Multiresolution dictionary learning for conditional distributions

Anonymous Author(s)
Affiliation
Address
email

1 Simulation studies

In order to assess the predictive performance of the proposed model, different simulation scenarios were considered. Let n be the number of observations, $y \in \Re$ the response variable and $x \in \Re^p$ a set of predictors. The Gibbs sampler was run considering $20,000$ as the maximum number of iterations with a burn-in of $1,000$. Gibbs sampler chains were stopped testing normality of normalized averages of functions of the Markov chain [**?**]. Parameters (a, b) and α involved in the prior density of parameters σ_{B_j}s and V_{B_j}s were set respectively equal to $(3, 1)$ and 1.

In all simulation scenarios, predictors were assumed to lie close a r-dimensional space, either a lower dimensional plane or a non linear manifold, with $r << p$. For each synthetic dataset, the proposed model was compared with CART and lasso in terms of mean squared error. Mean squared errors were computed based on leave-one=out predictions. For CART and Lasso standard Matlab packages were utilized and the regularization parameter of Lasso was chosen based on the AIC. The supplementary material includes more results about experiments involved in this section.

1.1 Illustrative Example

First let us consider the simple toy example of §**??**. We created an equally spaced grid of points $t_i = 0, \ldots, 20$. Then, we let $\eta_i = \sin(t_i)$ and predictors be a linear function of η_i plus Gaussian noise, i.e. $x_{ij} = \eta_i + \epsilon_{ij}$ with $\epsilon_{ij} \sim N(0, 0.1)$ for $j \in \{1, \ldots, p\}$. In particular, we set $p = 1,000$. The response was drawn from the following mixture of Gaussians

$$y_i \sim w_i \mathcal{N}(-2, 1) + (1 - w_i)\mathcal{N}(2, 1) \tag{1}$$

with $w_i = |\eta_i|$. Figure 1 shows the estimated density of two data points. These estimates were obtained by performing leave-one-out prediction for different number of observations in the training set. As the figure clearly shows our construction facilitates an estimate of the density y that become closer to the true density as the number of observations in the training set increases.

1.2 Linear lower dimensional space

In this section, predictors and response were assumed to lie close to a lower dimensional plane. In practice, we modeled $z_i = (y_i, x_i^t)^t$ through the following factor model

$$z_i = \Lambda \eta_i + \epsilon_i \tag{2}$$

with $\epsilon_i \sim \mathcal{N}(0, \Sigma_0)$, $\Sigma_0 = diag(\sigma_1, \ldots, \sigma_p)$, Λ being a $p \times r$ matrix, $\eta_i \sim \mathcal{N}(0, I)$ and $r << p$. The loading matrix was derived as the product of a matrix with orthogonal columns and a diagonal matrix with positive elements on the diagonal, i.e. $\Lambda = \Gamma\Theta$. In particular, the columns of Γ were uniformly sampled from the Stiefel manifold while the diagonal matrix of Θ were sampled from an inverse Gamma with shape and rate parameters $(1, 4)$. We set $r = 5$.

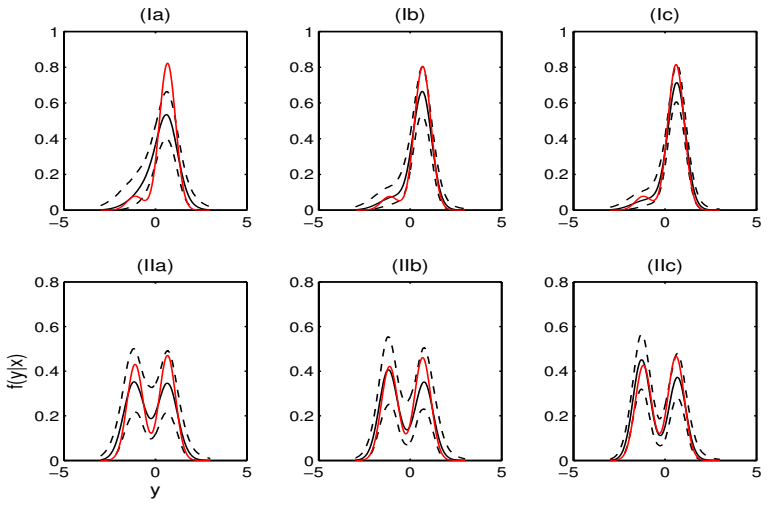

Figure 1: Illustrative example: Plot of true (red dashed-dotted line) and estimated (50th percentile: solid line, 2.5th and 97.5th percentiles: dashed lines) density for two data points (I, II) considering different training set size (a:100, b:150, c:200).

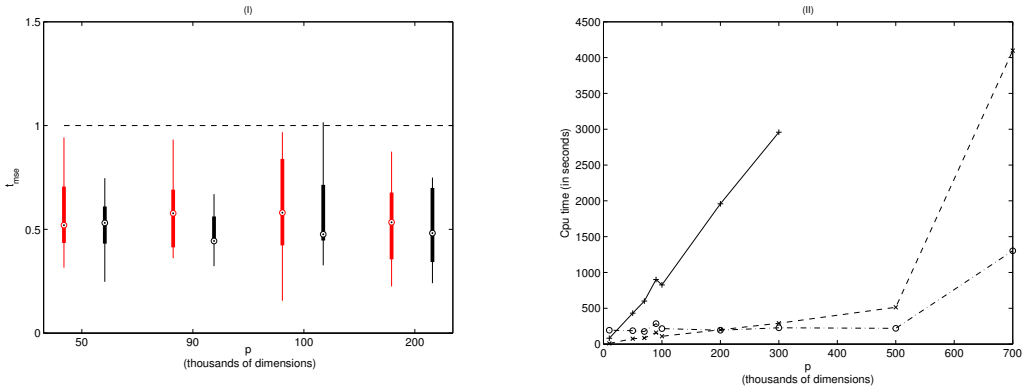

Figure 2: (I) Boxplots of t_{mse} as p increases. (II) Plot of Cpu time (in seconds) for Lasso (dash), Cart (solid) and MSB (dash-dot) under the first simulation scenario

As measure of comparison we adopted the ratio between cpu times and the ratio between mean squared errors. Define t_m^ℓ as

$$t_m^\ell = m(MSB)/m(\ell)$$

where m is either CPU time expressed in seconds or mean squared errors, MSB is our approach and ℓ is the competitor. For each simulation scenario, we sampled M datasets so that M values of t_m^ℓ were obtained. We sampled $M = 20$ datasets involving 100 observations and for each method we performed leave-one-out predictions. Figure **??**(I) shows boxplots of t_{mse}^ℓ as p increases. Clearly, our method outperforms the competitors in terms of mean squared errors. Furthermore, as shown in figure **??**(II), our approach can scale substantially better than competitors to huge dimensions of the predictor space.

1.3 Non-Linear lower dimensional space

In this section predictors were assumed to lie close to a lower dimensional non-linear manifold. In the first simulation study, predictors and response were jointly sampled from an N components mixture of factor analyzers. For each mixture components, the loading matrix and variances were sampled as in §1.2, while mixture weights were sampled from a $Dirichlet(1, \ldots, 1)$. The number

of latent factors was considered to be increasing in the number of components. In practice, we let the hth mixture component be modeled through h factors. We set $N = 5$ (the supplementary material shows results for different numbers of mixture components). In the other simulation scenario predictors were assumed to lie close to the Swissroll manifold (see figure 1 in the supplementary material), a two dimensional manifold embedded in $\Re^p$ while the response was sampled from a normal with mean equal to one coordinate of the manifold and standard deviation one.

For both data scenarios, we sampled $M = 20$ datasets involving 100 observations and we performed leave-one-out predictions. Figure 3(I) and 4 show boxplots of mean squared errors as p increases. Again our model is associated to better predictive performance compared to Cart and lasso. To show how the performance of our model varies for different sample sizes, we sampled datasets involving different number of observations. In practice, the dimension of the predictor space was considered fixed, i.e. $p = 300,000$ and ratios $t_{cpu}(\ell)$ were computed considering sample sizes $n \in \{100, 200, 300\}$. As shown in figure 3(II), the gap between our model and competitors improves as n increases.

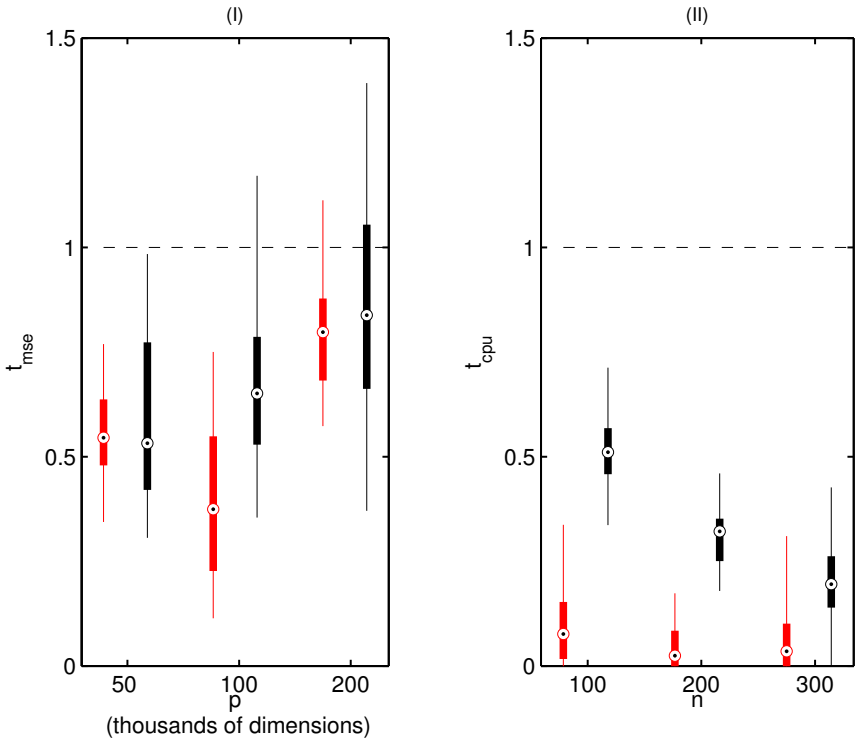

Figure 3: Boxplots of (I) t_{mse} as p increases and (II) t_{cpu} for a fixed $p = 300,000$ and different sample sizes under data drawn from a mixture of factor analyzers

2 Real application

We assessed the predictive performance of the proposed method on two very different neuroimaging datasets. First, we consider a structural connectome dataset collected at the Mind Research Network. Data were collected as described in Jung et al. [**?**]. For the analysis, all variables were normalized by subtracting the mean and dividing by the standard deviation. The same prior specification and Gibbs sampler as in §3 was utilized.

In the first experiment we investigated the extent to which we could predict creative (as measured via the Composite Creativity Index [**?**]). For each subject, we estimate a 70 vertex undirected weighted brain-graph using the Magnetic Resonance Connectome Automated Pipeline [**?**] from diffusion

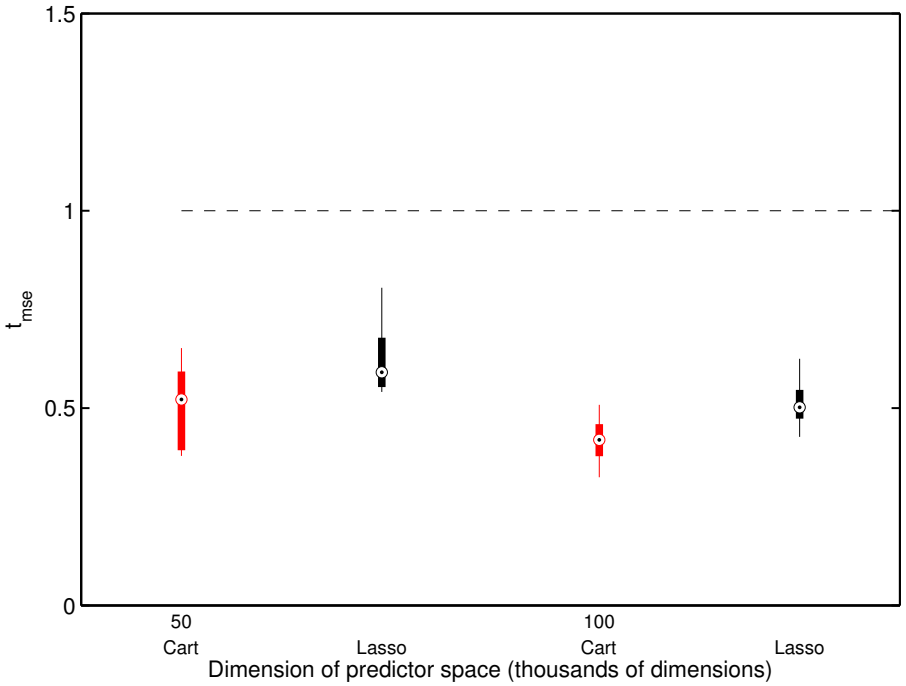

Figure 4: Boxplots of t_{mse} as p increases under the swiss roll simulation scenario

tensor imaging data [**?**]. Because our graphs are undirected and lack self-loops, we have a total of $\binom{70}{2} = 2,415$ potential weighted edges. The vector of covariates consists in the natural logarithm of the total number of connections between all pairs of cortical regions, i.e. $p = 2,415$.

The second dataset comes from a resting-state functional magnetic resonance experiment as part of the Autism Brain Imaging Data Exchange [**?**]. We selected the Yale Child Study Center for analysis. Each brain-image was processed using the Configurable Pipepline for Analysis of Connectomes [**?**]. For each subject we computed a measure of normalized power at each voxel called fALFF [**?**]. To ensure the existence of nonlinear signal relating these predictors, we let y_i correspond to an estimate of overall head motion in the scanner, called mean framewise displacement (FD) computed as described in Power et al. [**?**].

Table 1 shows mean and variance squared error based on leave-one-out predictions. Variable t_T is the amount of time necessary to obtain predictions for all subjects, while variables t_M and t_V are respectively the mean and the standard deviation of amount of time necessary to obtain one point predictions.

For the first data example, we compared our approach (multiresolution stick-breaking; MSB) to CART, lasso and random forests. Table 1 shows that MSB outperforms all the competitors in terms of mean square error; this is in addition to yielding an estimate of the entire conditional density for each y_i. It is also significantly faster that random forests, the next closest competitor, and faster than lasso. For this relatively low-dimensional example, CART is reasonably fast. For the second data application, given the huge dimensionality of the predictor space, we were unable to get either CART or random forest to run to completion, yielding memory faults on our workstation (Intel Core i7-2600K Quad-Core Processor memory 8192 MB). We thus only compare performance to lasso. As in the previous example, MSB outperforms lasso in terms of predictive accuracy measured via mean-squared error, and significantly outperforms lasso in terms of computational time.

Table 1: Real Data: Mean and standard deviations of squared error under multiscale stick-breaking (MSB), CART, Lasso and random forest (RF). Variable t_T is the amount of time necessary to obtain predictions for all subjects, while variables t_M and t_V are respectively the mean and the standard deviation of amount of time necessary to obtain one point predictions.

DATA	n	p	MODEL	MSE	t_T	t_M	t_V
(1)	108	2,415	MSB	0.56	100	1.1	0.02
			CART	1.10	87	0.9	0.01
			LASSO	0.63	50	0.40	0.10
			RF	0.57	$7,817$	78.2	0.59
(2)	56	$10e+05$	MSB	0.76	690	20.98	2.31
			LASSO	1.02	$5,836$	96.18	9.66