $\mathbf{u}_0^j=\mathbf{v}_0^j=\mathbf{p}_0^j=\mathbf{q}_0^j=\mathbf{0}$; $\mathbf{Z}_0^j=\mathbf{Q}_0^j=\mathbf{0}$; $\mu_0>0$; $\rho>1$; $k=0$;
while *not converged* **do**

$\mathbf{u}_{k+1}^j=(2\mathbf{K}^T\mathbf{K}+\mu_k\mathbf{I})^{-1}(2(\mathbf{K})^T\mathbf{y}^j-2\mathbf{K}^T\mathbf{G}^j\mathbf{v}_k^j+\mu_k\mathbf{p}_k^j+\mathbf{Z}_k^j)$;
$\mathbf{v}_{k+1}^j=(2(\mathbf{G}^j)^T\mathbf{G}^j+\mu_k\mathbf{I})^{-1}(2(\mathbf{G}^j)^T\mathbf{y}^j-2(\mathbf{G}^j)^T\mathbf{K}\mathbf{u}_{k+1}^j+\mu_k\mathbf{q}_k^j+\mathbf{Q}_k^j)$;
$\mathbf{p}_{k+1}^j=\mathcal{S}_{\frac{\lambda}{\mu_k}}[\mathbf{u}_{k+1}^j-\frac{\mathbf{Z}_k^j}{\mu_k}]$;
$\mathbf{q}_{k+1}^j=\mathcal{S}_{\frac{\beta}{\mu_k}}[\mathbf{v}_{k+1}^j-\frac{\mathbf{Q}_k^j}{\mu_k}]$;
$\mathbf{Z}_{k+1}^j=\mathbf{Z}_k^j+\mu_k(\mathbf{p}_{k+1}^j-\mathbf{u}_{k+1}^j)$;
$\mathbf{Q}_{k+1}^j=\mathbf{Q}_k^j+\mu_k(\mathbf{q}_{k+1}^j-\mathbf{v}_{k+1}^j)$;
$\mu_{k+1}=\rho\mu_k$;
$k=k+1$;

end
Output: Optimal solution ($\mathbf{u}^{j*}$, $\mathbf{v}^{j*}$, $\mathbf{p}^{j*}$, $\mathbf{q}^{j*}$)

IV. Experimental Results

A. Experimental Settings

Datasets: Similar to some other works [8], [31], we conduct our experiments on synthetic datasets and real-world datasets to verify the effectiveness of our approach. We utilize the synthetic datasets to test the ability of our method in exploiting the nonlinear correlations among outputs, and utilize two real-world datasets to validate the better performance in practical applications. The synthetic data is generated according to different correlation functions explicitly, which can be used to validate the ability in exploring various correlations. The real-world datasets correspond to two different applications: digit reconstruction and stock-price prediction. For the USPS [32] and (S&P 500) data, training samples with different sizes and different output dimensionalities are chosen respectively to test the robustness of our method.

Compared Approaches: First, our main contribution is using output features to augment Lasso, thus we choose Lasso [2] as the baseline to validate the effectiveness of using outputs as augmented features. Second, for employing the kernel technique to deal with the generalized correlations in our approach, we choose GLasso (Generalized Lasso) [12] as the second comparison, which also utilizes the kernel technique. Finally, to test the effect in exploiting the output correlations, we compare our method with MtLasso (Multi-task Lasso) [22], which also extends Lasso to multiple output regression by exploiting the output correlations implicitly.

Features: To verify the robustness of our approach in terms of different features, we use two types of features, i.e., the original input features and the features mapped by kernel function. Specifically, according to the different inputs, our experiments are divided into *group-I* (original input) and *group-II* (mapped by kernel). Note that, the output is always mapped by kernel to address the equivalence problem.

Evaluation Metric: We use the Mean Absolute Error (MAE) to evaluate the performance of prediction,

$$\text{MAE}=\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d}|y_{ij}-\hat{y}_{ij}|\ ,$$

where n and d are the size of test dataset and dimensionality of output, respectively. $\hat{y}_{ij}$ and y_{ij} correspond to the prediction and the true value, respectively.

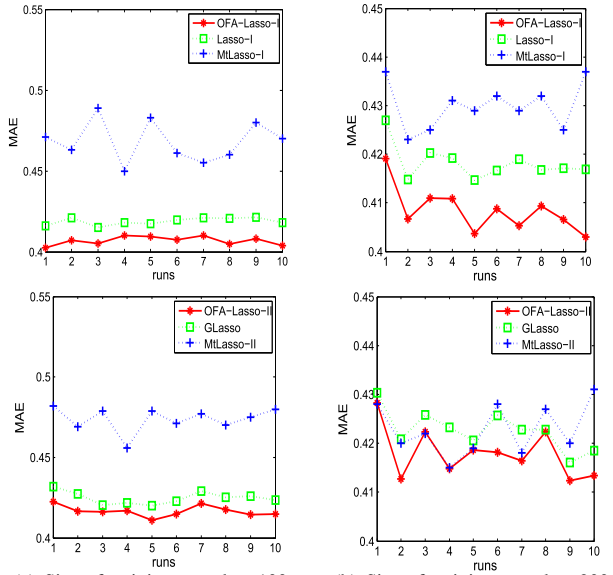
In this paper, the Gaussian kernel is used for both input

TABLE I. RESULTS OF *MAE*±*standard deviation* ON SIMULATION DATA. THE SUBINDEX $*_{1,2}$ IMPLIES THE DIFFERENT GROUPS OF FUNCTIONS.

Method	Lasso	MtLasso-I	OFA-Lasso-I	GLasso	MtLasso-II	OFA-Lasso-II
$sin(x)_1$	0.0401±0.0004	0.0351±0.0002	**0.0325**±0.0004	0.2588±0.0013	0.2593±0.0023	**0.2334**±0.0014
$(1/x)_1$	0.0649±0.0012	0.0644±0.0007	**0.0605**±0.0003	0.3049±0.0007	0.3546±0.0006	**0.2702**±0.0002
$exp(x)_1$	0.1211±0.0001	**0.1210**±0.0001	**0.1210**±0.0002	0.5132±0.0016	0.7995±0.0016	**0.4953**±0.0021
$sin(x)_2$	0.0696±0.0017	0.0649±0.0005	**0.0608**±0.0007	0.2834±0.0011	0.3082±0.0002	**0.2536**±0.0002
$(1/x)_2$	0.1396±0.0011	0.1349±0.0009	**0.1308**±0.0012	0.3609±0.0004	0.4220±0.0020	**0.3325**±0.0009
$exp(x)_2$	0.2458±0.0002	**0.2237**±0.0001	0.2433±0.0003	0.8237±0.0021	1.2623±0.0012	**0.8135**±0.0013

TABLE II. RESULTS OF *MAE*±*standard deviation* ON THE USPS DATA SET WITH DIFFERENT SIZE(n) OF TRAINING SET.

Method	Lasso	MtLasso-I	OFA-Lasso-I	GLasso	MtLasso-II	OFA-Lasso-II
$n = 100$	0.4189±0.0019	0.4664±0.0013	**0.4023**±0.0009	0.4273±0.0021	0.4736±0.0010	**0.4165**±0.0012
$n = 200$	0.4219±0.0012	0.4289±0.0008	**0.4017**±0.0005	0.4257±0.0020	0.4229±0.0027	**0.4216**±0.0019
$n = 500$	0.4084±0.0022	0.3982±0.0015	**0.3827**±0.0003	0.3658±0.0008	0.3671±0.0013	**0.3571**±0.0024
$n = 1000$	0.3815±0.0013	0.3825±0.0008	**0.3746**±0.0016	0.3443±0.0006	0.3486±0.0009	**0.3363**±0.0011



(a) Size of training samples: 100 (b) Size of training samples: 200

Fig. 2. Performance on the USPS. (a): Prediction error (MAE) with training size of 100 versus 10 runs; (b): Prediction error (MAE) with training size of 200 versus 10 runs.

$\mathbf{x}$ and output $\mathbf{y}$. For parameter selection of Gaussian kernel, we use the setting $\sigma = \text{median}(\{|\mathbf{x}_i - \mathbf{x}_j|\}_{i,j=1}^n)$, where the kernel $k(\mathbf{x}, \mathbf{x}') = \exp(\frac{-(\mathbf{x}-\mathbf{x}')^2}{2\sigma^2})$ is similar to the setting in [15]. To avoid choosing the parameters λ and β empirically, we use cross validation to select parameters and set the same value for the d models. We search the appropriate values from the candidate set $0.1 \times \{5^1, 5^2, 5^3, 5^4, 5^5\}$ for both λ and β.

B. Experiments on Synthetic Data

Firstly, we use the synthetic data to evaluate the performance of the our method. The dimensionalities of input and output are $k = 500$ and $d = 3$, respectively. The true sparsity pattern and the correlations of output variables are given. There are two groups of correlation functions corresponding to different numbers of correlated output variables. It is shown as follows:

$$y^{(j)} = \alpha \cdot (\mathbf{w}^j)^\mathsf{T}\mathbf{x} + f(y^{(k)}) \ ,$$
$$y^{(j)} = \alpha \cdot (\mathbf{w}^j)^\mathsf{T}\mathbf{x} + f(y^{(k)}) + f(y^{(l)}),$$

where α is used to tradeoff the effect between input and output, which is set as 1 in our experiments. $f(\cdot)$ stands for one of the following functions:

$$y^{(j)} = \sin(y^{(k)}),\ y^{(j)} = \exp(y^{(k)}),\ y^{(j)} = 1/(y^{(k)}).$$

Therefore, six different nonlinear functions (combinations with two groups of correlations and three types of specific functions) are used to model the different correlations in the output space. Accordingly, six groups of high dimensional data are generated using a standard linear model. Specifically, for each data associated with a specific correlation, we generate $n = 1000$ observations. Each dimension is independently sampled from a uniform distribution on the [0, 1] interval. First, a random positive matrix $\mathbf{X} \in \mathbb{R}^{n\times k}$ acting as the input matrix set is generated. Then, the output $\mathbf{Y} \in \mathbb{R}^{n\times d}$ corresponding to the input matrix $\mathbf{X}$ is computed according to different parameters $\mathbf{w}^j$. The parameters with unified sparsity of $s = 5$ nonzero coefficients are generated randomly from 0 to 1. We split the simulated data to the train (200)/test (800) sets.

We repeat the experiments 10 times to randomly generate synthetic data and report the average MAE. The performance of different methods on different synthetic datasets is shown in Table I. For both group-I and group-II, our approach has smaller prediction error on almost all datasets compared to Lasso and GLasso. Note that the performance of all the methods in the group-II is significantly worse than the group-I. The reason is that the dependencies between the input and output are linear in nature according to the correlation functions. Thus preserving the linear dependencies (as in group-I) is more reasonable for the synthetic data. However, the OFA-Lasso archives better result in both groups. It is interesting that our method outperforms the other methods significantly in group-II. The main reason is that when the effect of the input is weakened, exploiting the correlations of output will play a more important role. The results on synthetic data demonstrate that adopting the output features is reasonable when the dependencies in the output space indeed

TABLE III. RESULTS OF *MAE±standard deviation* ON THE S&P500 DATA SET WITH DIFFERENT OUTPUT DIMENSIONALITY (d).

Method	Lasso	MtLasso-I	OFA-Lasso-I	GLasso	MtLasso-II	OFA-Lasso-II
$d = 16$	1.7863±0.0005	**1.6987**±0.0009	1.7145±0.0005	3.1773±0.0019	8.5401±0.0023	**2.8287**±0.0016
$d = 32$	2.4041±0.0013	2.3417±0.0022	**2.3176**±0.0009	3.0155±0.0029	7.8591±0.0041	**2.9838**±0.0032
$d = 48$	2.0149±0.0009	2.5117±0.0012	**1.9880**±0.0003	3.9375±0.0005	9.4883±0.0004	**3.6347**±0.0008
$d = 64$	2.1704±0.0012	2.3244±0.0027	**1.9851**±0.0015	3.2856±0.0102	9.3996±0.0026	**2.7524**±0.0012

exist. Moreover, the performance shows the robustness of our method with *different types of correlations* among the output dimensions.

C. Experiments on Real-world Data

We also conduct experiments on two real-world datasets. For the USPS, we consider the reconstruction task of the handwriting digits. Given the outer 240 pixel values of a digit image, we aim to predict the 16 pixels that are in the center of the image [6], [33]. We use 2-fold cross validation to select the tradeoff parameters λ and β, taking 1000 samples from training set as the validation data. From the left training set, we randomly select 100, 200, 500, 1000 samples as our training data and evaluate the performance on the whole test set. For each size of training samples, we take 10 runs on the different training data randomly selected, and report the average MAE and standard deviation in Table II.

The performance of each run with training data size $n = 100$ and $n = 200$ can be found in Figure 2. Note that, with the increasing size of training samples, the performance of each method is improved. In all cases our method outperforms the Lasso/GLasso and MtLasso. Although the kernel mapping of input does not always work well in all cases, our method can improve the performance in the both cases (with or without kernel mapping). Furthermore, it is observed that when the size of training samples is 100, our method outperforms the others more significantly. This demonstrates that our method works relatively well under the condition of limited training data. The performance demonstrates the effectiveness of our method with *different sizes of training samples*.

For the S&P 500 dataset which consists of closing stock prices of 500 companies from August 21, 2009 to June 8, 2010 (200 entries in total, 150 for training and 50 for testing), we randomly select 200 companies at time $t-1$ as input and the first k (in the set $\{16, 32, 48, 64\}$) companies at time t as output for simplicity. We run 10 times and report the mean result in Table III. Our approach achieves the relatively good prediction compared to the other methods. OFA-Lasso outperforms the others significantly in group-II. This may be caused by that the mapped input features cannot fit the output as well as the original features, then the effect of the augmented outputs becomes more prominent. The performance demonstrates the effectiveness of our method with *different dimensionalities of output*.

We evaluate the performance of *OFA-Lasso* with various conditions and compare it to several most related sparse approaches. Our method performs reasonably well on both synthetic and real-world datasets. Specially, almost on every dataset, our approach is with the smallest test error. It implies that each output can be fitted more precisely with the other outputs as augmented feature. It is consistent with the motivation of this work in theory. Please note that, as the MAE (Mean Absolute Error) of all methods is relatively small, our method significantly outperforms other methods. Although the absolute improvement might be small, the percentage is rather high. For example, the improvement from 0.0401 to 0.0325 is about 20% (first row of Table I). To further demonstrate the significance of the performance improvement, we have done the Student's t-test of our results at the $(100*\alpha)\%$ significance level. In the experiment, the results on 22 out of 28 groups of t-test are all 1, which means our method being better than those of other methods is correct with the probability of 1-α = 0.9999.

V. CONCLUSION

We have proposed a novel sparse method for predicting the multiple, mutually dependent output variables. We call it Output Feature Augmented Lasso (*OFA-Lasso*). It models the correlations among the outputs by kernel functions to explore complex dependencies. Unlike mostly structured sparse methods which are application-dependent, our model does not require the output structure as an a priori. Instead, it learns the structure from the data directly. Thus, it is more generalized in application. Extensive experimental results demonstrate the effectiveness of the proposed method. Future work includes a detailed analysis to characterize the improvements of the proposed methods over competing sparse models and probing more efficient optimization methods for prediction. On the other hand, the results in this paper raise several interesting questions and following up directions. First, a detailed analysis is required to characterize the improvements of the proposed methods over competing sparse models. Second, the other optimization methods, such as adaptive-Lasso [34] and Foba [31] for sparse models should be analysis in detail for our model. Last but not least, inspired by the work in [35], [36], [37], extending our approach to multi-label task is very interesting and will be our future work.

ACKNOWLEDGMENT

This work was supported by National Natural Science Foundation of China (No. 61422213, 61332012), National Basic Research Program of China (2013CB329305), National High-tech R&D Program of China (2013AA01A601), and 100 Talents Programme of The Chinese Academy of Sciences. Dr. Yahong Han was supported by the NSFC (under Grant 61202166, 61472276) and Doctoral Fund of Ministry of Education of China (under Grant 20120032120042). Xiaojie Guo was supported by National Natural Science Foundation of China (No. 61402467) and Excellent Young Talent of the Institute Information Engineering, Chinese Academy of Sciences.

REFERENCES

[1] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[2] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[3] A. J. Rothman, E. Levina, and J. Zhu, "Sparse multivariate regression with covariance estimation," *Journal of Computational and Graphical Statistics*, vol. 19, no. 4, pp. 947–962, 2010.

[4] S. Kim, K.-A. Sohn, and E. P. Xing, "A multivariate regression approach to association analysis of a quantitative trait network," *Bioinformatics*, vol. 25, no. 12, pp. 204–212, 2009.

[5] L. Breiman and J. H. Friedman, "Predicting multivariate responses in multiple linear regression," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 59, no. 1, pp. 3–54, 1997.

[6] L. Bo and C. Sminchisescu, "Structured output-associative regression," in *CVPR*, 2009.

[7] M. A. Nicolaou, H. Gunes, and M. Pantic, "Output-associative rvm regression for dimensional and continuous emotion prediction," in *FG*, 2011.

[8] K. Balasubramanian and G. Lebanon, "The landmark selection method for multiple output prediction," in *ICML*, 2012.

[9] P. Zhao, G. Rocha, and B. Yu, "Grouped and hierarchical model selection through composite absolute penalties," *Department of Statistics, UC Berkeley, Tech. Rep*, vol. 703, 2006.

[10] X. Chen, S. Kim, Q. Lin, J. G. Carbonell, and E. P. Xing, "Graph-structured multi-task regression and an efficient optimization method for general fused lasso," *arXiv preprint arXiv:1005.3579*, 2010.

[11] S. Kim and E. P. Xing, "Tree-guided group lasso for multi-response regression with structured sparsity, with an application to eqtl mapping," *The Annals of Applied Statistics*, vol. 6, no. 3, pp. 1095–1117, 2012.

[12] V. Roth, "The generalized lasso," *IEEE Transactions on Neural Networks*, vol. 15, no. 1, pp. 16–28, 2004.

[13] P. Ravikumar, J. Lafferty, H. Liu, and L. Wasserman, "Sparse additive models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 71, no. 5, pp. 1009–1030, 2009.

[14] L. Huang, J. Jia, B. Yu, B.-G. Chun, P. Maniatis, and M. Naik, "Predicting execution time of computer programs using sparse polynomial regression," in *NIPS*, 2010.

[15] M. Yamada, W. Jitkrittum, L. Sigal, E. P. Xing, and M. Sugiyama, "High-dimensional feature selection by feature-wise kernelized lasso," *Neural Computation*, vol. 26, no. 1, pp. 185–207, 2014.

[16] J. Bien, J. Taylor, and R. Tibshirani, "A lasso for hierarchical interactions," *The Annals of Statistics*, vol. 41, no. 3, pp. 1111–1141, 2013.

[17] M. Schmidt, "Least squares optimization with l1-norm regularization," *Project Report, University of British Columbia*, 2005.

[18] S. K. Shevade and S. S. Keerthi, "A simple and efficient algorithm for gene selection using sparse logistic regression," *Bioinformatics*, vol. 19, no. 17, pp. 2246–2253, 2003.

[19] Z. Lin, M. Chen, and Y. Ma, "The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices," *preprint*, 2010.

[20] Z. Lin, A. Ganesh, J. Wright, L. Wu, M. Chen, and Y. Ma, "Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix," *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, vol. 61, 2009.

[21] J.-F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 1956–1982, 2010.

[22] G. Obozinski, B. Taskar, and M. I. Jordan, "Multi-task feature selection," *Statistics Department, UC Berkeley, Tech. Rep*, 2006.

[23] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization," *ACM Transactions on Mathematical Software (TOMS)*, vol. 23, no. 4, pp. 550–560, 1997.

[24] Y. Altun, M. Belkin, and D. A. Mcallester, "Maximum margin semi-supervised learning for structured variables," in *NIPS*, 2005.

[25] A. Kulesza and F. Pereira, "Structured learning with approximate inference," in *NIPS*, 2007.

[26] J. R. Doppa, A. Fern, and P. Tadepalli, "Hc-search: Learning heuristics and cost functions for structured prediction," in *AAAI*, 2013.

[27] M. B. Blaschko and C. H. Lampert, "Learning to localize objects with structured output regression," in *ECCV*, 2008.

[28] C.-N. J. Yu and T. Joachims, "Learning structural svms with latent variables," in *ICML*, 2009.

[29] J. Nocedal and S. Wright, *Numerical Optimization*. New York: Springer, 2006.

[30] J. Yang and Y. Zhang, "Alternating direction algorithms for ℓ_1-problems in compressive sensing," *SIAM journal on scientific computing*, vol. 33, no. 1, pp. 250–278, 2011.

[31] T. Zhang, "Adaptive forward-backward greedy algorithm for sparse learning with linear models," in *NIPS*, 2008.

[32] J. J. Hull, "A database for handwritten text recognition research," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, pp. 550–554, 1994.

[33] J. Weston, O. Chapelle, A. Elisseeff, B. Schölkopf, and V. Vapnik, "Kernel dependency estimation," in *NIPS*, 2002.

[34] H. Zou, "The adaptive lasso and its oracle properties," *Journal of the American statistical association*, vol. 101, no. 476, pp. 1418–1429, 2006.

[35] Y. Zhang and J. Schneider, "Maximum margin output coding," in *ICML*, 2012.

[36] ——, "Multi-label output codes using canonical correlation analysis," *Journal of Machine Learning Research*, vol. 15, pp. 873–882, 2011.

[37] D. Hsu, S. M. Kakade, J. Langford, and T. Zhang, "Multi-label prediction via compressed sensing," in *NIPS*, 2009.