Pattern Recognition ▮▮▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Pattern Recognition

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

Multi-output regression on the output manifold

Guangcan Liu[a,*], Zhouchen Lin[b], Yong Yu[a]

[a]Shanghai Jiao Tong University, Shanghai 200240, PR China
[b]Microsoft Research Asia, Zhichun Road #49, Haidian District, Beijing 100190, PR China

ARTICLE INFO

Article history:
Received 8 September 2008
Received in revised form 23 February 2009
Accepted 1 May 2009

Keywords:
Regression analysis
Support vector regression
Continuous structured prediction
Manifold learning

ABSTRACT

Multi-output regression aims at learning a mapping from an input feature space to a multivariate output space. Previous algorithms define the loss functions using a fixed global coordinate of the output space, which is equivalent to assuming that the output space is a whole Euclidean space with a dimension equal to the number of the outputs. So the underlying structure of the output space is completely ignored. In this paper, we consider the output space as a Riemannian submanifold to incorporate its geometric structure into the regression process. To this end, we propose a novel mechanism, called locally linear transformation (LLT), to define the loss functions on the output manifold. In this way, currently existing regression algorithms can be improved. In particular, we propose an algorithm under the support vector regression framework. Our experimental results on synthetic and real-life data are satisfactory.

1. Introduction

Recently, structured prediction [1,2] has gained renewed attention. This technique is important in a variety of applications, such as conjoint analysis in market research and object recognition in computer vision. As the outputs are structured objects (e.g., sequences, trees and vectors) rather than independent scalars, it is important to discover the possible structure underlying the outputs. Multi-output regression [3–5], which aims at predicting an output vector $y \in R^n$ given an input feature vector $x \in R^m$, can be considered as a *continuous* structured prediction problem, where the outputs form a *continuous* and *infinite* space with specific structures.[1] There has been sparse research in multi-output regression. The main literature we are aware of is [3–5]. Previous research all formulates the problem as learning a mapping from R^m to R^n:

$$f : R^m \to R^n.$$

This is equivalent to assuming that the output can fill the whole Euclidean space R^n. However, as the outputs are correlated, they should form an output manifold with an inherent dimension smaller than the number of the outputs. For example, the outputs of $f(x) = (\sin(x), \cos(x))$ form a one-dimensional manifold (the unit circle) in R^2. So a more reasonable formulation of the multi-output regression problem is to learn a mapping from an input *manifold* to an output *manifold*

$$f : \mathcal{M}_x \to \mathcal{M}_y,$$

where $\mathcal{M}_x$ and $\mathcal{M}_y$ are the Riemannian submanifolds of R^m and R^n, respectively. To handle such a problem, both the structures of $\mathcal{M}_x$ and $\mathcal{M}_y$ should be considered. There has been a lot of literature (e.g., [6]) on the exploration of $\mathcal{M}_x$. In this work, we aim at making use of the possible structure underlying $\mathcal{M}_y$ so as to improve the current regression schemes. Although it could be regarded that the methods in [4,5] also model the output manifold in some sense, to the best of our knowledge, there has no previous work that addresses the problem from a viewpoint of geometry.

Given a training set consisting of K input–output pairs $\{(x_1, y_1), \ldots, (x_K, y_K)\}$, where $x_i \in \mathcal{M}_x \subseteq R^m$ and $y_i \in \mathcal{M}_y \subseteq R^n$, $y_i = [y_{1i}, \ldots, y_{ni}]^t$, we want to learn a multivariate function $f(x) = [f_1(x), \ldots, f_n(x)]^t$ that maps the inputs to the outputs. Here $[\cdot]^t$ denotes the transpose of a matrix or a vector. Following the standard regression framework, we may estimate the mapping function by minimizing

$$\mathcal{R}(f) + C \sum_{i=1}^{K} \mathcal{L}(f(x_i), y_i), \tag{1}$$

* Corresponding author. Tel.: +86 10 58963089; fax: +86 10 88099511.
E-mail addresses: roth@sjtu.edu.cn (G. Liu), zhoulin@microsoft.com (Z. Lin), yyu@apex.sjtu.edu.cn (Y. Yu).

[1] It is worth noting that most currently existing structured prediction algorithms such as [2] only deal with *discrete* and *finite* output set. For the approaches that model the output space as a finite discrete set, a disadvantage is that it will be very time-consuming if the cardinality (total number of different output objects) of the output set is large. For example, in our multi-label classification experiments, there are 2^L possible outputs when there are L labels. Whereas, it is efficient to solve the problem by the approach proposed in this work.

0031-3203/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2009.05.001

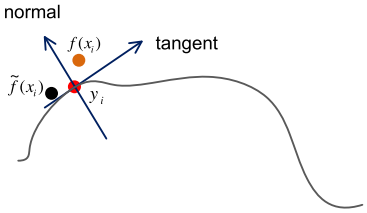

Fig. 1. The loss should be related to the local shape of the output manifold. In this example, the Euclidean distances between the two predictions ($f(x_i)$ and $\tilde{f}(x_i)$) and the target output (y_i) are the same. However, their losses should be different as $\tilde{f}(x_i)$ is much closer to the target manifold than $f(x_i)$, hence a better prediction.

where $\mathscr{L}$ is some loss function that measures the discrepancy between the predicted output $f(x_i)$ and the target output y_i, $\mathscr{R}$ is a function for regularization, and C is a parameter that balances the strength between the loss and the penalty of regularization. Previous multi-output regression algorithms [3,4] naively define the loss function as the sum of the loss w.r.t. each output variable (hence named "Naïve" mechanism herefrom). In [3], the loss term is defined as $\mathscr{L}(f(x_i),y_i)=\|f(x_i)-y_i\|_{\ell_2}=\sum_{j=1}^{n}(f_j(x_i)-y_{ji})^2$. And [4] adopts $\mathscr{L}(f(x_i),y_i)=\sum_{j=1}^{n}|f_j(x_i)-y_{ji}|_\varepsilon$, where $|\cdot|_\varepsilon$ denotes the ε-insensitive loss function.[2] So their loss functions can be viewed as mappings from $R^n\times R^n$ to R^+:

$$\mathscr{L}: R^n\times R^n\to R^+,$$

where R^+ denotes the set of nonnegative real numbers. These naïve mechanisms essentially ignore the local shape of the output manifold $\mathscr{M}_y$. To show that the loss should not be defined without referring to the shape of $\mathscr{M}_y$, we illustrate in Fig. 1. Given two predictions $f(x_i)$ and $\tilde{f}(x_i)$ of a target output y_i, their losses should be different even the Euclidean distances between the two predictions and the target output are the same: $\tilde{f}(x_i)$ should be viewed a better prediction of y_i than $f(x_i)$ as it is closer to the output manifold.

To take into account of the local shape of the output manifold, we should define the loss $\mathscr{L}$ as a mapping from $R^n\times\mathscr{M}_y$ to R^+:

$$\mathscr{L}: R^n\times\mathscr{M}_y\to R^+.$$

Inspired by the moving frame method in differential geometry [7], which uses tangent and normal vectors of a manifold as the local coordinates, we propose using local SVD (singular value decomposition) [8] to indicate the local orientation and shape of the manifold, and define the loss function at an output y_i in its local coordinates. This is equal to defining the loss after applying a linear transformation at y_i. The transformation is linear and local, hence entitled locally linear transformation (LLT). LLT has some appealing statistic and geometric properties. We can prove that the coordinates we choose for LLT is the *optimal* choice in the sense of the sum of the standard deviations of the outputs. It is convenient to adopt LLT in currently existing regression frameworks. In particular, we propose a multi-output regression algorithm based on the well-established support vector regression (SVR) [9] framework. Our experiments demonstrated on synthetic data and multi-label classification tasks obtain promising results.

[2] The method in [4] uses an estimate of cross-covariance among the outputs to incorporate the relationship among the outputs. So it also seems to capture the manifold structure in a global way. However, it is not easy to capture the manifold structure globally, as there seldom exists a global structure in nonlinear manifolds. Our approach models the manifold structure locally and our experiments will show that our approach works much better than [4].

The remainder of this paper is organized as follows. Section 2 introduces LLT. Section 3 presents a multi-output regression that realizes LLT under the SVR framework. Section 4 shows the experimental results. And Section 5 concludes this paper.

2. LLT: locally linear transformation

2.1. Choosing coordinate systems

As clarified in Introduction, the loss should be defined by choosing a coordinate system, either implicitly or explicitly. Suppose the orthonormal bases of the chosen Cartesian coordinate system at y_i are

$$\theta(y_i)=\{\theta_j(y_i)|j=1,\dots,n\}.$$

Then the loss at y_i should be defined as the function of the following quantities:

$$\{[\theta_j(y_i)]^t(f(x_i)-y_i)|j=1,\dots,n\}.$$

It is easy to see that in [3,4], the chosen coordinate system is

$$\theta_j(y_i)=e_j,\quad j=1,\dots,n,$$

where e_j is a vector whose j-th element is 1 and the rest elements are all 0. However, as argued in Introduction, it is not a good choice. A better choice should change with the local shape of the output manifold at y_i. Inspired by the moving frame method in differential geometry, a coordinate system consisting of tangent and normal vectors is a good choice because it can characterize the local orientation and shape of the output manifold.

However, the choice of tangent and normal vectors is not unique. We have to choose an "optimal" one. As y_i is actually a random point in R^n if it is not associated with the inputs, we want the variance measured in the new coordinate system be minimized such that the prediction can be made the most accurate. To this end, we use the sum of standard deviations of the output $\tilde{y}_i$ in the new coordinate $\theta=\{\theta_j|j=1,\dots,n\}$:

$$SStd(\theta)=\sum_{j=1}^{n}\sigma(\tilde{y}_{ji}),$$

where $\tilde{y}_{ji}=\theta_j^t y_i$ and $\sigma(\cdot)$ is the standard deviation of a random variable. A smaller $SStd(\theta)$ implies that the coordinate system θ is better for prediction. This doctrine has been proven in the traditional single-output regression [10]. In the case of multi-output regression, the doctrine is still reasonable although it is hard to rigorously prove it. Our experiments will verify its effectiveness.

Let Σ_i be the covariance matrix of the probability density $\mathscr{P}_i$ around y_i, and $\pi(y_i)=\{\pi_j(y_i)|j=1,\dots,n\}$ be the set of orthonormal eigenvectors of Σ_i, then we can prove that:

Theorem 1 (*Optimality of local principal directions*). *The coordinate system $\pi(y_i)$ has the minimum sum of standard deviation among all possible orthonormal coordinate systems at y_i, i.e.,*

$$SStd(\theta(y_i))\geq SStd(\pi(y_i)),\quad\forall\theta(y_i). \tag{2}$$

And if the eigenvalues of Σ_i are all simple, the equality holds only when $\theta(y_i)=\pi(y_i)$.

The proof can be found in Appendix. This theorem implies that the principal directions of Σ_i form the optimal local coordinate system.

Another advantage of using the local principal directions is that the "new" component variables $\{\tilde{y}_{1i},\dots,\tilde{y}_{ni}\}$ become unrelated to each other, and further be independent of each other if $\mathscr{P}_i$ is

Please cite this article as: G. Liu, et al., Multi-output regression on the output manifold, Pattern Recognition (2009), doi:10.1016/j.patcog.2009.05.001

Gaussian (locally Gaussian assumption) [11]. This property is very useful because in this case it is reasonable to compute the loss as the sum of loss in each component variable.

2.2. Computing ℓ_p-based loss functions on the output manifold

The ℓ_p-based loss functions are frequently used in machine learning community. So we also follow this tradition. However, our loss function will be defined in the local coordinates of the output manifold. For a point y_i with its local shape characterized by the coordinate system $\pi(y_i)$, we define the loss in the following way:

$$\mathscr{L}(f(x_i),y_i) = \|f(x_i)-y_i\|_{\pi(y_i);\ell_p} = \left(\sum_{j=1}^{n} |[\pi_j(y_i)]^t(f(x_i)-y_i)|^p\right)^{1/p},$$

where $\|\cdot\|_{\pi;\ell_p}$ denotes the ℓ_p norm in the coordinate π. The ε-insensitive loss can be defined similarly by replacing $|\cdot|^p$ with $|\cdot|_\varepsilon^p$ in the above equation (see Fig. 2).

To make the prediction mapping f favor fitting the shape of the output manifold $\mathscr{M}_y$, we require that the loss $L(f(x_i),y_i)$ becomes small when the predicted points move towards normal or tangent of $\mathscr{M}_y$. To achieve this, p must be less than 2. So it is not encouraged to apply ℓ_p ($p \geq 2$) regression methods to the output manifolds. Moreover, one may expect using a local distance metric A_i (e.g., $A_i = \Sigma_i^{-1}$) to model the structures at y_i: $\mathscr{L}(f(x_i),y_i)=(f(x_i)-y_i)^t A_i (f(x_i)-y_i)$. However, for an output space with a manifold structure, Σ_i is usually singular, making such definition meaningless.

3. LLT-SVR: LLT-based support vector regression

LLT is a general approach for defining the loss function of regression schemes. It can be combined with different regression algorithms. As an example, we present how to implement LLT under SVR. Given K training examples $\{(x_1,y_1),\ldots,(x_K,y_K)\}$, $y_i \in \mathscr{M}_y \subseteq R^n$ we want to learn a map $f(x)=[f_1(x),\ldots,f_n(x)]^t$ by solving the optimization problem as in Eq. (1). For simplicity, we begin with the case that $\{f_i(x)\}_{i=1}^n$ are linear functions, i.e., $f_i(x)=w_i^t x + b_i$. Provided with the coordinate systems $\{\pi(y_i)=\{\pi_1(y_i),\ldots,\pi_n(y_i)\}\}_{i=1}^K$ at each y_i, similar to SVR, we solve the following convex optimization problem:

$$\min_{\{w_j,b_j\}_{j=1}^n} \frac{1}{2}\sum_{j=1}^{n} w_j^t w_j + C\sum_{i=1}^{K} |f(x_i)-y_i|_{\pi(y_i);\varepsilon},$$

where $|f(x_i)-y_i|_{\pi(y_i);\varepsilon} = \sum_{j=1}^n |[\pi_j(y_i)]^t(f(x_i)-y_i)|_\varepsilon$. As $\pi_j(y_i) \neq e_j$, the above optimization problem *cannot* be decomposed into n independent subproblems. So it should be solved jointly. Let $W=[w_1,\ldots,w_n]$ and $b=[b_1,\ldots,b_n]^t$, the above problem can be rewritten as

$$\begin{aligned}
\min_{W,b}\quad & \frac{1}{2}\|W\|_F^2 + C\sum_{j=1}^{n}\sum_{i=1}^{K}(\xi_{ij}^* + \xi_{ij}) \\
\text{s.t.}\quad & \pi_j(y_i)^t y_i - (W\pi_j(y_i))^t x_i - \pi_j(y_i)^t b \leq \varepsilon + \xi_{ij}, \\
& (W\pi_j(y_i))^t x_i + \pi_j(y_i)^t b - \pi_j(y_i)^t y_i \leq \varepsilon + \xi_{ij}^*, \\
& \xi_{ij} \geq 0, \quad \xi_{ij}^* \geq 0, \quad i=1,\ldots,K, \quad j=1,\ldots,n.
\end{aligned}$$

where $\|\cdot\|_F$ is the Frobenius norm. To facilitate the implementation, we reformulate the above optimization problem into a vector form. Let w be an N-dimensional vector that refers to the vectorization of W formed by stacking the columns of W into a single column vector, where $N = n \cdot K$, and

$$\begin{aligned}
\{\tilde{\pi}_1,\ldots,\tilde{\pi}_N\} &= \{\pi_1(y_1),\ldots,\pi_n(y_1),\pi_1(y_2),\ldots, \\
&\qquad \pi_n(y_2),\ldots,\pi_1(y_K),\ldots,\pi_n(y_K)\}, \\
\{\tilde{y}_1,\ldots,\tilde{y}_N\} &= \{\pi_1^t(y_1)y_1,\ldots,\pi_n^t(y_1)y_1,\ldots,\pi_1^t(y_K)y_K,\ldots,\pi_n^t(y_K)y_K\}, \\
\{\tilde{x}_1,\ldots,\tilde{x}_N\} &= \{x_1,\ldots,x_1,\ldots,x_K,\ldots,x_K\}, \\
\{\tilde{\xi}_1,\ldots,\tilde{\xi}_N\} &= \{\xi_{11},\ldots,\xi_{1n},\ldots,\xi_{K1},\ldots,\xi_{Kn}\}, \\
\{\tilde{\xi}_1^*,\ldots,\tilde{\xi}_N^*\} &= \{\xi_{11}^*,\ldots,\xi_{1n}^*,\ldots,\xi_{K1}^*\ldots,\xi_{Kn}^*\}.
\end{aligned}$$

Note that $(W\pi_j(y_i))^t x_i = w^t(\pi_j(y_i)\otimes x_i)$, where $\otimes$ is the Kronecker product. Thus the optimization problem above can be converted into

$$\begin{aligned}
\min_{w,b}\quad & \frac{1}{2}w^t w + C\sum_{i=1}^{N}(\tilde{\xi}_i^* + \tilde{\xi}_i) \\
\text{s.t.}\quad & \tilde{y}_i - w^t(\tilde{\pi}_i\otimes\tilde{x}_i) - \tilde{\pi}_i^t b \leq \varepsilon + \tilde{\xi}_i, \\
& w^t(\tilde{\pi}_i\otimes\tilde{x}_i) + \tilde{\pi}_i^t b - \tilde{y}_i \leq \varepsilon + \tilde{\xi}_i^*, \\
& \tilde{\xi}_i \geq 0, \quad \tilde{\xi}_i^* \geq 0, \quad i=1,\ldots,N.
\end{aligned} \tag{3}$$

This is a strictly convex optimization problem. Similar to SVR [9], we obtain its dual problem as the following:

$$\begin{aligned}
\min_{\{\alpha_i,\alpha_i^*\}_{i=1}^N}\quad & \frac{1}{2}\sum_{i,j=1}^{N}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j*)(\tilde{\pi}_i^t\tilde{\pi}_j)(\tilde{x}_i^t\tilde{x}_j) \\
& + \sum_{i=1}^{N}\alpha_i(\varepsilon-\tilde{y}_i) + \sum_{i=1}^{N}\alpha_i^*(\varepsilon+\tilde{y}_i) \\
\text{s.t.}\quad & 0 \leq \alpha_i, \alpha_i^* \leq C, \quad i=1,\ldots,N, \\
& \sum_{i=1}^{N}(\alpha_i-\alpha_i^*)\tilde{\pi}_i = 0,
\end{aligned} \tag{4}$$

This is a standard Quadratical Programming problem and can be efficiently solved. Here, the definition of support vectors is slightly different from SVR. A sample x_i is called a support vector if and only if

$$\beta_i = \sum_{\tilde{x}_k=x_i}(\alpha_k-\alpha_k^*)\tilde{\pi}_k \neq 0,$$

where $\{\tilde{\pi}_k\}_{\tilde{x}_k=x_i}$ corresponds to the coordinate system $\pi(y_i)$ at y_i. So there can be at most K support vectors. Suppose now we have a new input vector x, then the corresponding prediction $\hat{y}=[\hat{f}_1(x),\ldots,\hat{f}_n(x)]^t$ is computed by

$$\begin{aligned}
\hat{f}_i(x) &= \sum_{j=1}^{N}(\alpha_j-\alpha_j^*)(e_i^t\tilde{\pi}_j)(\tilde{x}_j^t x) + e_i^t b \\
&= \sum_{j=1}^{K}(e_i^t\beta_j)(x_j^t x) + e_i^t b.
\end{aligned} \tag{5}$$

It can be seen that LLT-SVR is a natural extension of the single-output SVR. The coefficient vectors $\{\beta_i\}_{i=1}^K$ take the role of the coefficients in SVR. When the number of outputs is one, i.e., $n=1$, LLT-SVR reduces to SVR. To handle the nonlinear case, it just needs to replace the terms like $x_i^t x_j$ (and $\tilde{x}_i^t\tilde{x}_j$) with values of a predefined kernel function [9,12]. Algorithm 1 gives the detailed procedure of LLT-SVR.

Please cite this article as: G. Liu, et al., Multi-output regression on the output manifold, Pattern Recognition (2009), doi:10.1016/j.patcog.2009.05.001

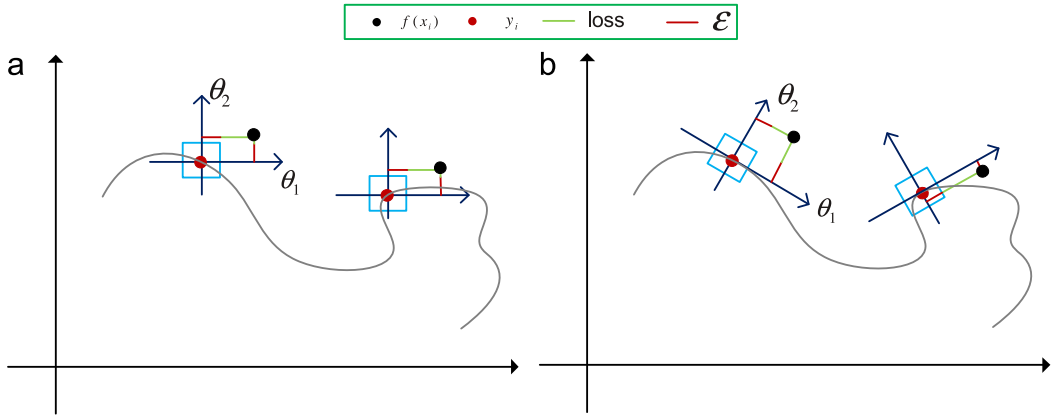

Fig. 2. An illustration of computing ε-insensitive loss function on manifolds. (a) Naïve, (b) LLT.

Algorithm 1. LLT-SVR: locally linear transformation based support vector regression.

Inputs: $\{(x_i, y_i)\}_{i=1}^K$ a collection of training points, $x_i \in \mathcal{M}_x \subseteq R^m$ and $y_i \in \mathcal{M}_y \subseteq R^n$.
Parameters: κ, the number of nearest neighbors for fitting the local shape of output manifold; C; ε; and other parameters required by SVR.
Procedure:
- *Identify neighbors.* For each output point y_i, $i = 1, \ldots, K$, identify the indices of its κ-nearest neighbors in Euclidean distance. Let $\{\mathcal{N}_i\}_{i=1}^K$ denote the collection of such neighborhoods. For each neighborhood $\mathcal{N}_i$, form an $n \times \kappa$ matrix M_i whose columns are the re-centered points $y_j - \bar{y}_i$, $j \in \mathcal{N}_i$, where $\bar{y}_i = (\sum_{j \in \mathcal{N}_i} y_j)/|\mathcal{N}_i|$.
- *Obtain local coordinate systems.* Perform singular value decomposition (SVD) of M_i: $M_i = U_i \Lambda_i V_i^t$. The columns of the $n \times n$ matrix U_i give the coordinate system $\pi(y_i)$ at the point y_i.
- *Train and predict.* Train the regression model by solving the convex quadratic programming problem defined in Eq. (4). Then use Eq. (5) to compute predictions for new inputs.

Given a dataset with n training examples and K outputs, LLT-SVR needs to solve a quadratic programming problem with a scale $n \cdot K$. So the computation complexity is $O(n^3 \cdot K^3)$, which is higher than the $O(n^3 \cdot K)$ of Naïve approach. This is expensive and we plan to develop more efficient algorithms by following the notions of online learning algorithms for SVM.

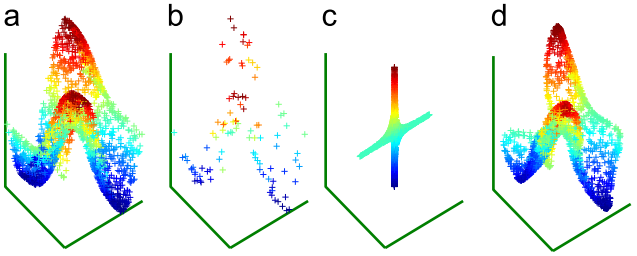

Fig. 3. A specific example of recovering the Twin Peaks manifold from its three-dimensional embedding. (a) The true manifold. (b) The distribution of training samples. (c) The manifold estimated by Naïve-SVR. (d) The manifold estimated by LLT-SVR.

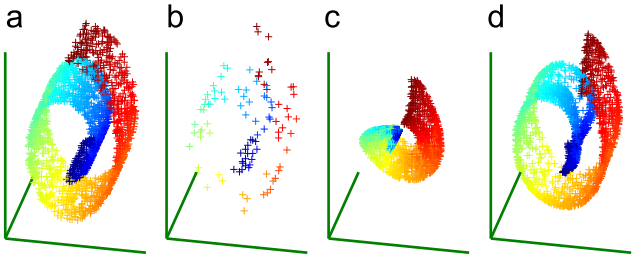

Fig. 4. A specific example of recovering Swill Roll from its three-dimensional embedding. (a) The true manifold. (b) The distribution of training samples. (c) The manifold estimated by Naïve-SVR. (d) The manifold estimated by LLT-SVR.

4. Experiments

4.1. Recovering the output manifolds

To testify the effectiveness of LLT, we define two regression tasks, which are to recover the Twin Peaks manifold and Swiss Roll manifold from their same-dimensional embedding computed by using LLE [13]. Or be more precisely, the three-dimensional embedded vectors are used as inputs, and the original three-dimensional vectors are considered as outputs. In this case, as shown in Figs. 3(a) and 4(a), the intrinsic dimension of the output manifold is smaller than the number of outputs. Note that here the embedded vectors computed by LLE are different from the original vectors even though their dimensions are the same. So the mapping defined here is *not* identity.

In the Twin Peaks recovering experiment, we randomly select 100 samples without noise added. In the Swiss Roll recovering experiment, we randomly select 100 samples and add Gaussian white noise ($\sigma = 0.1$) to them to train the regression models. For the Naïve approach, the Cartesian coordinate system of the ambient Euclidean space is used to define the loss function at each output training point. For LLT, the coordinate system at an output training point is estimated from its six nearest neighbors (i.e., $\kappa = 6$ in Algorithm 1). Both methods have the same setting on the rest parameters: the kernel function for SVR is chosen as Gaussian with a standard deviation being the average distance from each sample to its fifth nearest neighbor, $\varepsilon = 0.1$, and $C = 10$.

To see the effectiveness of LLT, we repeat the above random trials 10 times and compare LLT-SVR with the Naïve-SVR in each trial. The ℓ_2-error (or root of squared error) and ℓ_1-error (or absolute error) of 1900 test samples are considered to evaluate the performance of

Please cite this article as: G. Liu, et al., Multi-output regression on the output manifold, Pattern Recognition (2009), doi:10.1016/j.patcog.2009.05.001

Table 1
Evaluation results on *Swiss Roll* and *Twin Peaks*.

# of random trial	Swiss Roll				Twin Peaks			
	ℓ_2-error		ℓ_1-error		ℓ_2-error		ℓ_1-error	
	Naïve	LLT	Naïve	LLT	Naïve	LLT	Naïve	LLT
#1	0.802	0.7937	0.584	0.583	0.150	0.146	0.461	0.447
#2	0.675	0.666	0.543	0.535	0.157	0.163	0.472	0.476
#3	**4.307**	**0.595**	**1.414**	**0.507**	**0.874**	**0.153**	**1.118**	**0.457**
#4	**2.720**	**0.613**	**1.141**	**0.519**	0.160	0.157	0.483	0.468
#5	0.589	0.577	0.512	0.507	0.173	0.172	0.494	0.482
#6	0.559	0.547	0.497	0.498	0.163	0.162	0.477	0.471
#7	0.563	0.593	0.506	0.514	0.158	0.157	0.477	0.465
#8	0.581	0.569	0.504	0.495	0.153	0.148	0.468	0.458
#9	0.668	0.674	0.536	0.530	**0.810**	**0.147**	**1.091**	**0.452**
#10	0.673	0.658	0.557	0.551	0.142	0.147	0.451	0.450
Average	1.213	0.629	0.680	0.524	0.293	0.155	0.598	0.463
Std.	1.272	0.071	0.324	0.026	0.290	0.009	0.268	0.012

The performance under 100 random trail is considered (we only show the details of 10 trail). In these experiments, the Naïve-SVR and LLT-SVR have the same parameter setting except the definition of the loss function. The results with bold texts correspond to the cases that Naïve-SVR fail to recover the output manifold.

Table 2
Comparing LLT-SVR to Naïve-SVR and five multi-output regression methods quoted from [5].

Methods	LLR	MNW	MLLR	DNW	DLLR	Naïve-SVR	LLT-SVR
Average	3.4082	2.8516	2.5800	3.1935	3.4123	2.2480	**1.7513**
Std.	1.3926	0.8167	0.8202	0.8996	1.3622	1.3614	**0.7375**

different approaches:

$$\ell_2\text{-error} = \frac{1}{|T|}\sum_{i=1}^{|T|}\|y_i - f(x_i)\|_{\ell_2},$$

$$\ell_1\text{-error} = \frac{1}{|T|}\sum_{i=1}^{|T|}\|y_i - f(x_i)\|_{\ell_1},$$

where $T = \{(x_i, y_i)\}_{i=1}^{|T|}$ is a test set. Provided with these limited (100 samples) training examples, as shown in Table 1, LLT-SVR consistently outperforms the Naïve-SVR. One interesting phenomenon is that the Naïve-SVR sometimes (see the bold results in Table 1) fails to recover the output manifold. Figs. 3 and 4 show such two examples. The reason is that the training samples may not be *very* uniform on the manifold although we expect to select uniformly distributed training samples by random sampling. This phenomenon illustrates that LLT can help regression approaches to handle the problem of biased sampling of training data. Another thing that is worth noting is that the evaluation metric ℓ_1-error is chosen in "favor" of Naïve-SVR, as its loss function is defined with the same choice of coordinate system as Naïve-SVR. However, one can see that LLT-SVR performs better on the test data even though the evaluation metric is not the loss function that it uses during the training phase. This is natural because smaller training error may *not* produce smaller test error.

4.2. Robustness to correlated noises

The necessity of considering the correlation among different outputs is also supported by the regression model [5]:

$$y = f(x) + \varepsilon(x),$$

where $\varepsilon(x)$ is random noise such that the covariance matrix $Cov(\varepsilon(x))$ is not necessarily diagonal (i.e., correlated noises). To investigate the effectiveness of LLT in such cases, we consider a synthetic dataset that has been used by [5]. The data are generated according to the following regression functions:

$$f_1(x) = 0.25[1.5(1 - x_1) + e^{2x_1 - 1}\sin(3\pi(x_1 - 0.6)^2) + e^{3(x_2 - 0.5)}\sin(4\pi(x_2 - 0.9)^2)],$$

$$f_2(x) = \sin(2\pi x_1) + 4(x_2 - 0.5)^2$$

and the isotropic covariance matrix is defined in [5]. One hundred fixed samples are generated with the inputs being drawn from a uniform distribution on the square $[0,1] \times [0,1]$. Then we randomly select 36 samples for training and the rest for testing. For Naïve-SVR and LLT-SVR, we simply choose the linear kernel and set $\varepsilon = 0.1$, $C = 1$ and $\kappa = 6$ by 2-fold cross-validation. Table 2 shows the average for the 100 trail of the sum of the squared error obtained by LLT-SVR and its six competitors. It can be seen that on this dataset LLT-SVR works much better than its competitors.

4.3. Multi-label classification

Multi-label learning refers to problems where an instance can be assigned to multiple classes. This phenomenon is very common in practice. In text or music categorization, documents may belong to multiple genres, such as government and health, or rock and blues. Architecture may belong to multiple genres as well. In medical diagnosis, a disease may belong to multiple categories, and genes may have multiple functions. Multi-*label* learning differs from multi-*class* learning where every instance can only be assigned to one class even though the number of classes is more than two. Multi-label classification essentially needs to know the *exact* prediction functions $\{f_i(x)\}_{i=1}^n$ because x may be assigned to multiple classes. So we aim at learning the predication functions such that the class labels for a new input x are provided by

$$\mathscr{H}(x) = \bigcup\{i : f_i(x) > 0\}.$$

Such a problem is adequate to be solved by regression. Given K training examples $\{x_i\}_{i=1}^K$ and their corresponding labels, we create

Please cite this article as: G. Liu, et al., Multi-output regression on the output manifold, Pattern Recognition (2009), doi:10.1016/j.patcog.2009.05.001

Table 3
Statistics of the datasets used in the experiments.

Datasets	# train	# test	# class	# distinct labels	Dimension of inputs	Label density
Yeast	1500	917	14	198	103	0.302
Scene	1211	1196	6	15	294	0.179

Table 4
Results on *yeast* and *scene*.

	Yeast				Scene			
	Best in [15]	SVR			Best in [15]	SVR		
		Naïve	[4]	LLT		Naïve	[4]	LLT
Accuracy	0.5300	0.5082	0.4421	**0.5605**	0.7040	0.6537	0.6065	**0.7344**

In these experiments, the Naïve approach and LLT have the same parameter setting except the definition of the loss function. Although the approach of [4] also adopts the Naïve mechanism to define the loss function, it is different from the standard Naïve SVR as it uses a special kernel.

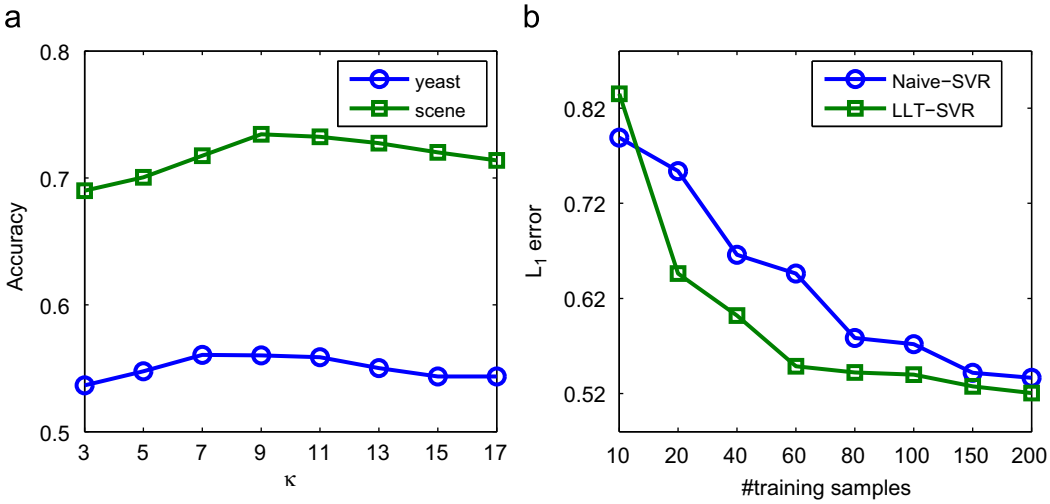

Fig. 5. (a) The influences of the parameter κ on *yeast* and *scene*. (b) The influences of the number of training samples on Swiss Roll (note there that we use the linear kernel to eliminate the influences of the kernel parameters). We plot the average error of 100 random trail.

K outputs $\{y_i\}_{i=1}^K$ with $y_i = [y_{1i}, \ldots, y_{ni}]^t$, where $y_{ji} = 1$ if x_i belongs to the j-th class and $y_{ji} = -1$ otherwise. With such training data, we can estimate the predication functions $\{f_i(x)\}_{i=1}^n$ using Algorithm 1.

We conduct experiments with two datasets, *yeast* and *scene*, from two different application domains: bioinformatics and semantic scene analysis, respectively. These datasets can be downloaded from the page of LibSVM [14]. Table 3 provides some information about these two datasets. We define *Accuracy* to evaluate the performance of different methods:

$$Accuracy = \frac{1}{|T|}\sum_{i=1}^{|T|}\frac{|H_i \cap Z_i|}{|H_i \cup Z_i|},$$

where T denotes a test dataset, and H_i and Z_i are the predicted and the ground truth labels of the data sample x_i, respectively. Besides the Naïve approach and LLT, we also consider the multi-output SVR algorithm proposed in [4] (see Introduction). For comparison, we also include the best results in a survey of multi-label classification [15]. Table 4 presents the evaluation results. It can be seen that the performance of LLT is promising.

One may have noticed that the outputs in these examples are discrete and they do *not* form a smooth manifold. Actually, the working of LLT does not depend on the assumption that the outputs form a smooth manifold. Due to Theorem 1, LLT can reduce the sum of standard deviation of the output variables, which can help improve prediction accuracy.

4.4. The influences of the parameters

There are several parameters in LLT-SVR. The distortion controller ε and the parameters of kernel function have the same properties as those of Naïve-SVR. They are set to be the same in our experiments. The parameter C is to balance the strengths of the loss and the regularization. There are $n{\cdot}K$ loss terms and K regularization terms in the optimization problem defined by Eq. (3). The ratio between them is the same as that in Naïve-SVR, which has n loss terms and one regularization term. So the parameter C is also set to be the same as that in Naïve-SVR. In our experiments, we first tune the parameters of Naïve-SVR and then assign them to LLT-SVR directly. The parameter κ depends on the smoothness of the manifold. Fig. 5(a) shows its influences. Generally, it cannot be too smaller as sufficient training samples are required to estimate the principal directions. And it is also not appropriate to use very large κ because this would make the locally linear assumption less accurate. According to our experience, the parameter κ should be around 7. Fig. 5(b) plots the ℓ_1 error as a function of the number of training samples. It can be seen that LLT-SVR does not break down even if there are very few training samples provided, in the case of which SVD should perform badly.

Please cite this article as: G. Liu, et al., Multi-output regression on the output manifold, Pattern Recognition (2009), doi:10.1016/j.patcog.2009.05.001

This is because LLT-SVR will reduce to Naïve-SVR when SVD fails to capture the principal directions of the output manifold.

5. Conclusion

In this paper, we propose a novel method, called locally linear transformation, to define loss functions for multi-output regression. LLT well models the underlying structure of the output manifold by using the orthonormal eigenvectors of the covariance matrix of local output samples as the local coordinate system. The loss functions are then defined in such local coordinate systems. And we prove that such local coordinate systems are the optimal choice that can produce the most stable prediction, if the stability is measured by the sum of standard deviation of the output vector in the chosen coordinate systems. Our experiments validate the effectiveness of LLT.

Appendix

Proof. Note that the chosen coordinate systems $\{\pi_1(y_i),\dots,\pi_n(y_i)\}$ are the eigenvectors of Σ_i. Let $\lambda_1,\dots,\lambda_n$ be the corresponding eigenvalues, which are non-negative. It can be computed that

$$SStd(\pi(y_i)) = \sum_{j=1}^{n} \sigma(\pi_j^t(y_i)y_i) = \sum_{j=1}^{n} \sqrt{\pi_j^t(y_i)\Sigma_i\pi_j(y_i)} = \sum_{j=1}^{n} \sqrt{\lambda_j}.$$

For any orthonormal coordinate system $\{\theta_1(y_i),\dots,\theta_n(y_i)\}$, as $\{\pi_1(y_i),\dots,\pi_n(y_i)\}$ span R^n, there exists an orthogonal matrix $A=[a_{ij}]_{n\times n}$ such that

$$\theta_j(y_i) = \sum_{k=1}^{n} a_{kj}\pi_k(y_i), \quad \forall j = 1,\dots,n.$$

Then it can be calculated that

$$SStd(\theta(y_i)) = \sum_{j=1}^{n} \sqrt{\theta_j^t(y_i)\Sigma_i\theta_j(y_i)} = \sum_{j=1}^{n} \sqrt{\sum_{k_1,k_2=1}^{n} a_{k_1 j}a_{k_2 j}\pi_{k_1}^t(y_i)\Sigma_i\pi_{k_2}(y_i)} = \sum_{j=1}^{n} \sqrt{\sum_{k=1}^{n}(a_{kj})^2\lambda_k}.$$

Since $\sqrt{x}$ $(x \geq 0)$ is a concave function, by Jensen's inequality we have that

$$\sqrt{\sum_{j=1}^{n}\alpha_j x_j} \geq \sum_{j=1}^{n}\alpha_j\sqrt{x_j} \quad \text{if } \sum_{j=1}^{n}\alpha_j = 1,$$

$$\alpha_j \geq 0, \quad x_j \geq 0, \ j = 1,\dots,n.$$

If x_j's are distinct, the above equality holds only when $\{\alpha_j\}_{j=1}^n = \{1,0,\dots,0\}$. Note that $\sum_{k=1}^n (a_{kj})^2 = \sum_{j=1}^n (a_{kj})^2 = 1$. Hence the above inequality applies:

$$SStd(\theta(y_i)) \geq \sum_{j=1}^{n}\sum_{k=1}^{n}(a_{kj})^2\sqrt{\lambda_k} = \sum_{k=1}^{n}\sqrt{\lambda_k} = SStd(\pi(y_i)).$$

And it is easy to see that if λ_j's are distinct, the quality holds only when $\theta(y_i) = \pi(y_i)$. □

References

[1] B. Taskar, S. Lacoste-Julien, M.I. Jordan, Structured prediction, dual extragradient and Bregman projections, J. Mach. Learn. Res. 7 (2006) 1627–1653.
[2] B. Taskar, B. Taskar, V. Chatalbashev, V. Chatalbashev, D. Koller, C. Guestrin, Learning structured prediction models: a large margin approach, in: ICML '05: Proceedings of the 22nd International Conference on Machine Learning, ACM, New York, NY, USA, 2005, pp. 896–903.
[3] S. Chen, Multi-output regression using a locally regularised orthogonal least-squares algorithm, IEE Proceedings—Vision, Image and Signal Processing 149 (4) (2002) 185–195.
[4] E. Vazquez, E. Walter, Multi output support vector regression, in: 13th IFAC Symposium on System Identification, SYSID 2003, 2003, pp. 1820–1825.
[5] J.M. Matias, Multi-output nonparametric regression, Progress in Artificial Intelligence (2005) 288–292.
[6] M. Belkin, P. Niyogi, V. Sindhwani, Manifold regularization: a geometric framework for learning from labeled and unlabeled examples, J. Mach. Learn. Res. 7 (2006) 2399–2434.
[7] S. Kobayashi, K. Nomizu, Foundations of Differential Geometry, 1963.
[8] G.H. Golub, C. Reinsch, Singular value decomposition and least squares solutions, Numerische Mathematik 14 (1970) 403–420.
[9] A.J. Smola, B. Schölkopf, A tutorial on support vector regression, Statistics and Computing 14 (3) (2004) 199–222.
[10] G.A.F. Seber, A.J. Lee, Linear regression analysis, Technometrics 45 (2003) 362–363.
[11] I.T. Jolliffe, Principal Component Analysis, 1986.
[12] J. Shawe-Taylor, N. Cristianini, Kernel Methods for Pattern Analysis, Cambridge University Press, New York, NY, USA, 2004.
[13] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.
[14] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, 2001, software available at ⟨http://www.csie.ntu.edu.tw/ cjlin/libsvm⟩.
[15] G. Tsoumakas, I. Katakis, Multi label classification: an overview, International Journal of Data Warehousing and Mining 3 (3) (2007) 1–13.

About the Author—GUANGCAN LIU is a Ph.D. Student of Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, PR China. He is a visiting student at Microsoft Research, Asia.

About the Author—ZHOUCHEN LIN received the Ph.D. degree in applied mathematics from Peking University, in 2000. He is currently a researcher in Visual Computing Group, Microsoft Research, Asia. His research interests include computer vision, computer graphics, pattern recognition, statistical learning, document processing, and human computer interaction. He is a senior member of the IEEE.

About the Author—YONG YU received his master degree at the Computer Science Department of East China Normal University. He began to work in Shanghai Jiao Tong University in 1986. Now he is the vice president of the Department of Computer Science and Engineering, Ph.D. candidate tutor and the chairman of E-generation technology research center (SJTU-IBM-HKU). He has received several prizes for his scholarship. He was the teacher of the course "Computer Graphics and Human–machine Interface" and the course "Next Generation Web Infrastructure" before. He has also got several prizes for his distinguished teaching career. As the head coach of SJTU ACM-ICPC team, he and his team have won the 2002 and 2005 ACM ICPC Championships and made SJTU the only world champion team of the contest in Asia.

Please cite this article as: G. Liu, et al., Multi-output regression on the output manifold, Pattern Recognition (2009), doi:10.1016/j.patcog.2009.05.001