Sparse Additive Convex Regression

1 Introduction

We consider the following nonparametric regression problem

$$y_i = h(\boldsymbol{x}_i) + \epsilon_i = \sum_{d=1}^{p} h_d(x_{di}) + \epsilon_i \quad (i = 1, 2, \cdots, n)$$

where $\boldsymbol{x}_i \in \mathbb{R}^p$ is the covariate, y_i is the response and ϵ_i is the noise. The regression function $h(\cdot)$ is the summation of functions $h_d(\cdot)$ in each variable dimension. For high-dimensional data, it would be interesting to select the active variables, or equivalently, non-zero $h_d(\cdot)$ that contributes to $h(\cdot)$. Without further assumptions, this is the Sparse Additive Model (SPAM). Here we impose an additional constraint that each $h_d(\cdot)$ is a convex function, which can be expressed by its supporting function, i.e.,

$$h_{dj} \geq h_{di} + \beta_{di}(x_{dj} - x_{di}) \quad (\forall i, j)$$

where $h_{di} \triangleq h_d(x_{di})$ and β_{di} is the subgradient at point x_{di}. Hence we can formulate a convex program for the nonparametric regression problem:

$$\min_{\boldsymbol{h},\boldsymbol{\beta}} \ \frac{1}{2}\sum_{i=1}^{n}(y_i - \sum_{d=1}^{p} h_{di})^2 + \lambda\|\boldsymbol{\beta}\|_{\infty,1} \ \text{ s.t. } \sum_{i=1}^{n} h_{di} = 0 \ (\forall d), \quad h_{dj} \geq h_{di} + \beta_{di}(x_{dj} - x_{di}) \ (\forall i, j, d). \tag{1}$$

The regularization term $\|\boldsymbol{\beta}\|_{\infty,1}$ is for variable selection, and the equality constraint is an identifiability constraint. An ADMM algorithm is developed to solve this convex program. Mathematical details can be found in Section 2. Notice that if we fix $\beta_{di} = \beta_{d0}$ which is the same for all samples, then this additive convex regression reduces to linear regression.

The program (1) contains $O(n^2p)$ constraints, which is computationally expensive for large scale problems. Since for scalar convex functions, the subgradient should increase monotonically, we can reformulate (1) with only $O(np)$ constraints as

$$\min_{\boldsymbol{h},\boldsymbol{\beta}} \ \frac{1}{2}\sum_{i=1}^{n}(y_i - \sum_{d=1}^{p} h_{di})^2 + \lambda\|\boldsymbol{\beta}\|_{\infty,1} \ \text{ s.t. } \sum_{i=1}^{n} h_{di} = 0 \ (\forall d), \ \ h_{d(i+1)} = h_{d(i)} + \beta_{di}(x_{d(i+1)} - x_{d(i)}) \ (\forall i, d), \ \ \beta_{d,i+1} \geq \beta_{di} \ (\forall i). \tag{2}$$

Here $\{(1), (2), \cdots, (n)\}$ is a reordering of $\{1, 2, \cdots, n\}$ such that $x_{d(1)} \leq x_{d(2)} \leq \cdots \leq x_{d(n)}$, and β_{di} is the subgradient at point $x_{d(i)}$. It is easy to verify that the constraints in (2) implies the constraints in (1), as

$$\forall j \geq i, \quad h_{d(j)} - h_{d(i)} = \sum_{t=i}^{j-1}(h_{d(t+1)} - h_{d(t)}) = \sum_{t=i}^{j-1}\beta_{dt}(x_{d(t+1)} - x_{d(t)}) \geq \beta_{di}\sum_{t=i}^{j-1}(x_{d(t+1)} - x_{d(t)}) = \beta_{di}(x_{d(j)} - x_{d(i)})$$

$$\forall j < i, \quad h_{d(j)} - h_{d(i)} = \sum_{t=j}^{i-1}(h_{d(t)} - h_{d(t+1)}) = \sum_{t=j}^{i-1}\beta_{dt}(x_{d(t)} - x_{d(t+1)}) \geq \beta_{di}\sum_{t=j}^{i-1}(x_{d(t)} - x_{d(t+1)}) = \beta_{di}(x_{d(j)} - x_{d(i)}),$$

hence $h_{d(j)} \geq h_{d(i)} + \beta_{di}(x_{d(j)} - x_{d(i)})$ $(\forall j)$. The ADMM algorithm for the convex program (2) can be found in Section 3.

2 ADMM for (1)

The convex program in (1) is equivalent to

$$\min \ \frac{1}{2}\sum_{i=1}^{n}(y_i - \sum_{d=1}^{p} h_{di})^2 + \lambda\|\boldsymbol{C}\|_{\infty,1} \ \text{ s.t. } \boldsymbol{C} = \boldsymbol{\beta}, \sum_{i=1}^{n} h_{di} = 0, \ \ h_{dj} = h_{di} + \beta_{di}(x_{dj} - x_{di}) + S_{dji}, \ \ S_{dji} \geq 0.$$

Then the ADMM objective function can be expressed as

$$\min \frac{1}{2}\sum_{i=1}^{n}(y_i - \sum_{d=1}^{p} h_{di})^2 + \lambda\|\boldsymbol{C}\|_{\infty,1} + \frac{\mu}{2}\|\boldsymbol{C} - \boldsymbol{\beta} + \mu^{-1}\boldsymbol{O}\|_F^2 + \sum_{d=1}^{p}\frac{\mu}{2}\|\sum_{i=1}^{n} h_{di} + \mu^{-1}q_d\|^2$$
$$+ \sum_{d=1}^{p}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\mu}{2}\|h_{dj} - (h_{di} + \beta_{di}(x_{dj} - x_{di}) + S_{dji}) + \mu^{-1}W_{dji}\|^2 \quad \text{s.t. } S_{di} \geq 0 \ (\forall i, j, d)$$

where $\boldsymbol{O}, \boldsymbol{q}, \boldsymbol{W}$ are the Lagrange multipliers and μ is the ADMM parameter. The ADMM update equations are derived as follows.

1. Update $\boldsymbol{C}$:
$$\boldsymbol{C} = \arg\min_{\boldsymbol{C}} \lambda\|\boldsymbol{C}\|_{\infty,1} + \frac{\mu}{2}\|\boldsymbol{C} - \boldsymbol{\beta} + \mu^{-1}\boldsymbol{O}\|_F^2$$
which has analytical solution.

2. Update $\boldsymbol{h}$:
$$\boldsymbol{h} = \Big(\mu^{-1}\sum_i(\sum_d \boldsymbol{e}_{di})(\sum_d \boldsymbol{e}_{di})^\top + \sum_d(\sum_i \boldsymbol{e}_{di})(\sum_i \boldsymbol{e}_{di})^\top + \sum_d\sum_i\sum_j(\boldsymbol{e}_{dj} - \boldsymbol{e}_{di})(\boldsymbol{e}_{dj} - \boldsymbol{e}_{di})^\top\Big)^{-1}$$
$$\Big(\mu^{-1}\sum_i(\sum_d \boldsymbol{e}_{di})y_i - \sum_d(\sum_i \boldsymbol{e}_{di})\mu^{-1}q_d + \sum_d\sum_i\sum_j(\boldsymbol{e}_{dj} - \boldsymbol{e}_{di})(\beta_{di}(x_{dj} - x_{di}) + S_{dji} - \mu^{-1}W_{dji})\Big)$$
where $\boldsymbol{h}$ is a np-dimensional vector, and $\boldsymbol{e}_{di}$ is an unit vector with all zeros except a one at the subscript position. Matrix inversion lemma can be used to avoid direct matrix inversion.

3. Update $\boldsymbol{\beta}$:
$$\beta_{di} = \Big(1 + \sum_j(x_{dj} - x_{di})^2\Big)^{-1}\Big(C_{di} + \mu^{-1}M_{di} + \sum_j((x_{dj} - x_{di}))(h_{dj} - h_{di} - S_{dji} + \mu^{-1}W_{dji})\Big)$$

4. Update $\boldsymbol{S}$:
$$S_{dji} = \max(h_{dj} - (h_{di} + \beta_{di}(x_{dj} - x_{di})) + \mu^{-1}W_{dji}, 0).$$

5. Update the Lagrange multipliers:
$$\boldsymbol{O} = \boldsymbol{O} + \mu\cdot(\boldsymbol{C} - \boldsymbol{\beta}), \quad q_d = q_d + \mu\cdot\sum_i h_{di}, \quad W_{dji} = W_{dji} + \mu\cdot(h_{dj} - (h_{di} + \beta_{di}(x_{dj} - x_{di}) + S_{dji})).$$

3 ADMM for (2)

The convex program in (2) is equivalent to

$$\min \frac{1}{2}\sum_{i=1}^{n}(y_i - \sum_{d=1}^{p} h_{di})^2 + \lambda\|\boldsymbol{C}\|_{\infty,1} \text{ s.t. } \boldsymbol{C} = \boldsymbol{\beta}, \sum_{i=1}^{n} h_{di} = 0, \ h_{d(i+1)} = h_{d(i)} + \beta_{di}(x_{d(i+1)} - x_{d(i)}), \ \beta_{d,i+1} = \beta_{di} + S_{di}, S_{di} \geq 0.$$

Then the ADMM objective function can be expressed as

$$\min \frac{1}{2}\sum_{i=1}^{n}(y_i - \sum_{d=1}^{p} h_{di})^2 + \lambda\|\boldsymbol{C}\|_{\infty,1} + \frac{\mu}{2}\|\boldsymbol{C} - \boldsymbol{\beta} + \mu^{-1}\boldsymbol{O}\|_F^2 + \sum_{d=1}^{p}\frac{\mu}{2}\|\sum_{i=1}^{n} h_{di} + \mu^{-1}q_d\|^2$$
$$+ \sum_{d=1}^{p}\sum_{i=1}^{n-1}\frac{\mu}{2}\|h_{d(i+1)} - (h_{d(i)} + \beta_{di}(x_{d(i+1)} - x_{d(i)})) + \mu^{-1}M_{di}\|^2 + \sum_{d=1}^{p}\sum_{i=1}^{n-2}\frac{\mu}{2}\|\beta_{d,i+1} - (\beta_{di} + S_{di}) + \mu^{-1}W_{di}\|^2 \text{ s.t. } S_{di} \geq 0$$

where $\boldsymbol{O}, \boldsymbol{q}, \boldsymbol{W}, \boldsymbol{M}$ are the Lagrange multipliers and μ is the ADMM parameter. The ADMM update equations are derived as follows.

1. Update $\boldsymbol{C}$:

$$\boldsymbol{C} = \arg\min_{\boldsymbol{C}} \lambda\|\boldsymbol{C}\|_{\infty,1} + \frac{\mu}{2}\|\boldsymbol{C} - \boldsymbol{\beta} + \mu^{-1}\boldsymbol{O}\|_F^2$$

which has analytical solution.

2. Update $\boldsymbol{h}$:

$$\boldsymbol{h} = \Big(\mu^{-1}\sum_i(\sum_d \boldsymbol{e}_{di})(\sum_d \boldsymbol{e}_{di})^\top + \sum_d(\sum_i \boldsymbol{e}_{di})(\sum_i \boldsymbol{e}_{di})^\top + \sum_d\sum_i(\boldsymbol{e}_{d(i+1)} - \boldsymbol{e}_{d(i)})(\boldsymbol{e}_{d(i+1)} - \boldsymbol{e}_{d(i)})^\top\Big)^{-1}$$
$$\Big(\mu^{-1}\sum_i(\sum_d \boldsymbol{e}_{di})y_i - \sum_d(\sum_i \boldsymbol{e}_{di})\mu^{-1}q_d + \sum_d\sum_i(\boldsymbol{e}_{d(i+1)} - \boldsymbol{e}_{d(i)})(\beta_{di}(x_{d(i+1)} - x_{d(i)}) - \mu^{-1}M_{di})\Big)$$

Here $\boldsymbol{h} = (\boldsymbol{h}_1^\top, \boldsymbol{h}_2^\top, \cdots, \boldsymbol{h}_p^\top)^\top$ is a np-dimensional vector, $\boldsymbol{h}_d \in \mathbb{R}^n$ is the response corresponding to the d^{th} feature dimension, and $\boldsymbol{e}_{d(i)}$ is an unit vector with all zeros except a one at the subscript position. Then

$$\mu^{-1}\sum_i(\sum_d \boldsymbol{e}_{di})(\sum_d \boldsymbol{e}_{di})^\top = \mu^{-1}\begin{pmatrix}\boldsymbol{I}_n\\ \boldsymbol{I}_n\\ \vdots\\ \boldsymbol{I}_n\end{pmatrix}(\boldsymbol{I}_n, \boldsymbol{I}_n, \cdots, \boldsymbol{I}_n); \quad \sum_d(\sum_i \boldsymbol{e}_{di})(\sum_i \boldsymbol{e}_{di})^\top = \begin{pmatrix}\boldsymbol{1}\boldsymbol{1}^\top & & & \\ & \boldsymbol{1}\boldsymbol{1}^\top & & \\ & & \cdots & \\ & & & \boldsymbol{1}\boldsymbol{1}^\top\end{pmatrix}$$

and

$$\sum_d\sum_i(\boldsymbol{e}_{d(i+1)} - \boldsymbol{e}_{d(i)})(\boldsymbol{e}_{d(i+1)} - \boldsymbol{e}_{d(i)})^\top = \begin{pmatrix}\boldsymbol{\Pi}_1\boldsymbol{\Sigma}\boldsymbol{\Pi}_1^\top & & & \\ & \boldsymbol{\Pi}_2\boldsymbol{\Sigma}\boldsymbol{\Pi}_2^\top & & \\ & & \cdots & \\ & & & \boldsymbol{\Pi}_p\boldsymbol{\Sigma}\boldsymbol{\Pi}_p^\top\end{pmatrix}$$

where $\boldsymbol{\Pi}_d = (\boldsymbol{e}_{d(1)}, \boldsymbol{e}_{d(2)}, \cdots, \boldsymbol{e}_{d(n)})$ is a permutation matrix, and

$$\boldsymbol{\Sigma} = \sum_i(\boldsymbol{e}_{d,i+1} - \boldsymbol{e}_{di})(\boldsymbol{e}_{d,i+1} - \boldsymbol{e}_{di})^\top = \begin{pmatrix}1 & -1 & & \\ -1 & 2 & -1 & \\ & & \cdots & \\ & & -1 & 1\end{pmatrix}.$$

Hence the matrix inversion for computing $\boldsymbol{h}$ can be expressed as

$$\left(\mu^{-1}\begin{pmatrix}\boldsymbol{I}_n\\ \boldsymbol{I}_n\\ \vdots\\ \boldsymbol{I}_n\end{pmatrix}(\boldsymbol{I}_n, \boldsymbol{I}_n, \cdots, \boldsymbol{I}_n) + \begin{pmatrix}\boldsymbol{\Pi}_1\boldsymbol{\Sigma}\boldsymbol{\Pi}_1^\top + \boldsymbol{1}\boldsymbol{1}^\top & & & \\ & \boldsymbol{\Pi}_2\boldsymbol{\Sigma}\boldsymbol{\Pi}_2^\top + \boldsymbol{1}\boldsymbol{1}^\top & & \\ & & \cdots & \\ & & & \boldsymbol{\Pi}_p\boldsymbol{\Sigma}\boldsymbol{\Pi}_p^\top + \boldsymbol{1}\boldsymbol{1}^\top\end{pmatrix}\right)^{-1}$$

Hence we can apply the matrix inversion lemma to speed up the computation.
Notice that $(\boldsymbol{\Pi}_d\boldsymbol{\Sigma}\boldsymbol{\Pi}_d^\top + \boldsymbol{1}\boldsymbol{1}^\top)^{-1} = \boldsymbol{\Pi}_d(\boldsymbol{\Sigma} + \boldsymbol{1}\boldsymbol{1}^\top)^{-1}\boldsymbol{\Pi}_d^\top$.

3. Update $\boldsymbol{\beta}$:

$$\boldsymbol{\beta}_{\boldsymbol{d}\cdot} = \Big(\boldsymbol{I} + \sum_i(x_{d(i+1)} - x_{d(i)})^2\boldsymbol{e}_{di}\boldsymbol{e}_{di}^\top + \sum_i(\boldsymbol{e}_{d,i+1} - \boldsymbol{e}_{d,i})(\boldsymbol{e}_{d,i+1} - \boldsymbol{e}_{d,i})^\top\Big)^{-1}$$
$$\Big(\boldsymbol{C}_{d\cdot} + \mu^{-1}\boldsymbol{O}_{d\cdot} + \sum_i \boldsymbol{e}_{di}(x_{d(i+1)} - x_{d(i)})\cdot(h_{d(i+1)} - h_{d(i)} + \mu^{-1}M_{di}) + \sum_i(\boldsymbol{e}_{d,i+1} - \boldsymbol{e}_{d,i})(S_{di} - \mu^{-1}W_{di})\Big)$$

4. Update $\boldsymbol{S}$:

$$S_{di} = \max(\beta_{d,i+1} - \beta_{di} + \mu^{-1}W_{di}, 0).$$

5. Update the Lagrange multipliers:

$$\boldsymbol{O} = \boldsymbol{O} + \mu\cdot(\boldsymbol{C} - \boldsymbol{\beta}), \quad q_d = q_d + \mu\cdot\sum_i h_{di},$$
$$M_{di} = M_{di} + \mu\cdot(h_{d(i+1)} - (h_{d(i)} + \beta_{di}(x_{d(i+1)} - x_{d(i)}))), \quad W_{di} = W_{di} + \mu\cdot(\beta_{d,i+1} - (\beta_{di} + S_{di})).$$

4 Initial Results

We apply the ADMM algorithm in Section 3 to the following simulated examples.

First, we consider a quadratic regression problem. The full feature dimension is 100 and the number of active features is 5. The function is $h(\boldsymbol{x}) = \boldsymbol{x}_{\text{active}}^{\top} \cdot \boldsymbol{Q} \cdot \boldsymbol{x}_{\text{active}}$. We tested our algorithm on both diagonal and non-diagonal $\boldsymbol{Q}$. For the non-diagonal cases, each off-diagonal element is set to be 0.5 with probability α, and the diagonal elements are adjusted such that $\boldsymbol{Q}$ is positive definite with condition number 5. We considered $\alpha = 0.2$, 0.5 and 1.0 (full matrix). The four different $\boldsymbol{Q}$ considered are plotted in Figure 1. We run the ADMM algorithm for each case, with the number of samples ranging from 100 to 1000. The probability of support recovery is computed from 100 independent runs. When determining whether $\|\boldsymbol{\beta}_{d\cdot}\|_{\infty}$ is zero or not, we used a threshold of 10^{-16}. The result is depicted in Figure 2. We fixed the regularization parameter $\lambda = 10\sqrt{n}$ in the above experiment. In order to see the impact of setting this parameter, we run the algorithm for the $\alpha = 1.0$ case using 500 data samples, with λ varied from $\sqrt{n}$ to $20\sqrt{n}$. The inferred L_{∞} norm of the regression coefficients for each feature dimension is plotted in Figure 3.

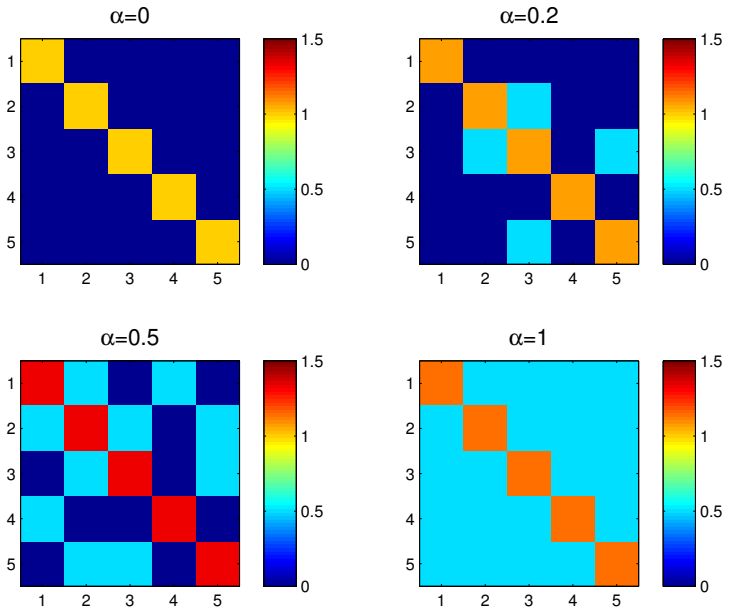

Figure 1: Different quadratic matrices considered.

Second, we use different convex function for different dimensions. The full feature dimension is 100, and the number of active features is 5. The convex functions in these features are $|x|$, $|x|^2$, $|x|^3$,$|x|^4$ and $exp(x)$. These functions are standardized according to the data. The data sample size is 500. The algorithm correctly identified the five active features. We then only use these active features to fit the model again without regularization. The true functions and the inferred functions are plotted in Figure 4.

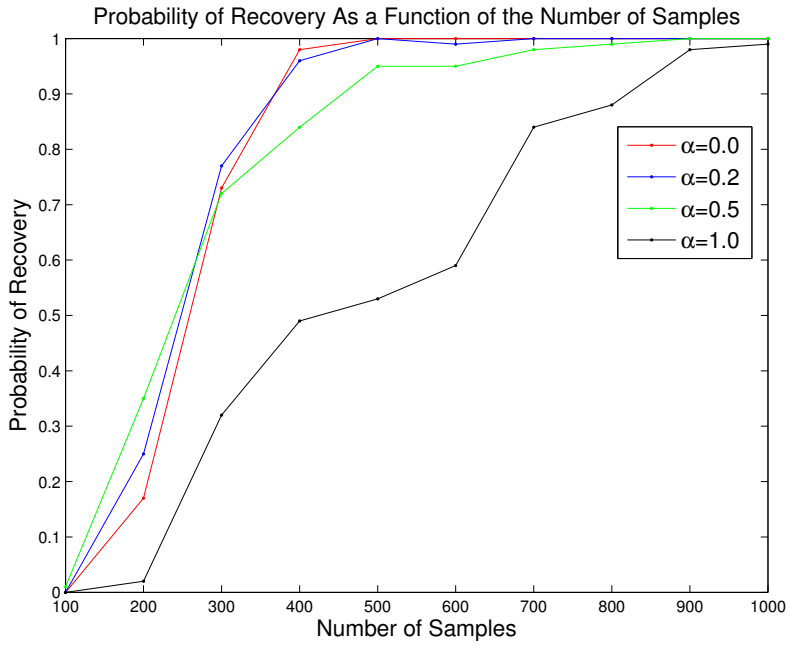

Figure 2: Support recovery result for quadratic functions.

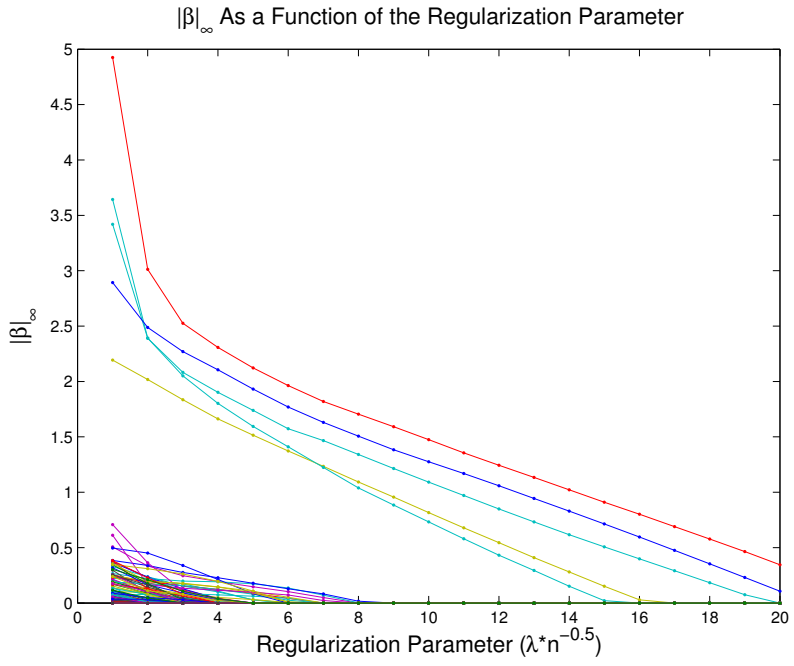

Figure 3: $\|\boldsymbol{\beta}\|_\infty$ as a function of λ.

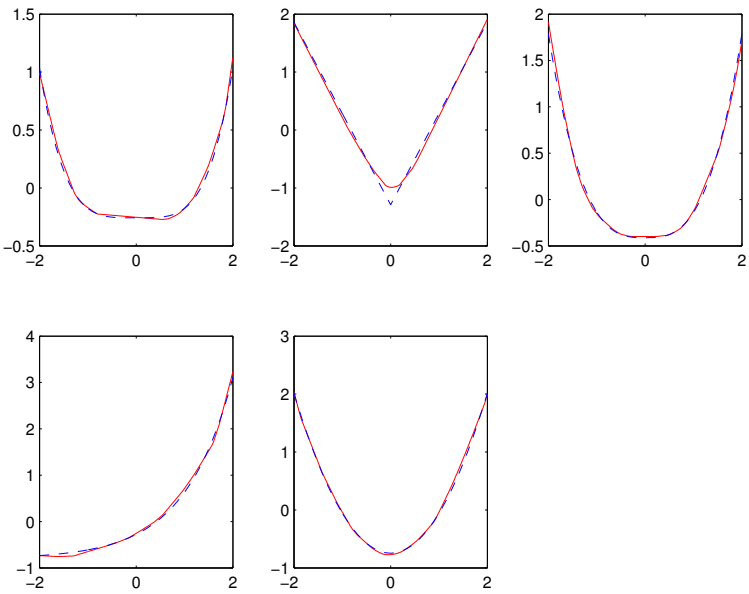

Figure 4: Inferred functions after refitting. Blue lines are the ground truth, and red lines are inferred.