Multi-task Copula by Sparse Graph Regression

Tianyi Zhou
[1] Centre for Quantum Comp. & Intelligent Sys.
University of Technology, Sydney, Australia
[2] Computer Science & Engineering,
University of Washington, Seattle, WA, USA
tianyi.david.zhou@gmail.com

Dacheng Tao
Centre for Quantum Comp. & Intelligent Sys.
Faculty of Engineering & IT,
University of Technology, Sydney
235 Jones Street, Ultimo, NSW 2007, Australia
dacheng.tao@uts.edu.au

ABSTRACT

This paper proposes multi-task copula (MTC) that can handle a much wider class of tasks than mean regression with Gaussian noise in most former multi-task learning (MTL). While former MTL emphasizes shared structure among models, MTC aims at joint prediction to exploit inter-output correlation. Given input, the outputs of MTC are allowed to follow arbitrary joint continuous distribution. MTC captures the joint likelihood of multi-output by learning the marginal of each output firstly and then a sparse and smooth output dependency graph function. While the former can be achieved by classical MTL, learning graphs dynamically varying with input is quite a challenge. We address this issue by developing sparse graph regression (SpaGraphR), a non-parametric estimator incorporating kernel smoothing, maximum likelihood, and sparse graph structure to gain fast learning algorithm. It starts from a few seed graphs on a few input points, and then updates the graphs on other input points by a fast operator via coarse-to-fine propagation. Due to the power of copula in modeling semi-parametric distributions, SpaGraphR can model a rich class of dynamic non-Gaussian correlations. We show that MTC can address more flexible and difficult tasks that do not fit the assumptions of former MTL nicely, and can fully exploit their relatedness. Experiments on robotic control and stock price prediction justify its appealing performance in challenging MTL problems.

Categories and Subject Descriptors

K.4.1.4 [**Computing Methodologies**]: Machine Learning—*Learning Paradigms, Multi-task Learning*; K.4.3.5.3 [**Computing Methodologies**]: Machine Learning Approaches—*Learning in Probabilistic Graphical Models*

General Terms

Machine Learning, Statistics

Keywords

Multi-task learning; Copula; Semi-parametric model; Sparse Structured learning; Covariance Regression

KDD'14, August 24–27, 2014, New York, NY, USA.

http://dx.doi.org/10.1145/2623330.2623697.

1. INTRODUCTION

Effectiveness of multi-task learning (MTL) has been broadly proved in massive real applications such as gene expression analysis [41][24], brain activity prediction [22], collaborative filtering [38], information retrieval [25] and economic forecasting [13]. MTL studies how to exploit the underlying task relatedness in learning multiple tasks' models so that tasks can benefit from each other. Precisely, let $x \in \mathcal{X}$ be input features in domain $\mathcal{X} \subseteq \mathbb{R}^p$ and $y = \{y_i\}_{i=1}^k \in \mathcal{Y}$ be outputs of k tasks in domain $\mathcal{Y} = \prod_{i=1}^k \mathcal{Y}_i \subseteq \mathbb{R}^k$. Given associated k training sets $\{X_i, Y_i\}_{i=1}^k$ with $X_i \in \mathbb{R}^{n_i \times p}$ and $Y_i \in \mathbb{R}^{n_i \times 1}$, in machine learning it is of general interest to learn a joint conditional probability model $p(y|x)$, which contains all predictive information of outputs y given x. Classical MTL methods study the case when $p(y|x)$ is a product distribution $\prod p(y_i|x)$ with each component a mean regression model $f_i(x) : \mathcal{X} \to \mathcal{Y}_i$ plus an i.i.d. Gaussian noise $\epsilon_i \sim \mathcal{N}(0, \sigma_i^2)$

$$p(y|x) = \prod_{i=1}^k p(y_i|x), y_i|x \sim \mathcal{N}\left(f_i(x), \sigma_i^2\right), \qquad (1)$$

where $y_i = f_i(x) + \epsilon_i$. Note the above model reduces the model complexity of $p(y|x)$ on the expense of ignoring the dependency among outputs. Two families of MTL methods have been developed based on (1) and exploit task relatedness in different ways.

1.1 Classical Multi-task Learning

We first consider the case when $f_i(x)$ is a parametric function $f_i(x; w_i)$ with parameter w_i. While maximizing the likelihood $p(y|x; W)$ with $W = \{w_i\}_{i=1}^k$ leads to k independent M-estimators of $\hat{w}_i = \arg\max p(y_i|x; w_i)$, which introduce trivial single-task learning regardless of task-relatedness; maximizing the posterior of W with a joint prior $p(W)$ has motivated a line of MTL methods, which relate different tasks by endowing shared structures across $\{w_i\}_{i=1}^k$. MAP estimator $\hat{W} = \arg\max_W \log[p(W) \prod_{i=1}^k p(y_i|x; w_i)]$ yields a regularization based learning framework d

$$\hat{W} = \arg\min_W \sum_{i=1}^k \|f_i(X_i; w_i) - Y_i\|_F^2 + \lambda \mathcal{R}(W), \qquad (2)$$

where $f_i(X_i; w_i)$ extends $f_i(x; w_i)$ by applying f_i to each sample in X_i, and $\mathcal{R}(W)$ is a regularization term resulting from $p(W)$ and encoding the shared structure. In previous MTL methods, a linear function $f_i(x; w_i) = xw_i$ is usually adopted, whereas their used regularizers $\mathcal{R}(W)$ are of different types. For instance, an ℓ_1/ℓ_q ($q > 1$) norm regularizer $\mathcal{R}(W) = \sum_{i=1}^k \|w_i\|_q$ results in shared nonzero patterns among different w_i and thus yields joint feature selection across tasks [2, 23, 29]; a trace norm regularizer $\mathcal{R}(W) = \|W\|_*$ restricts vectors $\{w_i\}_{i=1}^k$ lying in a low-dimensional space

that indicates each w_i is generated from a few shared basis vectors [16, 30]; the vector fields [3] of predictor functions span a low-dimensional subspace, a graph regularizer $\mathcal{R}(W) = \mathrm{tr}(WLW^T)$ can equip $\{w_i\}_{i=1}^k$ with clustered [39], tree [17], manifold or graph weighted fusion structure [6]. Moreover, sum mixture of two regularizers for $W = P+Q$ is proposed, where P encodes shared structure and Q captures task-specific structure [1, 5, 15, 14], which includes alternating structure optimization (ASO), incoherent sparse low-rank (ISLR) formulation, robust multi-task feature learning (rMTFL), robust multi-task learning (rMTL), and so forth. In these methods, the task relatedness refers to shared model structures that lead to smaller hypothesis parameter space. When the prior $p(W)$ agrees with the true models, a better generalization performance is provably achieved.

The model (1) also gives rise to a non-parametric MTL scheme named "multi-task Gaussian process (GP) regression (MTGP)" [4], which admits more flexible form of $f_i(x)$ and assumes identical input $X \equiv X_{i\in[k]}$ in training set (so $n \equiv n_{i\in[k]}$). However, rather than imposing a GP prior to each single $f_i(x)$, a task-related GP prior is placed over all latent functions $\{f_i(x)\}_{i=1}^k$. This GP covariance is the Kronecker product of a covariance function $k^x(x, x')$ over input x and a "free form" positive semi-definite task similarity matrix $K^f \in \mathbb{R}^{k\times k}$. The key assumption added to (1) here is $\langle f_i(x), f_j(x')\rangle = K^f_{ij}k^x(x, x')$, which defines correlation between different predictive function values on different input data points. Learning hyper-parameters θ_x in k^x and K^f is conducted by likelihood maximization rather than MAP. In MTGP, it is the stationary hyperparameter K^f rather than the structure shared by task-wise parameters $\{w_i\}_{i=1}^k$ encodes the task relatedness. Instead of reducing the hypothesis parameter space to gain a better generalization performance, it abridges the output space.

So the inference of one task output on a new data point requires weighted averaging of kn outputs from the whole training set. Inverse of an $kn \times kn$ (assume identical input $X \equiv X_{i\in[k]}$ so $n \equiv n_{i\in[k]}$) covariance matrix is needed in weight computing and might cause computational burden. Learning hyperparameters θ_x of k^x and K^f is conducted by likelihood maximization rather than MAP,

$$\left\{\hat{\theta_x}, \hat{K^f}\right\} = \arg\max_{\theta_x, K^f} \log \prod_{i=1}^{k} p\left(y_i \,\middle|\, x, \theta_x, \left\{K^f_{ij}\right\}_{j=1}^{k}\right), \quad (3)$$

which can be solved by EM algorithm that requires $kn \times kn$ matrix inverse per iterate.

1.2 Motivation and Main Contributions

Although the model in (1) has long been favored and thoroughly studied in former MTL research, its effectiveness is only limited to tasks that are mean regressions with independent Gaussian noise.

Firstly, in a wider range of MTL problems, the k tasks may vary in diverse types and in different combinations. This implies the conditional marginal $p(y_i|x)$ may be non-Gaussian, or the marginals for different outputs y_i are from different classes of distributions. In the following, we discuss some non-Gaussian examples of $p(y_i|x)$. Comparing to mean regression with Gaussian noise, median regression with symmetric heavy-tailed noise such as Cauchy or Laplace is more robust to outliers in practice; quantile regression with asymmetric heavy-tailed noise is commonly used in ecology and econometrics to discover input-output relationship when input has weak ties with output mean; Poisson distribution is naturally used in count data regression; Gamma distribution is the best model for studying the emission mechanism of gamma-ray burst data; mixture model is better to fit a multi-mode distribution that can capture $p(y_i|x)$ where y_i is the prediction from multi-model or multi-expert [27], which could be multiple user groups in recommendation system [19], multiple mechanisms resulting in brain activities, or multiple systems for economics forecasting. Thus a flexible and expressive multi-task learning model should allow the combination of such different choices of $p(y_i|x)$. However, most previous MTL models cannot provide such rich possible options of $p(y_i|x)$.

Secondly, correlations among output variables $\{y_i\}_{i=1}^k$ are ignored in (1) to obtain a decomposable model in learning. However, they are natural task-relatedness that are possible to be directly estimated from data and improve prediction. Instead of employing a joint likelihood to capture such relatedness, exploration of task-relatedness in previous MTL relies on the joint prior of parameters or prediction functions for multiple models. In this case, it is required either to know them (e.g., joint prior $p(W)$ or K^f) in advance, or to learn them indirectly from data in a complicated fashion, like EM algorithm. Both are difficult.

In contrast, if we consider about learning the joint likelihood as a multivariate Gaussian $p(y|x)$, a direct idea embedding output correlations in $p(y|x) = p(\epsilon = y - f(x))$ is to posit dependent noises $\{\epsilon_i\}_{i=1}^k$. We also face difficulty because this demands learning a covariance function $\Sigma(x) : \mathcal{X} \to \mathbb{R}^{k\times k}$, which is a difficult open problem in former studies [10, 20]. It becomes even harder when marginals $\{p(y_i|x)\}_{i=1}^k$ are allowed to be arbitrary continuous densities to express more general tasks, because in this case the joint density $p(y|x)$ falls into a broad category of general distributions, in which the output correlations are usually hard to be even parametrized.

In this paper, we propose multi-task copula (MTC) that can choose arbitrary continuous marginal distributions $p(y_i|x)$ for different tasks to build a joint predictive distribution $p(y|x)$, which also encodes the output correlations as a decomposable parametric part $c(\cdot|x)$ in its density function. Thanks to the flexibility of copula model, MTC overcomes the aforementioned two main drawbacks of previous MTL methods, and can handle more general task types as well as fully exploit their relatedness via output correlations. A two-stage learning scheme for MTC will be advocated, in which we first learn the marginal $p(y_i|x)$ for each task independently, and then learn the copula density $c(\cdot|x)$ encoding outputs dependency. One merit of this scheme is that previous MTL methods can be seamlessly incorporated into the first stage, so the task relatedness can be encoded and fully exploited by both joint prior from previous MTL and joint likelihood in MTC. Another merit is that, in the special case of Gaussian copula, learning the non-Gaussian correlations among outputs encoded in $c(\cdot|x)$ can be reduced to estimating a Gaussian covariance on each given x.

In order to provide a reliable estimation of the Gaussian covariance function of x that defines the Gaussian copula, we develop an efficient non-parametric estimator "sparse graph regression" (SpaGraphR) to predict input-dependent Gaussian covariance, whose inverse (i.e., precision matrix) is sparse and encodes a structured dependency graph. The zero entries correspond to the conditional independence between outputs on the graph. In MTC, rather than estimating a stationary dependency graph by "covariance selection" [7], SpaGraphR enables the graph of y to dynamically vary with input x and thus provides a more expressive and flexible model for describing output correlations. Such local covariance has been broadly observed in various real problems, and verified to be a helpful information for prediction. For example, the correlations of different factors describing climate or market behaviors are always changing with time and locations. The prediction of these factors essentially relies on these changes in their correlations.

Similar problem of learning covariance function has rarely been studied before. SpaGraphR updates graphs on all given input data

points in a hierarchical order and is a result of kernel smoothing in conjunction with ℓ_1 regularized likelihood maximization. In learning algorithm, it starts with sparse precision matrices on a small set of representative points, and then updates the precision matrices on other points for several rounds by a fast proximal operator. In each round, a few new points joined in the set of points with updating graphs, and thus the graph estimator is refined in a coarse-to-fine multi-resolution style. Each update merely requires small matrix multiplications and entry-wise soft-thresholding. Hence, SpaGraphR has promising efficiency on big data. Experiments on challenging real problems, i.e., robotic control and stock prediction, rigidly demonstrate the effectiveness of MTC and SpaGraphR in MTL problems.

2. MULTI-TASK COPULA

Table 1: Comparison between MTL and MTC models.

	Marginals $p(y_i\|x)$	Joint Prior $p(W)$	Outputs Dependency $c(\{F_i(y_i)\}_{i=1}^k\|x)$
MTL	Gaussian/Logistic	Allow	-
MTC	Any continuous dist.	Allow	Allow

We propose MTC that tailors the copula model [35] to the MTL problem as

$$p(y|x) = c\left(F_1(y_1), F_2(y_2), \cdots, F_k(y_k)|\, x\right) \prod_{i=1}^{k} p(y_i|x), \quad (4)$$

where $F_i(y_i)$ is the cumulative distribution function (CDF) of output y_i associated with density $g_i(y_i) = p(y_i|x)$ so that $F_i(y_i) \equiv \Pr(z \leq y_i|x) \equiv \int_{-\infty}^{y_i} g_i(z)\mathrm{d}z$. The copula density function $c(\cdot|x)$ takes all marginal CDFs $\{F_i(y_i)\}_{i=1}^k$ as its augments, and maintains the output correlations in a parametric form conditioned on x. As a semi-parametric model for joint prediction, MTC is easier to learn and less prone to over-fitting than a fully non-parametric model, while more expressive and flexible than a fully parameterized model. In theory, let $F(y)$ be the joint CDF, (1) can be derived from the k^{th} order partial derivative of a copula function $C(\cdot)$.

DEFINITION 1. *Let $Y_1, \ldots, Y_k$ be real random variables marginally uniformly distributed on $[0, 1]$. A copula function $C : [0, 1]^k \to [0, 1]$ is a joint (cumulative) distribution*

$$C\left(y_1, \ldots, y_k\right) = F(y) \equiv \Pr\left(\bigwedge_{i=1}^{k} Y_i \leq u_i\right). \quad (5)$$

By Sklar's seminal theorem [35], any joint CDF $F(y)$ can be represented as a copula function $C(\cdot)$ of its univariate marginals $\{F_i(y_i)\}_{i=1}^k$. Moreover, $C(\cdot)$ is unique in the case of continuous marginals. Furthermore, the converse is also true, i.e., any copula function defined on a combination of any marginal functions gives a valid joint distribution with the same marginals.

This property is critical to MTC. Firstly, it provides a theoretical guarantee that MTC is able to model the multi-task learning model $p(y|x)$ by arbitrary joint joint distribution with continuous marginals. So it overcomes the two limitations of previous MTL models mentioned in the beginning of Section 1.2. Secondly, it indicates that the model is decomposable, so we can separately learn the marginal prediction model $p(y_i|x)$ for each task and the then learn the outputs dependency conditional on x encoded in $C(\cdot)$. In addition, we do not need to estimate the partition function for normalization in MTC. The copula model $p(y|x)$ (4) automatically turns into a legal joint prediction model taking output correlations into account.

Theoretically, we can construct any legal copula $C(\cdot)$ by a copula trick derived from inverting Sklar's theorem. For example, Gaussian copula [9, 21]

$$C\left(\{F_i(y_i)\}_{i=1}^k\right) = \Phi_\Sigma\left(\{\Phi^{-1}\left(F_i(y_i)\right)\}_{i=1}^k\right), \quad (6)$$

where Φ is the standard normal distribution and Φ_Σ is the zero mean multivariate Gaussian with covariance matrix Σ, and Archimedean copula [26]

$$C\left(\{F_i(y_i)\}_{i=1}^k\right) = \psi\left(\sum_{i=1}^{k} \psi^{-1}\left(F_i(y_i)\right)\right), \quad (7)$$

where ψ is the so called generator function with a single parameter are two families of copulas widely applied in practice. Both have the strength to model dependency in high dimensions (i.e., with large k). While Archimedean copula is easier to estimate due to its single parameter, Gaussian copula is more powerful for expressing the sophisticated dependency graph structure.

When the structure of a graphical model $G = (V, E)$ (where V is the vertex set and E is the edge set) is known in advance, we can decompose the joint density into the product of several components based on local copulas. Two recent examples are tree-structured copula [18] with density

$$g(y) = \left[\prod_{i=1}^{k} f_i(y_i)\right] \prod_{(i,j)\in E} c_{ij}\left(F_i(y_i), F_j(y_j)\right), \quad (8)$$

where c_{ij} is a bivariate copula density associated with y_i and y_j, and copula Bayesian network (CBN) [8] with density

$$g(y) = \prod_{i=1}^{k} R_{c_i}\left(F_i(y_i), F_{i1}(\mathrm{pa}_{i1}), \ldots, F_{i1}(\mathrm{pa}_{ik_i})\right) f_i(y_i), \quad (9)$$

where $\{\mathrm{pa}_{ij}\}_{j=1}^{k_i}$ is the parents of node i and R_{c_i} is the so called "copula ratio" computed from a local copula associated with y_i and its parents.

It is not hard to verify that the density functions of all the above copula models can be written in the same form as $p(y|x)$ in MTC (4), i.e., a parametric component [1] encoding the conditional dependency of y multiplied by the product of all marginals $\{p(y_i|x)\}_{i=1}^k$. In addition, since all of them are capable of handling an arbitrary number of outputs, after they have been learned from training data, they can be freely plugged into MTC (4) and results in a joint prediction model.

In Table 1, we provide a comparison between classical MTL (1) and MTC (4) from the perspective of model. In addition, an illustration of MTC is also given in Figure 2.

3. TWO-STAGE LEARNING FOR MULTI-TASK COPULA

Note the parameters in both $c(\cdot|x)$ and the marginals $\{p(y_i|x)\}_{i=1}^k$ of copula models introduced before are assumed to depend on x in MTC. Therefore, before producing a valid prediction model, these

[1] In CBN it is not always a legal copula density $c(\cdot|x)$, but can still be treated as a parametric component conditioned on x in MTC, and thus causes no difference in learning and inference.

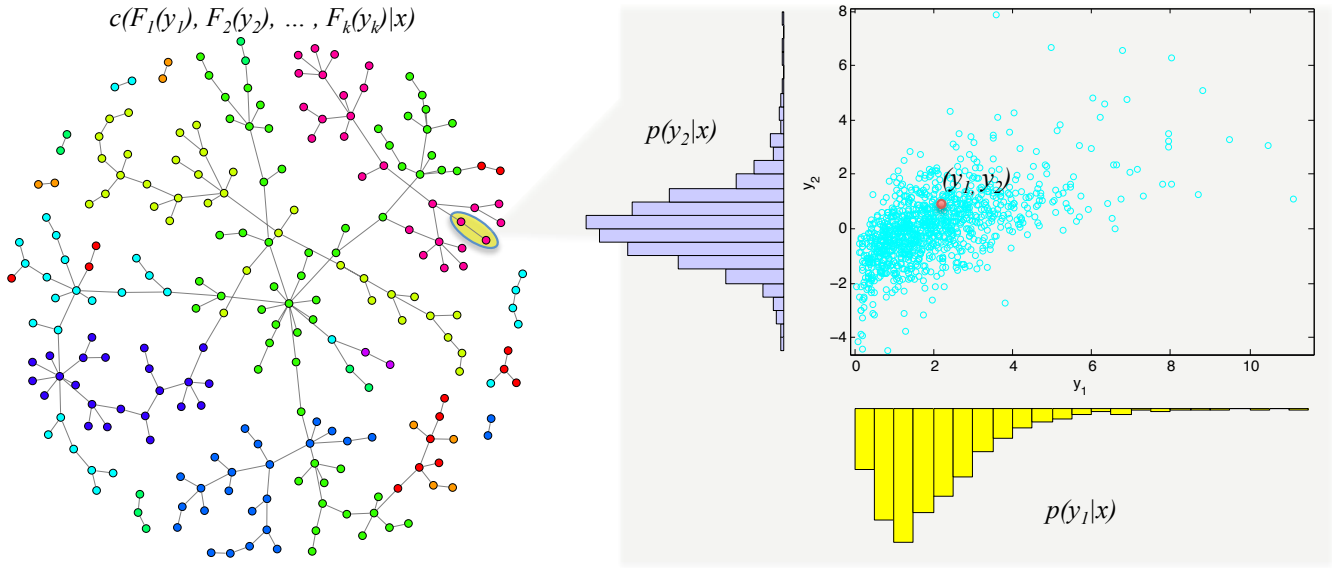

Figure 1: Illustration of multi-task copula (4) in generating outputs y given inputs x. Left: the copula density $c\left(F_1(y_1), F_2(y_2), \cdots, F_k(y_k)|\, x\right)$ as a function of x and defined by an outputs dependency graph. Right: the marginal conditional density for y_1 and y_2 can be any continuous density functions, and MTC defines the dependent joint distribution of y, from which we draw a sample (y_1, y_2). MTC fully captures the outputs correlations varying with inputs, and fits the conditional marginals by arbitrary continuous density functions. Thus it provides an expressive joint likelihood model with small complexity.

parameters have to be learned as parametric or non-parametric functions of x from training data. According to the "distribution generation" mechanism of copula suggested by Sklar's theorem, we adopt a two-stage learning scheme that estimates the marginal distributions $\{p(y_i|x)\}_{i=1}^k$ at first and then the parameters in $c(\cdot|x)$ as functions of x to build the joint prediction model.

A primary advantage of this scheme is that its first stage is exactly equal to likelihood or posterior maximization ignoring outputs dependency, and thus can be accomplished by off-the-shelf classical MTL methods in Section 1.1 or their trivial variants. For example, in the case of robust regression, we consider a general task with output $y_i = f_i(x) + \epsilon_i$, where noise ϵ_i is a random variable obeying symmetric (for mean regression) or asymmetric (for quantile regression) distribution, and function $f_i(x)$ is a constant computed from x. When $f_i(x)$ is assumed to have a parametric form $f_i(x; w_i)$ like linear or log-linear functions, the parameters $\{w_i\}_{i=1}^k$ sharing structures can be immediately learned by the regularization based MTL (2). When a more flexible non-parametric function $f_i(x)$ is preferred, the multi-task GP regression can be applied to achieve k predictive functions with related GP priors.

In addition, in the case of multi-model or multi-expert, we can consider a more general task whose output y_i obeys a conditional 1-D Gaussian mixture model such that

$$p(y_i|x) = \sum_{j=1}^{m} \frac{w_j}{\sigma\sqrt{2\pi}} \exp\left\{\frac{(y_i - f_{ij}(x))^2}{2\sigma^2}\right\}. \qquad (10)$$

When $f_{ij}(x) = x w_{ij}$, each task is exactly the "Gaussian mixture regression" (GMR) [12], and an MAP estimator of W with any joint prior $p(W)$ mentioned in Section 1.1 can result in an EM algorithm, whose M-step invokes the regularized MTL (2). When $f_{ij}(x)$ is selected as a non-parametric function, each task collapses to the "mixture of Gaussian processes" (MGP) model [33]. The GPs associated with different tasks can be related by the same trick used in multi-task GP regression via positing a task-similarity matrix K^f. An EM algorithm is also required in learning K^f and kernel parameters. Since the above EM algorithms can be directly derived by standard procedures, we will not go into their details in this paper and leave them to a longer version.

Therefore, the first stage allows lots of flexible choices of tasks and their corresponding marginal prediction models. Moreover, the first stage enables MTC to take the advantage of the task relatedness exploited by joint prior in classical MTL methods, whose merits thus can be fully inherited in MTC. Together with the task relatedness complied in the output dependency graph, which will be learned in the second stage, MTC can achieve a considerable performance boost. This is because the output dependency encoded by $c(\cdot|x)$ and the task-relatedness encoded by joint prior in classical MTL play independent roles in MTC model, and thus cannot be transferred to each other. To see this, we investigate the two terms on the right hand side of (4). While the outputs dependency is encoded by the first term $c(\cdot|x)$, the coherent latent structures exploited by classical MTL are embedded in the second term, i.e., the product of marginals. It is worthy noting that the marginals can also be trained independently as single tasks, and the MTC can still give promising prediction. The benefit of doing this is we can enjoy faster speed from divide-and-conquer or distributed computation naturally in this stage.

After the first stage, we obtain the estimates of all the marginal densities, as well as those of their associated cumulative functions $\{\hat{F}_i(y_i)\}_{i=1}^k$. In the second stage, we estimate copula density $c(\cdot|x)$ in the form of $c\left(\{F_i(y_i)\}_{i=1}^k \middle|\, \Theta(x)\right)$ from the training set

$$\left\{Z \equiv \{\hat{F}_i(Y_i)\}_{i=1}^k, X\right\}, \qquad (11)$$

where the copula parameter $\Theta(x)$ is a function of x. For an Archimedean copula with only one parameter, and copula models built from local

copulas defined on a given graphical model, the regular maximum likelihood estimate is able to achieve an accurate estimate $\hat{\Theta}(x)$.

In this paper, an primary challenge is to learn a locally smooth or piece-wise linear covariance function $\Sigma(x)$ in accompany with sparse precision $S(x) = (\Sigma(x))^{-1}$ for Gaussian copula (6) applied to MTC. The main reason for choosing Gaussian copula here is its capability of modeling high dimensional distributions. Learning Gaussian copula in MTC requires a simultaneous learning of the graph structure and the corresponding edge weights, both varying with x. This problem is extremely challenging because: 1) a parametric $\Sigma(x)$ easily suffers from over-fitting in large k and p case, for example, k^2p parameters are required to be estimated even when each entry of $\Sigma(x)$ is a linear function of x; 2) a non-parametric $\Sigma(x)$ by kernel smoothing of local covariance matrices is unlikely to also be sparse; 3) point-wise covariance selection based on only one sample of y on each point x is seriously ill-posed and computationally intractable.

4. SPARSE GRAPH REGRESSION

In this section, we propose an efficient algorithm called "sparse graph regression" (SpaGraphR), which learns a non-parametric covariance function $\Sigma(x)$ that has sparse inverse $S(x)$ on any x in the dataset and are locally smooth. SpaGraphR leverages the local smoothness to avoid a great amount of costly computations, and achieves sparse . From the perspective of non-parametric regression, we need to estimate a sparse precision $S(x)$ for multivariate Gaussian variable $z = \{\Phi^{-1}(\hat{F}_i(y_i))\}_{i=1}^k$ from its i.i.d. samples on each point x. However, we usually have only one output sample $y^i \in \mathbb{R}^k$ on the i^{th} input sample x^i in training set $X = \{x^i\}_{i=1}^n$, which indicates only one sample z^i is available for estimating $S(x^i)$. This brings difficulty in estimation. We will firstly discuss two possible classes of covariance estimator that have been investigated recently.

Covariance selection (CS), first proposed in [7] and drawing tremendous interests recently [11, 34], aims to estimate a sparse precision matrix from insufficient samples in a high-dimensional space. In this case, maximum likelihood estimate $\hat{S} = \hat{\Sigma}^{-1}$ cannot offer a legal $\hat{S}$, because the sample covariance matrix $\hat{\Sigma}$ is singular and thus does not have an inverse. Additional ℓ_1 regularization is applied to S in CS for two purposes: 1) to obtain a valid S, and 2) to make the entries of S sparse, where the zeros indicate conditional independence and suggest a structured Gaussian Markov random field. CS is formulated as

$$\hat{S}_{\hat{\Sigma},\lambda} = \arg\min_{S\succ 0} -\log\det(S) + \mathrm{tr}(\hat{\Sigma}\mathrm{S}) + \lambda \left\|\mathrm{vec}(S)\right\|_1 . \quad (12)$$

The estimate $\hat{S}_{\hat{\Sigma},\lambda}$ has been proved to converge to the true S at different rates in different norms by recent papers. These convergence rates differs in the order of dimension p yet share the same order $\mathcal{O}(1/\sqrt{n})$ of sample number n. However, in our case $n = 1$ for each x^i, the variance of CS estimate (12) with $\hat{\Sigma} = z^i z^{i^T}$ (z^i is z corresponding to x^i) is unbearable. In addition, applying expensive CS algorithm to each data point is impractical on computation.

Different from CS, another strategy that is able to achieve a $\hat{\Sigma}$ with lower variance is to take the local smoothness of $\Sigma(x)$ into account, which results in Nadaraya-Watson (NW) estimator [28, 36] based on kernel smoothing (with kernel function $K(x,\cdot)$) of covariance, i.e.,

$$\Sigma_K(x) = \frac{\sum_{i=1}^n K(x,x^i) z^i z^{i^T}}{\sum_{i=1}^n K(x,x^i)}. \quad (13)$$

NW estimator involves the neighbors of x^i in the covariance estimation on x^i, and thus could provide a more precise estimator $\Sigma_K(x^i)$ than $z^i z^{i^T}$ under the assumption that $\Sigma(x)$ is a smooth function of x. However, in practice, the resulting $\hat{S}(x) = (\Sigma_K(x))^{-1}$ does not hold a guarantee of sparsity, so we usually cannot gain a structured graph from directly taking the inverse. A straightforward approach to obtain a sparse graph regression in this case is to plug a NW estimator into CS by replacing $\hat{\Sigma}$ in (12) with $\Sigma_K(x)$, i.e., $\hat{S}(x) := \hat{S}_{\Sigma_K(x),\lambda}$. However, its computational cost is prohibitive when applied to a great number of data points, because we have to apply CS algorithm to each point.

According to the density function of multivariate Gaussian, it is the entries in precision matrix S that encode the weights of all edges on the dependency graph. Therefore, it is more preferable to seek a smooth precision function $S(x)$ rather than a smooth covariance function $\Sigma(x)$, when the generated sparse graphs are expected to exhibit structures and edge weights smoothly varying with x. In contrast to (13), we define an NW estimator for $S(x)$ as

$$S_K(x,X) = \frac{\sum_{i\in B_\epsilon(x,X)} K(x,x^i) S(x^i)}{\sum_{i\in B_\epsilon(x,X)} K(x,x^i)}. \quad (14)$$

Note $S_K(x,X)$ is defined as the kernel smoothing of the precision matrices on training data points within the ϵ-ball $B_\epsilon(x,X) \equiv \left\{i : \|x - x^i\|_2 \le \epsilon, x^i \in X\right\}$ of x. If the neighboring precision matrices $\{S(x^i)\}_{i\in B_\epsilon(x,X)}$ are sparse, $S_K(x,X)$ is also sparse on the union of their support sets. In addition, due to the non-negative kernel weights, $S_K(x,X)$ is a symmetric positive definite (PSD) matrix if all the involved $S(x^i)_{i:i\in B_\epsilon(x)} \succ 0$ is PSD. Thus $S_K(x,X)$ is a legal precision matrix estimator encoding underlying graph structure, and will be used for outputs dependency graph prediction in SpaGraphR.

Unfortunately, $\{S(x^i)\}_{i\in B_\epsilon(x,X)}$ in (14) are unknown and need to be estimated in advance. We cannot obtain them by applying $S_K(x,X)$ again because it turns out to be the "chicken or the egg" dilemma. We also cannot apply the aforementioned CS estimate $\hat{S}_{\Sigma_K(x),\lambda}$ because the training set could be huge and thus causes heavy computational burden. Nevertheless, it is possible to start from a few graphs by applying $\hat{S}_{\Sigma_K(x),\lambda}$ to a small subset of training points $\Omega \subseteq X$, and then develop an efficient operator to update graphs on the other points from the known ones in an incremental or propagation manner. By incorporating the likelihood maximization on point x, ℓ_1 regularization in (12), and local smoothness of $S_K(x,X)$, we develop an operator $S_+(x,\Omega)$ for updating $S(x)$ in SpaGraphR

$$\begin{aligned} S_+(x,\Omega) :=\ & \eta_\lambda\left\{\left[(1-\gamma)zz^T + \gamma\left(S_K(x,\Omega)\right)^{-1}\right]^{-1}\right\} \\ =\ & \eta_\lambda\left\{\tfrac{1}{\gamma}\left[S_K(x,\Omega) - \tfrac{1-\gamma}{\gamma+(1-\gamma)z^T[S_K(x,\Omega)z]} \right.\right. \\ & \left.\left.[S_K(x,\Omega)z]\left[S_K(x,\Omega)z\right]^T\right]\right\}, \end{aligned} \quad (15)$$

where $\eta_\lambda(\cdot)$ is an matrix element-wise soft-thresholding operator $[\eta_\lambda(X)]_{i,j} = \mathrm{sgn}(X_{i,j})\max(|X_{i,j}| - \lambda, 0)$ that results in sparse update $S_+(x,\Omega)$, and γ is a parameter adjusting the trade-off between likelihood maximization and local smoothness. Kailath variant of Woodbury identity is used to obtain the equality between the two representations in (15). As an important consequence, the second representation merely requires simple matrix-vector multiplication and entry-wise substraction, and thus makes $S_+(x,\Omega)$ considerably efficient to compute.

Let's look at some motivations behind (15). Equivalently, it can be derived by applying a proximal operator [31] to PSD matrix $S_V(x,\Omega) = \left[(1-\gamma)zz^T + \gamma\left(S_K(x,\Omega)\right)^{-1}\right]^{-1}$ with ℓ_1 regular-

ization, i.e.,

$$\begin{aligned}&\text{prox}_{\lambda\|\text{vec}(S)\|_1}\left(S_V(x,\Omega)\right)=\\&\arg\min_S\left\{\|\text{vec(S)}\|_1+\tfrac{1}{2\lambda}\|S-S_V(x,\Omega)\|_F^2\right\}.\end{aligned}\tag{16}$$

According to the property of proximal operator, $S_+(x,\Omega)$ finds a sparse precision matrix (whose sparsity is controlled by λ) close to $S_V(x,\Omega)$, whose inverse is a "pseudo sample covariance" $(1-\gamma)zz^T+\gamma\left(S_K(x,\Omega)\right)^{-1}$ comprised of a rank-1 sample covariance zz^T and a covariance matrix estimated by inverting the NW estimator $S_K(x,\Omega)$. The weight $\gamma\in[0,1]$ adjusts the contributions of the two terms in building the pseudo sample covariance. When the neighboring graphs used in $S_K(x,\Omega)$ are precise with preferred sparsity, a large γ is applied so that zz^T is a minor maximum likelihood correction term to $(S_K(x,\Omega))^{-1}$. Hence the resulting $S_+(x,\Omega)$ has sparse graph structure and edge weights close to those of $S_K(x,\Omega)$ and fitting training data z well. This can be verified more clearly in the second row of (15), in which $S_V(x,\Omega)$ is comprised of the NW estimator $S_K(x,\Omega)$ and a rank-1 correction term caused by z. With a large γ, the resulting $S_+(x,\Omega)$ encodes a similar dependency graph as $S_K(x,\Omega)$, while the likelihood on the unique sample z is properly maximized and the sparsity is maintained by soft-thresholding. Hence, the proximal operator (15) allows us to update sparse precision matrices on arbitrary points given those on a small subset Ω.

In SpaGraphR, we start from a small subset Ω_0 storing the most representative points whose $S(x)$ have been estimated by CS $\hat{S}_{\Sigma_K(x),\lambda}$ as "seeds", and a sequence of disjoint subsets $\{\Omega_i\}_{i=1}^M\subseteq X$ that are ordered by their capability to represent the whole training set X. When the representative capability of a subset is measured by the squared error, the subset sequence can be built by top-down hierarchical k-means clustering. In this case, cluster centers of the second layer constitute Ω_0, and $\Omega_{t\in[M]}$ is comprised of the training points closest to the cluster centers at the $(t+2)^{th}$ layer. In iterations, $S(x)$ on points of each subset is updated given $S(x)$ estimated on all the prior subsets by (15). For example, $S(x)$ on points in subset Ω_t is updated as $\left\{S(x^j):=S_+(x^j,\cup_{l=1}^{t-1}\Omega_l)\right\}_{j\in\Omega_t}$.

In SpaGraphR, graph regression $S(x)$ on a smaller Ω places a warm start for the regression on a larger Ω, and thus the dependency graphs encoded by $S(x)$ in $\mathcal{X}$ are updated in a coarse to fine multi-resolution manner by incrementing Ω of $S_K(x,\Omega)$ in $S_+(x,\Omega)$. In addition, SpaGraphR is equal to applying a special proximal Newton-like method [31] to a CS problem (12) with $\hat{\Sigma}=S_V(x,X)$ for each involved x. Since we do not know $\hat{\Sigma}=S_V(x,X)$, in the t^{th} iterate, a quadratic approximation of CS object function (12) on $S=S_V(x,\cup_{l=1}^{t-1}\Omega_l)$ and with $\hat{\Sigma}=S_V(x,\cup_{l=1}^{t-1}\Omega_l)$ is minimized. We summarize SpaGraphR in Algorithm 4.

Algorithm 1 Sparse Graph Regression (SpaGraphR)

Input: $\left\{X,\{Y_i\}_{i=1}^k\right\}$, $\{\hat{F}_i(y_i)\}_{i=1}^k$, λ, γ, M
Output: precision matrices $S(x)$ on all points
Compute $\{Z_i\}_{i=1}^k$ by applying $z_i=\Phi^{-1}\left(\hat{F}_i(y_i)\right)$ to each sample in $\{Y_i\}_{i=1}^k$;
Generate subset sequence $\{\Omega_i\}_{i=0}^M$ by $(M+2)$-layer top-down hierarchical k-means on X;
Estimate $S(x)$ on $x\in\Omega_0$ by $\hat{S}(x)=\hat{S}_{\Sigma_K(x),\lambda}$, where any CS algorithm can be applied;
for $t=1\rightarrow M$ **do**
  Update $\{S(x)\}_{x\in\Omega_t}$:$\left\{S(x^j):=S_+(x^j,\cup_{l=1}^{t-1}\Omega_l)\right\}_{j\in\Omega_t}$;
end for

5. JOINT PREDICTION BY MAP INFERENCE

Given a new instance x, the predictive distribution of y can be obtained by plugging x into the learned MTC model (4). Specifically, the precision matrix S in Gaussian copula density is estimated by $S_K(x,X)$, while the marginals are achieved by plugging x into the classical MTL model learned in the first stage. If the goal is to achieve a distribution of y or its clustering structure, the above procedure directly provides the results. However, when exact values for outputs y are required for prediction, an additional MAP inference needs to be conducted to $p(y|x)$ in (4)

$$\hat{y}=\arg\min_y\left\{-\log c\left(\left\{F_i(y_i)\right\}_{i=1}^k\middle|x\right)-\sum_{i=1}^k\log p(y_i|x)\right\}.\tag{17}$$

Without loss of generality, we study the case of Gaussian copula, the above optimization is equal to

$$\hat{y}=\arg\min_{y:\{F_i(y_i)=\Phi(z_i)\}_{i=1}^k}\left\{\frac{1}{2}z^TSz-\sum_{i=1}^k\log p(y_i|x)\right\},\tag{18}$$

which can be solved by alternating direction method of multipliers (ADMM) [31]

$$\begin{cases}z^{t+1}:= & \arg\min_z\sum_{i=1}^k\left[\frac{p}{2}\left\|F_i(y_i^t)-\Phi(z_i)\right\|_2^2+\right.\\ & \left.q_i^k\left(F_i(y_i^t)-\Phi(z_i)\right)\right]+\frac{1}{2}z^TSz,\\ \left\{y_i^{t+1}:= & \arg\min_{y_i}\frac{p}{2}\left\|F_i(y_i)-\Phi(z_i^{t+1})\right\|_2^2+\\ & q_i^k\left(F_i(y_i)-\Phi(z_i^{t+1})\right)-\log p(y_i|x)\right\}_{i=1}^k,\\ q^{t+1}:= & q^t+p\cdot\left\{F_i(y_i^{t+1})-\Phi(z_i^{t+1})\right\}_{i=1}^k,\end{cases}\tag{19}$$

where the subscript $\cdot^t$ denotes the t^{th} iterate, $p/2$ is the weight of quadratic penalty in augmented Lagrangian, $q\in\mathbb{R}^k$ is the Lagrangian multiplier, and the diagonal of S is set to zero. In the above inference algorithm, the k outputs in y are updated independently as in classical MTL except an additional regularizer linking y_i with z_i. The k elements in z are jointly updated according to their dependency graph encoded in S and relationship to y. Dual variable q is updated by gradient ascent. In practice, we usually initialize y by classical MTL such that

$$\{y_i^0=\arg\min_{y_i}-\log p(y_i|x)\}_{i=1}^k.\tag{20}$$

So only a few subsequent iterates are required to gain an appealing joint prediction. Moreover, linear approximation could be applied to $F_i(y_i)-\Phi(z_i)$ in the updates of y and z for obtaining analytical solutions to the minimizations. Hence, the joint prediction conducted by (19) is efficient.

Another optimization algorithm for solving (18) can be derived by computing the finite difference approximation of Jacobi matrix in computing the first order gradient. In particular, if we substitute $z(y)=\{\Phi^{-1}(\hat{F}_i(y_i))\}_{i=1}^k$ into (18), the problem is reduced to an unconstrained optimization, which can be efficiently solved by gradient descent method or accelerated first order optimization like Nesterov's method. By using chain rule, the gradient of the objective function $G(y)$ in (18) can be approximated by

$$\frac{\partial G(y)}{\partial y}\approx\left[\frac{2}{h}\right]\left[z\left(y+h\right)-z\left(y-h\right)\right]^TSz-\frac{\partial\sum_{i=1}^k\log p(y_i|x)}{\partial y},\tag{21}$$

where h is a difference vector for y.

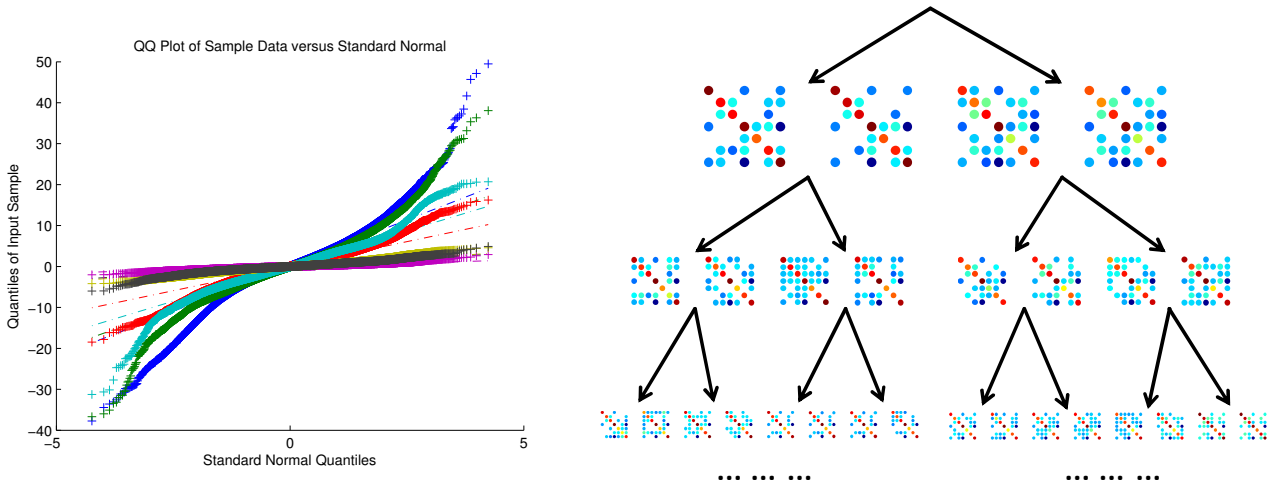

Figure 2: LEFT: Q-Q plot showing non-Gaussian noises obtained by classical MTL. RIGHT: sparse precision matrices by SpaGraphR on representative points selected by hierarchical k-means.

6. NUMERICAL RESULTS

6.1 Robotic Control

In the first experiment, we apply MTC and MTL algorithms to inverse dynamics dataset Sarcos [2] collected from a 7 degrees-of-freedom robot arm. It contains 48933 samples, each has 21 input features including 7 joint positions, 7 joint velocities plus 7 joint accelerations, and 7 joint torques as task outputs. In experiments, 400 samples are randomly selected for training and the rest are for test, in which averaged normalized MSE (nMSE) is measured for evaluation. We firstly apply single-task regression and regularization based MTL methods embedded in [40], and obtain 7 mean regression models. However, the Q-Q plot of the model noises $\{\epsilon_i = y_i - xw_i\}_{i=1}^7$ in Figure 2(a) shows that 3 outputs tend to have non-Gaussian noises, and thus implies the contradiction to assumed model in (1). This may cause sensitiveness to outliers and weaken the prediction performance.

We then apply MTC with Gaussian copula for achieving a joint conditional model $p(y|x)$ that can fit data better. In particular, we change the predictive models for the 3 outputs to a median regression with symmetric Laplacian noise and two quantile regression with asymmetric Laplacian noise, while keep the mean regression models for the rest outputs. In learning, these marginal models are estimated at first and then SpaGraphR is invoked to obtain a sparse precision function of x depicting the correlations among the Gaussian and non-Gaussian marginals. In Figure 2(b), we show the resultant sparse precision matrices on representative points identified by hierarchical k-means (2 points per cluster and 2^t clusters on t^{th} layer). It could be seem that both the graph sparse structure and the edge weights vary smoothly in local area of $\mathcal{X}$. Hence SpaGraphR provides a flexible model for outputs conditional correlations without losing robustness. In prediction, MTC has competitive performance comparing ro most classical MTL methods. The nMSE of different methods are listed in Table 2. In this case when the number of tasks is closed to the number of features, MTC exploiting outputs dependency is more effective in reducing model complexity.

[2] http://gaussianprocess.org/gpml/data/

6.2 Stock Price Prediction

In this experiment, we establish an MTC model for stock price prediction by learning from historical quote data collected from Yahoo! Finance [3]. Then we test and compare the prediction performance of MTC and other MTL methods for recently stock prices. In particular, we collect the weekly closing price of 35 famous companies' stocks from their historical quotes on NASDAQ between 01/01/2000 and 09/23/2013, which are 35 time series of stock prices in 716 weeks. For each stock, given historical prices in the pasting v weeks $\{q_{t-i}\}_{i=1}^v$, the price q_t in t^{th} week can be predicted by linear autoregression (AR) model $\hat{q}_t = \sum_{i=1}^v w_i q_{t-i} + w_0$. Hence, we sample 600 weeks with equal time intervals, and let the stock prices in these 600 weeks as responses. For each week out of the 600, we let the stock prices in the pasting 10 weeks as its input features. Therefore, for s^{th} stock, we have a dataset with outputs $Y_s \in \mathbb{R}^{600}$ and inputs $X_s \in \mathbb{R}^{600\times 10}$.

Table 2: Comparison of single-task regression (STR), dirty model for MTL [15], clustered MTL (CMTL) [39] and MTC on Sarcos dataset (nMSE).

	STR	Dirty	CMTL	MTC
nMSE	0.1828	0.1651	0.2733	0.1204

However, when predicting multiple stocks, we expect the historical prices of other stocks can also help the prediction. Therefore, we expand the input features and the AR model predicting the s^{th} stock's price in the t^{th} week turns to

$$\hat{q}_{s,t} = \sum_{i=1}^{k}\sum_{j=1}^{v} w_{i,j} q_{i,t-j} + w_{s,0}, \tag{22}$$

where $s \in [35]$ indexes the $k = 35$ stocks and $t \in [600]$ indexes the 600 weeks, and $v = 10$. The inputs of the dataset for each stock changes to the same matrix $[X_1, \ldots, X_{35}] \in \mathbb{R}^{600\times 350}$. In the experiment, we split the dataset by choosing the prices in the

[3] http://finance.yahoo.com/

first 400 weeks out of the 600 weeks as training set, and the prices in the rest 200 weeks for test. Note this setting is more challenging than random split and is closer to real problem, in which we expect to gain a reliable model for future prices only from historical prices.

Our goal here is to build a joint prediction model for all 35 stocks by not only using all of their historical prices' information captured by the AR model (22), but also by leveraging the dynamically varying correlations among different stocks. When modeling the conditional marginal $p(q_{s,t}|\{q_{i,t-j}\}_{i\in[35],j\in[10]})$ in (4) by AR model (22), an interesting and broadly verified phenomenon is that $\hat{q}_{s,t}$ in (22) performs more robust and reliable when assigned to the τ^{th} quantile (e.g., median, 25%, 75%) of the conditional marginal rather than the mean. This indicates that the conditional marginals are non-Gaussian, possibly asymmetric and heave tailed, and the conditional correlations among outputs are also non-Gaussian. The former fact leads to the failure of most MTL methods based on mean regression with Gaussian noise, while the latter fact leads to the failure of most covariance regression/selection methods based on Gaussian covariance assumption.

Fortunately, MTC is able to precisely model the non-Gaussian conditional marginals and correlations by learning the marginal model and copula density model separately. In particular, we adopt a τ^{th} quantile regression with skewed Laplace noise for each marginal model, i.e.,

$$p(y_i|x,w) = b\tau(1-\tau)\cdot \begin{cases} \exp(-b(1-\tau)(xw-y_i)), & y_i < xw; \\ \exp(-b\tau(y_i - xw)), & y_i \geq xw; \end{cases} \quad (23)$$

In the first learning stage, we impose additional ℓ_1 regularization to the coefficients w as a sparse prior. This is essentially helpful to our problem because 1) the current price of one stock is usually related to the prices of a few other stocks at a few historical time intervals rather than all stocks over all intervals; and 2) the number of training samples 400 is very close to the feature dimensions 350 and thus may introduce large variance. Given w, b can be easily obtained via MLE. The quantile τ (reported in Table 3) and the weight of ℓ_1 regularization (either 0.2 or 0.3) for each stock are achieved by hypothesis testing and cross-validation, respectively.

In the second learning stage, we compute 400 Gaussian samples Z from Y and the CDF of marginal models obtained in the first stage as training set for SpaGraphR, which learns the sparse and smooth graph function describing the dynamic graph over stocks. The sparse prior of the graph function also plays an important role here, because one stock currently is only related to a subset of other stocks in the same or related areas, and the influence of one area to another varies over time, which makes the edges in the graph changing over time.

We report both the ℓ_1 relative error $|\hat{q}-q|/|q|$ averaged over all 200 testing weeks for each stock in Table 3. The performance of marginal models before joint inference (STR) is also reported in order to make the improvement solely brought by the sparse graph function more clear. For comparison, we also apply existing MTL methods to this problem and report their performance in the same ways. Different from modeling $\hat{q}_{s,t}$ in (22) by quantile regression, all of them model it by least square regression yet with different joint priors/regularization over all w.

The results in Table 3 show that MTC successfully and precisely predict most of the stock prices over time, while the other classic MTL fail on most of them (their training errors are between $[0.5, 2]$ but the test errors are much worse, indicating their joint likelihood is very far from the truth, which can be accurately modeled by MTC). This is because the mean regression with Gaussian noise has large variance on stock data, and classical MTL cannot correctly capture and leverage the conditional non-Gaussian correlations by joint prior. Comparing to the prediction by single quantile regression models, the additional improvement of MTC prediction proves the essential role of SpaGraphR in relating multiple tasks to help prediction.

7. CONCLUSION

Multi-task learning (MTL) covers a rich class of machine learning and statistical problems expecting to let different tasks benefit each other by exploring their relatedness [32][37]. However, in order to reduce MTL to a easier regularization problem, most previous MTL methods limit the types of tasks, ignore the conditional dependency among outputs and merely rely on the pre-defined joint prior distribution of models to capture the relatedness. So they can hardly handle more complex tasks in practice, and usually achieve limited improvement over single task learning.

In this paper, we adapt copula model from semi-parametric statistics to the joint likelihood function $p(y|x)$ in MTL. The capability of copula in generating arbitrary continuous joint distribution results in an expressive model that allows combination of different types of marginal models $p(y_i|x)$ for different tasks, and an unified parametric description of outputs dependency. This structure enables us to develop a two-stage learning scheme for the proposed "multi-task copula (MTC)". While the first stage learns the marginals by using single task learning or any previous MTL method, the second stage aims at learning a conditional covariance function $\Sigma(x)$ (or precision function $\Sigma^{-1}(x)$ encoding a sparse dependency graph of y varying with inputs x.

Although this covariance regression is an open challenging problem in recent machine learning community, we proposes an efficient nonparametric estimator for $\Sigma(x)$ called "sparse graph regression (SpaGraphR)" that incorporates local likelihood maximization, kernel smoothing, and sparse structure of graph. SpaGraphR enables both the edge weighs and the sparse graph structure smoothly changing with varying x, which can be rarely achieved by previous methods. In addition, SpaGraphR starts from a few seed graphs on a small number of representative x and updates the graphs on other points from the known graphs by cheap matrix multiplication and entry-wise soft-thresholding. So it can refine the estimation accuracy of $\Sigma^{-1}(x)$ in a coarse-to-fine grained space of x efficiently for big data.

Different from previous MTL methods relying on given joint prior applied to task models, MTC automatically learns the conditional dependency of task outputs which directly serves the joint prediction. The joint prediction can be obtained by solving an easy optimization problem.

We applied MTC to robotic control and stock price prediction problems, in which there exist strong correlations between task outputs that cannot be correctly captured by previous MTL methods due to the non-Gaussian and asymmetric properties of their data distribution. In experimental comparison, we verify that the capability of both integrating different types of marginal distributions and capturing their conditional dependency lends MTC the power to outperform other MTL methods.

It is worth noting that the models presented in this paper and used in experiments are merely special instances of MTC. MTC allows rich combination of marginal models from different distribution classes or even non-parametric distributions, and arbitrary copula density functions that describe the output dependency. Similar to copula which is a "distribution generator", MTC plays a role of "model generator" for multi-task learning that can fit each task with the suitable marginal model and simultaneously fully leverage the task-relatedness encoded by both output dependency and joint

Table 3: Comparison of Dirty, CMTL, MTL with trace norm penalty, STR, MTC on stock price data (test relative ℓ_1 error).

Stock	τ	Dirty	CMTL	Trace	STR	MTC
ATT	0.1	3.9653	4.8156	2.2068	0.1239	0.0937
BOA	0.1	6.0347	4.4945	5.7678	0.0920	0.0679
Boeing	0.9	8.0096	8.3798	8.142	0.2544	0.2102
CVS	0.2	6.2283	4.8732	3.7163	0.0879	0.0841
FeDex	0.1	2.7012	9.976	5.6893	0.2353	0.2193
Ford	0.5	9.7238	2.1568	6.481	0.1142	0.1048
GE	0.4	3.9807	4.7537	2.388	0.0733	0.0707
HP	0.2	3.8119	8.3573	17.4959	0.1000	0.0910
IBM	0.1	1.9552	3.8116	4.0757	0.0997	0.0948
JPMorgan	0.2	3.2578	5.5316	2.4995	0.1555	0.1262
Macy	0.1	5.8561	5.8208	3.2293	0.1128	0.1115
Adobe	0.1	5.2789	5.1415	4.3769	0.1317	0.0814
Amazon	0.2	6.5161	0.6358	31.5366	0.3619	0.3204
Apple	0.1	8.7289	1.8109	21.0517	0.7708	0.5172
Autodesk	0.1	5.5317	8.3711	5.12	0.1936	0.1458
Chevron	0.1	2.8925	3.8882	2.8397	0.0366	0.0366
Cisco	0.2	5.2039	5.5257	5.1972	0.1619	0.1071
Costco	0.4	8.1616	5.6682	5.6519	0.4100	0.3601
Dell	0.1	7.0798	5.3801	8.6039	0.1481	0.1339
Ebay	0.9	4.6712	3.1633	3.374	0.1150	0.1050
Intel	0.1	7.1312	3.7495	7.6896	0.1004	0.0947
Maxim	0.2	4.7308	3.4045	2.473	0.1706	0.1145
Microsoft	0.9	6.7137	3.1567	4.2504	0.0485	0.0492
Oracle	0.2	16.283	4.0686	3.7193	0.0886	0.0820
Priceline	0.2	44.4155	148.1859	151.665	1.3897	1.0205
Qualcomm	0.1	2.5908	3.8174	2.5528	0.1125	0.0944
Sandisk	0.2	9.0629	5.6423	12.1588	0.1947	0.1748
SouthwestAir	0.1	12.7343	4.438	6.107	0.0878	0.0704
Target	0.9	8.8795	6.8072	5.1398	0.2976	0.2432
TI	0.1	6.3444	4.4869	4.503	0.1427	0.1085
Vodafone	0.1	14.3038	5.5485	3.5616	0.0818	0.0721
Walmart	0.2	4.6541	4.4191	3.9601	0.0551	0.0550
Wholefood	0.1	3.3218	6.0992	7.2131	0.1108	0.0857
Xilinx	0.1	6.1487	3.1605	1.6395	0.1889	0.1555
Yahoo	0.1	7.4028	6.0218	6.2141	0.2003	0.1862

prior of marginal models. Therefore, we believe lots of possible future contributions can be made to improve either MTC model for specified MTL problems or the efficiency of its learning/inference algorithms.

Acknowledgement

We would like to thank Prof. Carlos Guestrin, Prof. Ben Taskar and Prof. Emily Fox for their important suggestions in improving this draft. This work is supported by Australian Research Council Projects FT-130101457 and DP-140102164.

References

[1] R. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853, 2005.

[2] A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

[3] C. Z. J. Y. B. Lin, S. Yang and X. He. Multi-task vector field learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

[4] E. Bonilla, K. Chai, and C. Williams. Multi-task gaussian process prediction. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

[5] J. Chen, J. Liu, and J. Ye. Learning incoherent sparse and low-rank patterns from multiple tasks. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2010.

[6] X. Chen, Q. Lin, S. Kim, J. Carbonell, and E. Xing. Smoothing proximal gradient method for general structured sparse learning. In *The Conference on Uncertainty in Artificial Intelligence (UAI)*, 2011.

[7] A. Dempster. Covariance selection. *Biometrics*, 28:157–175, 1972.

[8] G. Elidan. Copula bayesian networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

[9] P. Embrechts, F. Lindskog, and A. McNeil. *Modeling dependence with copulas and applications to risk management*. Handbook of Heavy Tailed Distributions in Finance. 2003.

[10] E. Fox and D. Dunson. Bayesian nonparametric covariance regression. *arXiv:1101.2017*, 2011.

[11] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[12] Z. Ghahramani and M. Jordan. Supervised learning from incomplete data via an em approach. In *Advances in Neural Information Processing Systems (NIPS)*, 1994.

[13] J. Ghosn and Y. Bengio. Multi-task learning for stock selection. In *Advances in Neural Information Processing Systems (NIPS)*, 1997.

[14] P. Gong, J. Ye, and C. Zhang. Robust multi-task feature learning. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2012.

[15] A. Jalali, P. Ravikumar, S. Sanghavi, and C. Ruan. A dirty model for multi-task learnin. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

[16] S. Ji and J. Ye. An accelerated gradient method for trace norm minimization. In *International Conference on Machine Learning (ICML)*, 2009.

[17] S. Kim and E. Xing. Tree-guided group lasso for multi-task regression with structured sparsity. In *International Conference on Machine Learning (ICML)*, 2010.

[18] S. Kirshner. Learning with tree-averaged densities and distributions. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

[19] B. Liu, Y. Fu, Z. Yao, and H. Xiong. Learning geographical preferences for point-of-interest recommendation. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1043–1051, 2013.

[20] H. Liu, X. Chen, J. Lafferty, and L. Wasserman. Graph-valued regression. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

[21] H. Liu, J. Lafferty, and L. Wasserman. The nonparanormal: Semiparametric estimation of high dimensional undirected graphs. *Journal of Machine Learning Research*, 10:2295–2328, 2009.

[22] H. Liu, M. Palatucci, and J. Zhang. Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery. In *International Conference on Machine Learning (ICML)*, 2009.

[23] J. Liu, S. Ji, and J. Ye. Multi-task feature learning via efficient l2,1-norm minimization. In *The Conference on Uncertainty in Artificial Intelligence (UAI)*, 2009.

[24] Q. Liu, Q. Xu, V. W. Zheng, H. Xue, Z. Cao, and Q. Yang. Multi-task learning for cross-platform sirna efficacy prediction: an in-silico study. *BMC Bioinformatics*, 11:181, 2010.

[25] X. Mao, B. Lin, D. Cai, X. He, and J. Pei. Parallel field alignment for cross media retrieval. In *Proceedings of the 21st ACM International Conference on Multimedia*, MM '13, pages 897–906, 2013.

[26] A. McNeil and J. Nešlehová. Multivariate archimedean copulas, d-monotone functions and ℓ_1-norm symmetric distributions. *Annals of Statistics*, 37(5b).

[27] K. Mo, E. Zhong, and Q. Yang. Cross-task crowdsourcing. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 677–685, 2013.

[28] E. Nadaraya. On estimating regression. *Theory of Probability and its Applications*, 9(1):141–200.

[29] G. Obozinski, B. Taskar, and M. Jordan. High-dimensional union support recovery in multivariate regression. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

[30] G. Obozinski, B. Taskar, and M. Jordan. Joint covariate selection and joint subspace selection for multiple classi?cation problems. *Statistics and Computing*, 20:231–252, 2010.

[31] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 2013.

[32] N. Quadrianto, A. J. Smola, T. S. Caetano, S. V. N. Vishwanathan, and J. Petterson. Multitask learning without label correspondences. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1957–1965, 2010.

[33] C. Rasmussen and Z. Ghahramani. Infinite mixtures of gaussian process experts. In *Advances in Neural Information Processing Systems (NIPS)*, 2001.

[34] B. Rolfs, B. Rajaratnam, D. Guillot, I. Wong, and A. Maleki. Iterative thresholding algorithm for sparse inverse covariance estimation. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[35] A. Sklar. Fonctions de repartition a n dimensions et leurs marges. *Publications de l'Institut de Statistique de L'Universite de Paris*, 8:229–231.

[36] G. Watson. Smooth regression analysis. *Sankhya: The Indian Journal of Statistics, Series A*, 26(4):359–372.

[37] S. Yang, Y. Jiang, and Z.-H. Zhou. Multi-instance multi-label learning with weak label. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2013.

[38] S. Yu, K. Yu, V. Tresp, and H. Kriegel. Collaborative ordinal regression. In *International Conference on Machine Learning (ICML)*, 2006.

[39] J. Zhou, J. Chen, and J. Ye. Clustered multi-task learning via alternating structure optimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[40] J. Zhou, J. Chen, and J. Ye. *MALSAR: Multi-tAsk Learning via StructurAl Regularization*. Arizona State University, 2011.

[41] J. Zhu, B. Zhang, E. Smith, B. Drees, R. Brem, L. Kruglyak, R. Bumgarner, , and E. Schadt. Integrating large-scale functional genomic data to dissect the complexity of yeast regulatory networks. *Nature Genetics*, 40:854–861, 2008.