Additive Models with Sparse Convexity Patterns

YJ Choe

The University of Chicago

yjchoe@uchicago.edu

August 21, 2014

This talk is based on an ongoing work in Professor John Lafferty's group, which includes Sabyasachi Chatterjee, YJ Choe (the presenter), Max Cytrynbaum, Wei Hu, Yuxue Qi, and Min Xu.

Overview

Section 1

Introduction

Regression

Suppose we have data $(X_1, Y_1), \ldots, (X_n, Y_n) \in \mathbb{R}^p \times \mathbb{R}$, where $X_i = (X_{i1}, \ldots, X_{ip})^T$ for each $i = 1, \ldots, n$.
We assume that this data comes from a true regression function m with a Gaussian noise $\varepsilon_i \overset{IID}{\sim} \mathcal{N}(0, \sigma^2)$:

$$Y_i = m(X_i) + \varepsilon_i$$

for $i = 1, \ldots, n$.

Least-Squares Fit

We are interested in finding the function that minimizes the **mean squared error (MSE)** within an assumed function space $\mathcal{F}$:

$$\hat{m} = \operatorname*{argmin}_{m \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (Y_i - m(X_i))^2 .$$

Nonparametric Regression

Data: $(X_1, Y_1), \ldots, (X_n, Y_n) \in \mathbb{R}^p \times \mathbb{R}$
Model: $Y_i = m(X_i) + \varepsilon_i$, where $\varepsilon_i \overset{IID}{\sim} \mathcal{N}(0, \sigma^2)$
Goal: Minimize the MSE on $\mathcal{F}$, i.e.

$$\hat{m} = \underset{m \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} (Y_i - m(X_i))^2$$

Nonparametric? *Weak* assumptions on $\mathcal{F}$, i.e. (much) larger $\mathcal{F}$:

- smooth functions
- convex functions

Additive Models

Suppose $p > 1$. We assume that the true regression function m is **additive**:

$$m(x) = \sum_{j=1}^{p} f_j(x_j)$$

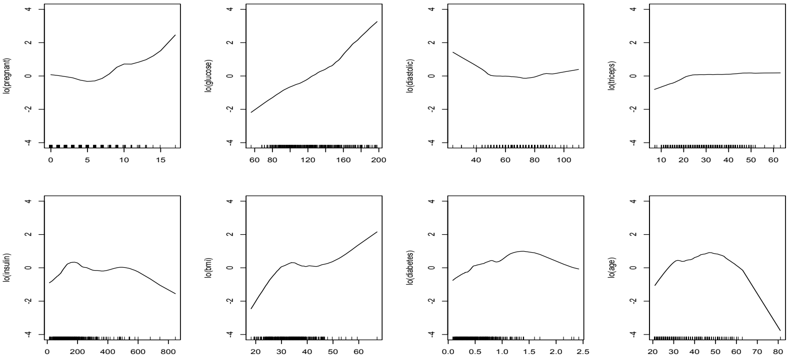
for any $x = (x_1, \ldots, x_p)^T \in \mathbb{R}^p$. We call each univariate function f_j **components** for $j = 1, \ldots, p$.

Why Additive Models?

- Nonparametric
- Tractable i.e. easier to fit
- Interpretable

...(sometimes) even when the true model is not additive!

Interpretability (Generalized Additive Model, Logistic)



(Data: `pima` from [Faraway, 2014] in `R` package `faraway`)

The Backfitting Algorithm

Algorithm 1 The Backfitting Algorithm

Given $\{(X_i, Y_i)\}_{i=1}^n \subseteq \mathbb{R}^p \times \mathbb{R}$, where $\sum_{i=1}^n Y_i = 0$
Initialize $\hat{f}_j \equiv 0$ for each $j = 1, \ldots, p$
repeat
  for $j = 1, \ldots, p$ (or in random order) **do**
    $R_i = Y_i - \sum_{k \neq j} \hat{f}_k(X_{ik})$ for $i = 1, \ldots, n$ # Residuals
    $\hat{f}_j = fit.1d(\{(X_{ij}, R_i)\}_{i=1}^n)$ # 1-D Regression on Residuals
    $\hat{f}_j = \hat{f}_j - mean(\{f_j(X_{ij})\}_{i=1}^n)$ # Mean Centering
  end for
until change in fitted values is small

Sparsity

With high-dimensional models, we usually hope that the fit is **sparse**, i.e. we want it to be "effectively" a lower-dimensional model which we can describe with only a few parameters/components.

Regularization

For now, assume the parametric linear regression model

$$Y = X\beta + \varepsilon.$$

Instead of the usual mean squared error, we minimize

$$\frac{1}{n}\|Y - X\beta\|_2^2 + \lambda J(\beta)$$

where $J(\beta)$ is a penalty term, which is a function of the coefficient vector β, and λ is some positive constant. This process is called **regularized least-squares**; the general technique of adding a penalty term to the objective is called **regularization**.

ℓ^1-Regularization a.k.a. the Lasso

In particular, if we have the ℓ^1-penalty $J(\beta) = \|\beta\|_1 = \sum_{j=1}^{p} |\beta_j|$, we call this the **Lasso** [Tibshirani, 1996]. The objective becomes

$$\frac{1}{n} \|Y - X\beta\|_2^2 + \lambda \|\beta\|_1$$

Note: The Lasso is a quadratic program (QP).

Lasso Induces Sparsity!

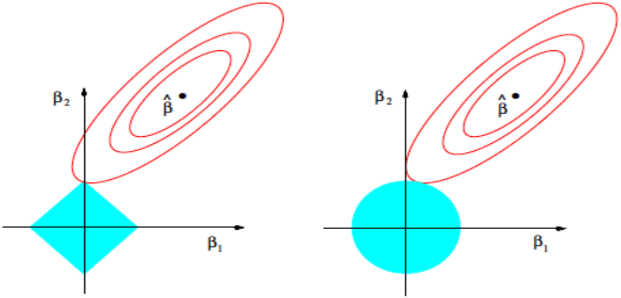


FIGURE 3.11. *Estimation picture for the lasso (left) and ridge regression (right). Shown are contours of the error and constraint functions. The solid blue areas are the constraint regions* $|\beta_1| + |\beta_2| \leq t$ *and* $\beta_1^2 + \beta_2^2 \leq t^2$*, respectively, while the red ellipses are the contours of the least squares error function.*

[Hastie et al., 2009]

Sparse Additive Models (SpAM) [Ravikumar et al., 2009]

In terms of additive models, this would mean that we want a majority of components to be identically zero.

Sparsity pattern: whether each component is sparse.

Shape Constraints

We assume that functions in our model have certain shapes, e.g.

- monotonicity
- convexity, log-convexity, and SOS-convexity

In general, models with “nice” shape constraints come with more tractable estimation techniques that still works for a variety of examples.

Convexity

A function f on a convex set $C \subseteq \mathbb{R}^p$ is **convex** if

$$f((1-\lambda)x_1 + \lambda x_2) \leq (1-\lambda)f(x_1) + \lambda f(x_2).$$

for all $x_1, x_2 \in C$ and $\lambda \in [0,1]$. f is **concave** if $-f$ is convex.

Convex/concave functions naturally appear in various cases. For example, a utility function with diminishing returns is concave [Qi, Xu, and Lafferty, to appear].

The Problem

Here, we attempt to combine additive models with shape constraints!

Specifically, we consider a regression model in which the true function is *additive* and each component is either *convex, concave, or identically zero*.

Convexity Pattern

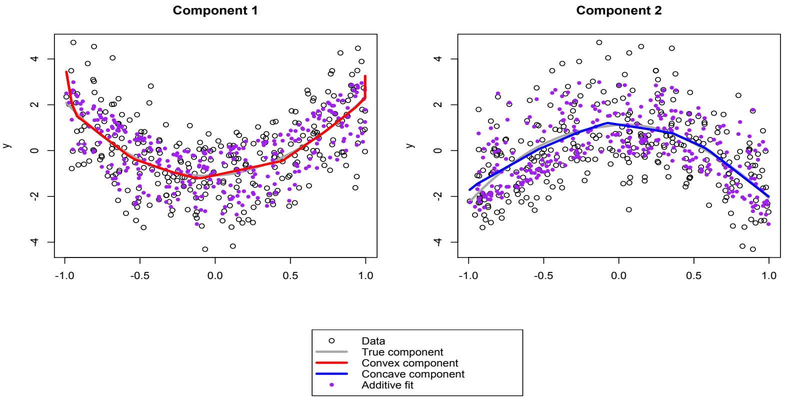
The model:

$$Y_i = \sum_{j=1}^{p} [f_j(X_{ij}) + g_j(X_{ij})] + \varepsilon_i$$

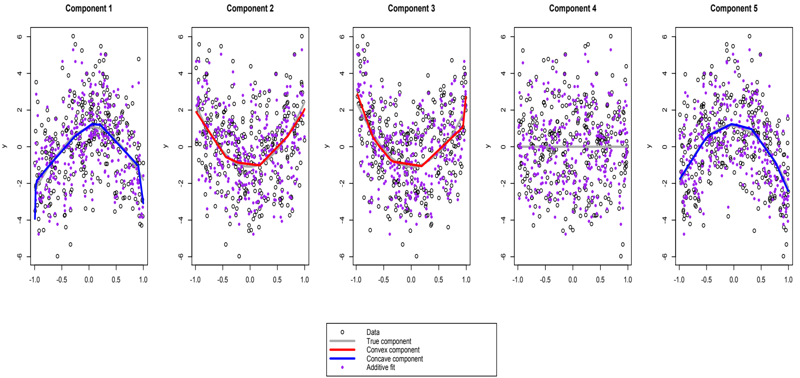
where, for each $j = 1, \ldots, p$, f_j is convex, g_j is concave, and *at most one* of f_j and g_j is nonzero.

That is, each component is either convex, concave, or identically zero. We call this ternary pattern a **sparse convexity pattern**, or simply, a **convexity pattern**.

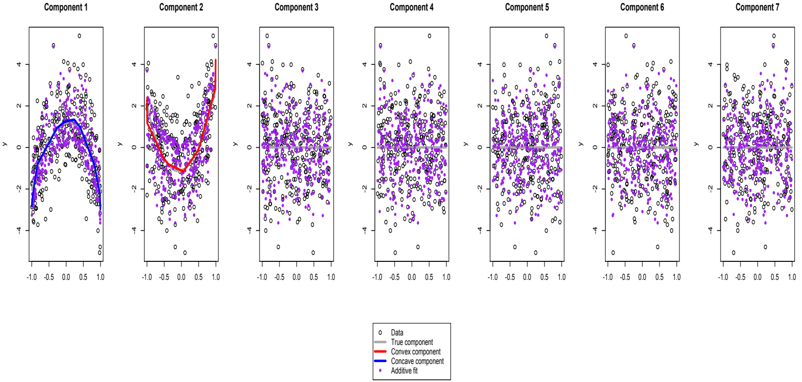
Example: 2 Components, 1 Convex & 1 Concave



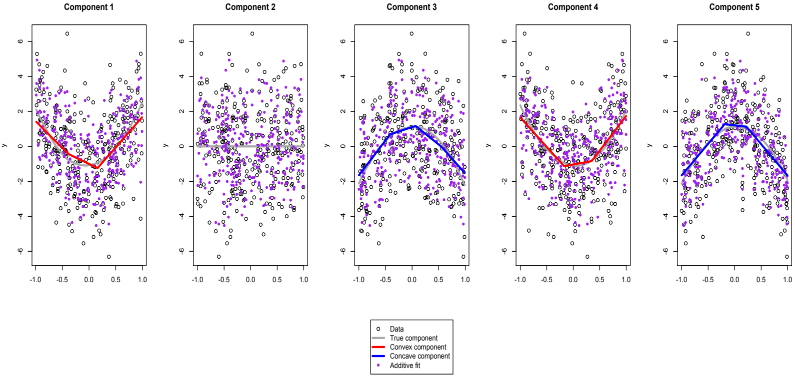
Example: 5 Components with 1 Sparse Component



Example: 7 Components with 5 Sparse Components



Example: 5 Components with 1 Sparse Component (2)



Section 2

MISOCP Formulation

Convex Regression

Suppose we have a p-variate regression problem in which the true function is assumed to be convex. This is:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{n}\sum_{i=1}^{n}(Y_i - m(X_i))^2 \\ \text{s.t.} \quad & m \text{ is convex.} \end{aligned}$$

Convex Regression as a QP

It can be shown that this problem is, in fact, equivalent to the following finite-dimensional quadratic program (QP):

$$\begin{aligned} \underset{f,\beta}{\text{minimize}} \quad & \frac{1}{n}\sum_{i=1}^{n}(Y_i - f_i)^2 \\ \text{s.t.} \quad & f_{i'} \geq f_i + \beta_i^T (X_{i'} - X_i) \\ & i, i' = 1, \ldots, n. \end{aligned}$$

Here, $f = (f_1, \ldots, f_n)^T$ is a vector of fitted values and $\beta_1, \ldots, \beta_n$ are the *subgradients* at each point.

$$\begin{aligned}
\underset{f,\beta}{\text{minimize}} \quad & \frac{1}{n}\sum_{i=1}^{n}(Y_i - f_i)^2 \\
\text{s.t.} \quad & f_{i'} \geq f_i + \beta_i^T (X_{i'} - X_i) \\
& i, i' = 1, \ldots, n.
\end{aligned}$$

Why? The solution can be viewed as a piecewise-linear convex function whose slopes are precisely the subgradient β_i's:

$$\hat{m}(x) = \max_{i=1,\ldots,n} \left(f_i + \beta_i^T (x - X_i) \right).$$

It is important to note that $\hat{m}$ *interpolates* $\{(X_i, f_i)\}_{i=1}^n$.

(For a proof, see [Boyd and Vandenberghe, 2009].)

The Univariate Case

In the case where the true convex function is univariate, we can do even better.

Note that this is the case with our model because each component function is univariate.

In the univariate case, *sort* the points. Then,

$$\text{convexity} \iff \text{subgradients are nondecreasing!}$$

We only need $n-1$ linear inequalities, instead of $\binom{n}{2}$:

$$\beta_i \leq \beta_{i+1}$$

for $i = 1, \ldots, n$.

Thus, the 1-D convex regression corresponds to the following QP:

$$\begin{aligned} \underset{f,\beta}{\text{minimize}} \quad & \frac{1}{n}\sum_{i=1}^{n}(Y_i - f_i)^2 \\ \text{s.t.} \quad & f_{i+1} = f_i + \beta_i(X_{i+1} - X_i) \\ & \beta_i \leq \beta_{i+1} \\ & \text{for } i = 1, \ldots, n-1. \end{aligned}$$

where the X_i's are sorted, i.e. $X_i < X_{i+1}$.

Additive Convex Regression

Now, we assume an additive model whose components are convex. Then, assuming $\sum_{i=1}^n Y_i = 0$, we obtain the analogous QP:

$$\begin{aligned}
\underset{f,\beta}{\text{minimize}} \quad & \frac{1}{n}\sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p} f_{ij})^2 \\
\text{s.t.} \quad & f_{(i+1)_j,j} = f_{(i)_j,j} + \beta_{(i)_j,j}(X_{(i+1)_j,j} - X_{(i)_j,j}) \\
& \beta_{(i)_j,j} \leq \beta_{(i+1)_j,j} \\
& \text{for } i = 1,\ldots,n-1 \text{ and } j = 1,\ldots,p \\
& \sum_{i=1}^{n} f_{ij} = 0 \quad \text{for } j = 1,\ldots,p
\end{aligned}$$

where $(i)_j$ denotes the ith rank statistic with respect to the values of the jth components of $X_1,\ldots,X_n$.

Identifiability Constraints

$$\sum_{i=1}^{n} f_{ij} = 0 \quad \text{for } j = 1, \ldots, p.$$

These are often called *identifiability constraints* of additive models.

Given that the outputs $Y_1, \ldots, Y_n$ are centered, it is necessary to center the fitted values from each component, since otherwise we can add and subtract the same amount to different components and get the same solution.

The Convexity Pattern Problem

Recall that our regression model is

$$Y_i = \sum_{j=1}^{p} [f_j(X_{ij}) + g_j(X_{ij})] + \varepsilon_i$$

for $i = 1, \ldots, n$, where for each $j = 1, \ldots, p$, f_j is convex and g_j is concave such that *at most* one of f_j and g_j is nonzero.

Good News

The convexity/concavity constraints as well as identifiability constraints are analogous to those in additive convex regression.

For $i = 1, \ldots, n-1$ and $j = 1, \ldots, p$:

$$\begin{aligned} f_{(i+1)_j,j} &= f_{(i)_j,j} + \beta_{(i)_j,j}(X_{(i+1)_j,j} - X_{(i)_j,j}) \\ g_{(i+1)_j,j} &= g_{(i)_j,j} + \gamma_{(i)_j,j}(X_{(i+1)_j,j} - X_{(i)_j,j}) \\ \beta_{(i)_j,j} &\le \beta_{(i+1)_j,j} \\ \gamma_{(i)_j,j} &\ge \gamma_{(i+1)_j,j} \end{aligned}$$

For $j = 1, \ldots, p$:

$$\sum_{i=1}^{n} f_{ij} = 0; \sum_{i=1}^{n} g_{ij} = 0.$$

Not-So-Good News

"...such that *at most* one of f_j and g_j is nonzero."

Integer Variables

We introduce logical (0-1) variables to describe the constraint. For $j = 1, \ldots, p$ and some constant $B > 0$,

$$\|f_j\|_2 = \sqrt{\sum_{i=1}^{n} f_{ij}^2} \leq z_j B$$

$$\|g_j\|_2 = \sqrt{\sum_{i=1}^{n} g_{ij}^2} \leq w_j B$$

$$z_j + w_j \leq 1$$

$$z_j, w_j \in \{0, 1\}.$$

where $f_j = (f_{1j}, \ldots, f_{nj})^T$ and $g_j = (g_{1j}, \ldots, g_{nj})^T$. (cf. SpAM)

Sparsity by Regularization

For each $j = 1, \ldots, p$, $z_j + w_j$ is 1 if the jth component is nonzero and 0 if it is zero. But with the previous construction, $z_j + w_j$ will always tend to 1. Thus, we add a penalty term with some regularization parameter $\lambda > 0$ to the objective:

$$\frac{1}{n}\sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p}(f_{ij} + g_{ij}))^2 + \lambda\sum_{j=1}^{p}(z_j + w_j).$$

Note that the penalty term is exactly the number of nonzero components. It essentially corresponds to a ℓ^0-regularization term, which is not a convex problem.

Towards a Convex Program: Replacing the Objective

We are almost there! One more trick will turn this program into a 0-1 mixed-integer second-order cone program (MISOCP).

We replace

$$\text{minimize} \quad \frac{1}{n}\sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p}(f_{ij} + g_{ij}))^2 + \lambda \sum_{j=1}^{p}(z_j + w_j)$$

by

$$\text{minimize} \quad \frac{t}{n} + \lambda \sum_{j=1}^{p}(z_j + w_j)$$

$$\text{s.t.} \quad \sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p}(f_{ij} + g_{ij}))^2 \leq t$$

Now, the objective is linear and the inequality is a second-order cone.

The MISOCP Formulation

$$\underset{f,g,\beta,\gamma,z,w,t}{\text{minimize}} \quad \frac{t}{n} + \lambda \sum_{j=1}^{p} (z_j + w_j)$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \Big(Y_i - \sum_{j=1}^{p} (f_{ij} + g_{ij})\Big)^2 \leq t$$

$$f_{(i+1)_j,j} = f_{(i)_j,j} + \beta_{(i)_j,j}(X_{(i+1)_j,j} - X_{(i)_j,j})$$

$$g_{(i+1)_j,j} = g_{(i)_j,j} + \gamma_{(i)_j,j}(X_{(i+1)_j,j} - X_{(i)_j,j})$$

$$\beta_{(i)_j,j} \leq \beta_{(i+1)_j,j}$$

$$\gamma_{(i)_j,j} \geq \gamma_{(i+1)_j,j}$$

$$\text{for } i = 1, \ldots, n-1 \text{ and } j = 1, \ldots, p$$

$$\sum_{i=1}^{n} f_{ij} = 0; \quad \sum_{i=1}^{n} g_{ij} = 0$$

$$\|f_j\|_2 \leq z_j B; \quad \|g_j\|_2 \leq w_j B$$

$$z_j + w_j \leq 1$$

$$z_j, w_j \in \{0, 1\}$$

$$\text{for } j = 1, \ldots, p$$

Mixed-Integer Convex Programming

A convex program in which some of the program variables are integers.

Works from [Gomory, 1958], [Sherali and Adams, 1990], [Lovász and Schrijver, 1991], [Balas et al., 1993], ...
Our focus is on 0-1 MISOCPs. Stubbs and Mehrotra (1999) generalized the works in [Balas et al., 1993] on branch-and-cut for general 0-1 mixed-integer convex programming. Drewes (2009) analyzed the results in the case of second-order cone programming.

Mixed-Integer Second-Order Cone Program (MISOCP)

The general form of a 0-1 mixed-integer second-order cone program (MISOCP) can be stated as the following:

$$\begin{aligned} \underset{x \in \mathbb{R}^l}{\text{minimize}} \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & \|P_i x + q_i\|_2 \leq r_i^T x + s_i \quad \forall\, i = 1, \ldots, m \\ & x_j \in \{0, 1\} \quad\quad\quad\quad\quad\quad\quad \forall\, j \in J \subseteq [l] \end{aligned}$$

where l is the total number of program variables and $[l] = \{1, \ldots, l\}$.

Relaxations

Since we have efficient solvers for convex programs, perhaps the most natural way is to relax the integer variables and solve the **relaxed program** for an approximate optimum. That is, we replace the integer constraint $x_j \in \{0, 1\}$ with

$$x_j \in [0, 1]$$

for $j \in J$. The relaxed problem is then convex.

Branch-and-Bound

Let $x^* = (x_1^*, \ldots, x_l)^T$ be an optimal solution to the relaxed problem. For any $j \in J$, if $x_j^* \notin \{0, 1\}$, which is not what we want, we generate two **subproblems**, one with $x_j = 0$ and the other with $x_j = 1$.

We can repeatedly "branch out" to get a binary tree with at most $2^{|J|}$ leaves, corresponding to the $2^{|J|}$ different configurations of the integer variables.

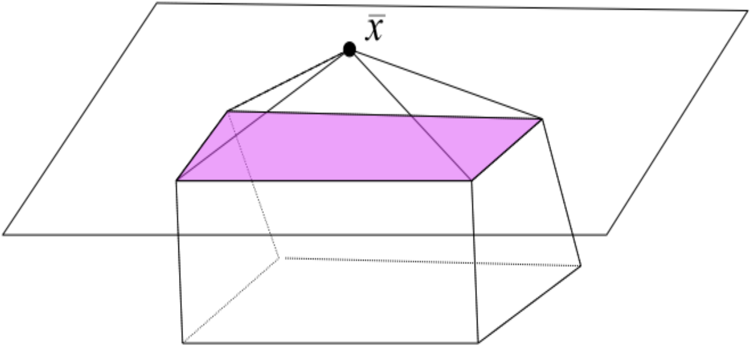
Cuts

We want the tree search to be more efficient by *pruning* the tree!

If x^* is a non-integral solution to some relaxation, then we try to find a linear hyperplane that separates x^* from *all* of the feasible integer points. Such hyperplane is called a **cut**.

A cut need not exist in every case; we need a systematic framework in which we can *generate* cuts.

Example: A Cut



[Ceria]

Branch-and-Cut

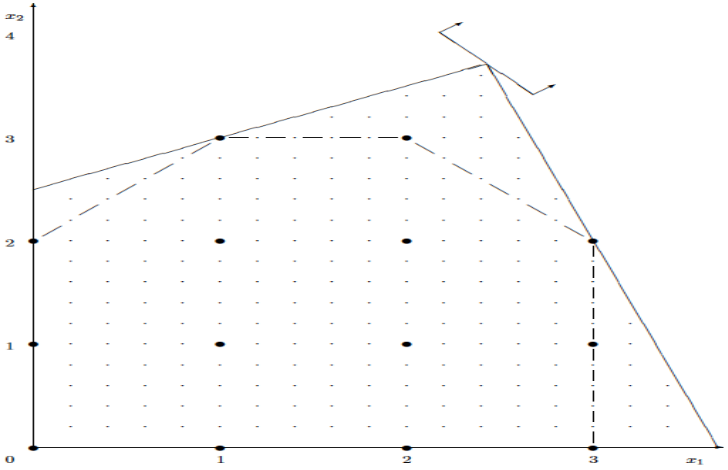
Combines the branch-and-bound algorithm with additional pruning by cuts!

Algorithm 2 Branch-and-Cut (with Most Infeasible Branching)

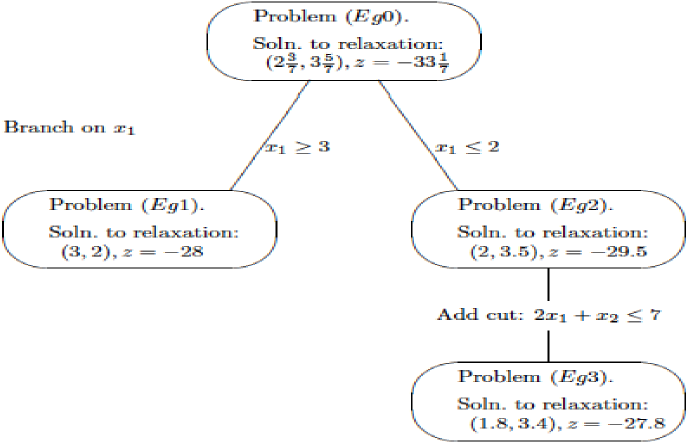
```
Initialize x* ← NULL; OPT ← ∞; 𝒫 ← {The MISOCP problem}
while 𝒫 not empty do
    Remove a problem P from 𝒫
    if relaxation of P is infeasible then
        Continue to next iteration of the loop
    end if
    Solve the relaxed version of P and obtain (x_P, t_P)
    if x_P ∈ {0,1}^|J| and t_P < OPT then
        x* ← x_P; OPT ← t_P
    else if t_P < OPT then
        if there is a cut for x_P then
            Add the cut to P and insert P to 𝒫
            Continue to the next iteration of the loop
        else
            Find j = argmin_{j∈J} |(x_P)_j − 0.5|          # Most Infeasible Branching
            Define P_0 ← (P with x_j = 0); P_1 ← (P with x_j = 1)
            Add P_0 and P_1 to 𝒫
        end if
    end if
end while
return x* and OPT
```

Example: Branch-and-Cut with MILP [Mitchell, 2002]

$$\begin{aligned}
\underset{x_1,x_2}{\text{minimize}} \quad & -6x_1 - 5x_2 \\
\text{s.t.} \quad & 3x_1 + x_2 \le 11 \\
& -x_1 + 2x_2 \le 5 \\
& x_1, x_2 \ge 0 \\
& x_1, x_2 \in \mathbb{Z}
\end{aligned}$$



[Mitchell, 2002]



[Mitchell, 2002]

Lift-and-Project Cuts

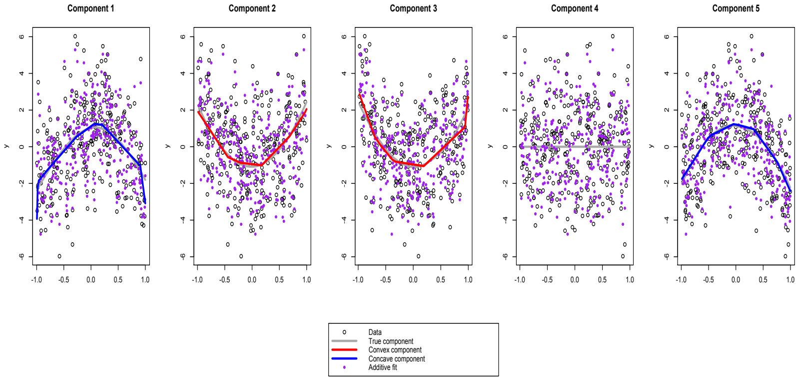
As with other mixed-integer convex programs, for MISOCPs there is a **lift-and-project** construction of a hierarchy of sets that allows one to *generate* cuts [Drewes, 2009].

The Backfitting Version

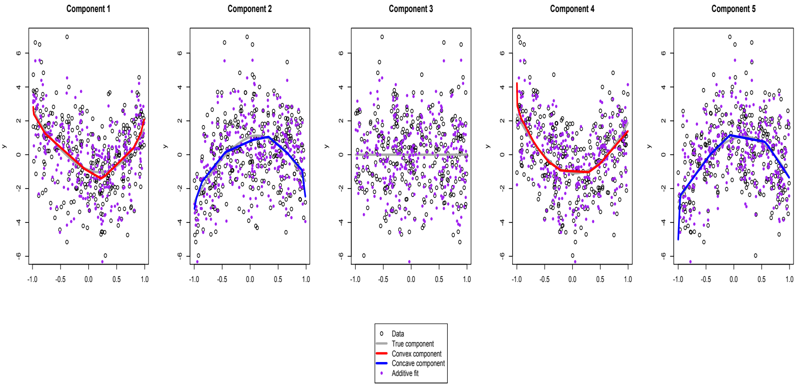
Algorithm 3 The Convexity Pattern Backfitting Algorithm

Given $\{(X_i, Y_i)\}_{i=1}^n \subseteq \mathbb{R}^p \times \mathbb{R}$, where $\sum_{i=1}^n Y_i = 0$
Initialize $\hat{f}_j \equiv 0$ for each $j = 1, \dots, p$
repeat
  for $j = 1, \dots, p$ (or in random order) **do**
    $R_i = Y_i - \sum_{k \neq j} \hat{f}_k(X_{ik})$ for $i = 1, \dots, n$
    $\hat{f}_j = convexity.pattern.1d(\{(X_{ij}, R_i)\}_{i=1}^n)$ # Output is centered
  end for
until change in fitted values is small

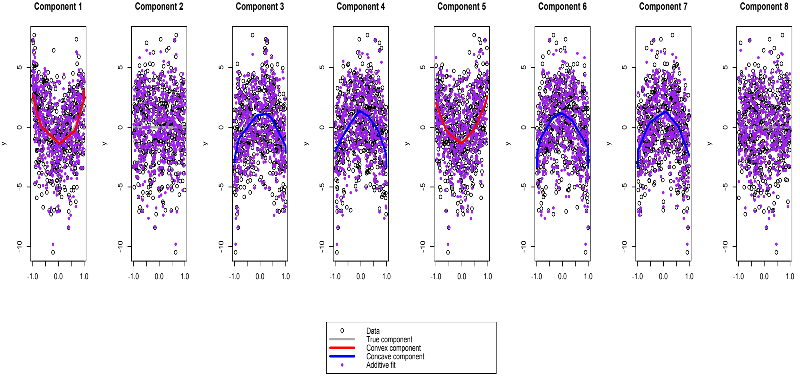
Example: Full MISOCP



Example: MISOCP Backfitting Version



Example: MISOCP Backfitting Version



Limitations of the MISOCP Formulation

- It's still an NP-hard problem, and it does not scale.
- The full MISOCP: for just $n = 500$ and $p = 8$, there are $\sim 20,000$ constraints. In practice (using `Rmosek`), this amounts to $\sim 2,000$ branches with ~ 200 cuts. On a laptop, it takes around 5 minutes.
- The backfitting version: for $p = 20$ or larger, it rarely converges. Also, difficult to analyze theoretically.
- Close/identical points: $\beta_i = \dfrac{f_{i+1} - f_i}{X_{i+1} - X_i}$

Section 3

Lasso Formulation

The Lasso

$$\frac{1}{n}\|Y - X\beta\|_2^2 + \lambda\|\beta\|_1$$

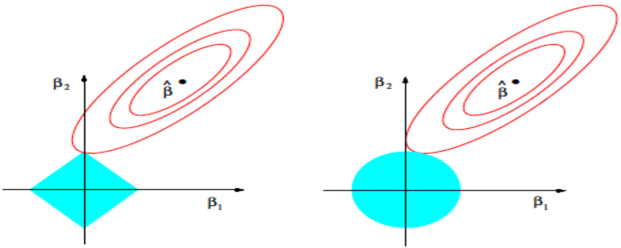


FIGURE 3.11. *Estimation picture for the lasso (left) and ridge regression (right). Shown are contours of the error and constraint functions. The solid blue areas are the constraint regions $|\beta_1| + |\beta_2| \leq t$ and $\beta_1^2 + \beta_2^2 \leq t^2$, respectively, while the red ellipses are the contours of the least squares error function.*

[Hastie et al., 2009]

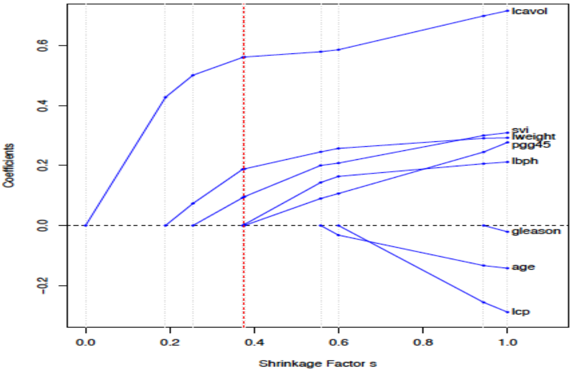


FIGURE 3.10. *Profiles of lasso coefficients, as the tuning parameter t is varied. Coefficients are plotted versus $s = t/\sum_1^p |\hat{\beta}_j|$. A vertical line is drawn at $s = 0.36$, the value chosen by cross-validation. Compare Figure 3.8 on page 65; the lasso profiles hit zero, while those for ridge do not. The profiles are piece-wise linear, and so are computed only at the points displayed; see Section 3.4.4 for details.*

[Hastie et al., 2009]

The Isotonic Pattern Problem

Also known as: the *monotonicity* pattern problem.
In the 1-D, assuming sorted data,

$$\begin{aligned}
\underset{f,g}{\text{minimize}} \quad & \frac{1}{n}\sum_{i=1}^{n}(Y_i - (f_i + g_i))^2 + \lambda\{penalty\} \\
\text{s.t.} \quad & f_i \leq f_{i+1} \\
& g_i \geq g_{i+1} \\
& \text{for } i = 1, \ldots, n-1 \\
& \sum_{i=1}^{n} f_i = 0; \quad \sum_{i=1}^{n} g_i = 0 \\
& \textit{at most one of } f \textit{ and } g \textit{ is nonzero.}
\end{aligned}$$

The Lasso Penalty for the 1-D Isotonic Pattern Problem

Define $\Delta f_i = f_{i+1} - f_i$ and $\Delta g_i = g_{i+1} - g_i$ for $i = 1, \ldots, n-1$. *Because the points are centered, we can recover the points exactly from just knowing the differences.*

Define the penalty as

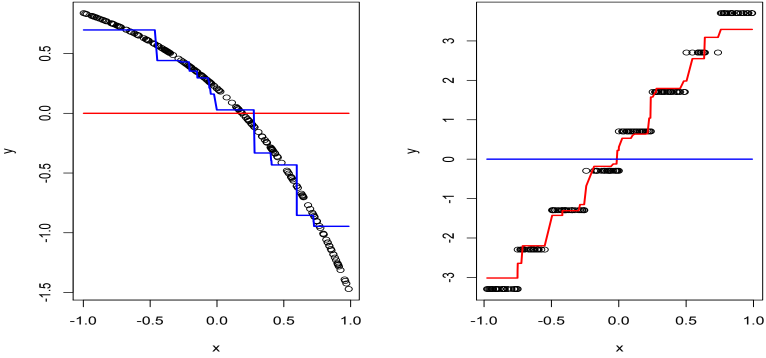
$$\begin{aligned} penalty &= \left\| \begin{bmatrix} \Delta f \\ \Delta g \end{bmatrix} \right\|_1 = \|\Delta f\|_1 + \|\Delta g\|_1 \\ &= \sum_{i=1}^{n-1} (f_{i+1} - f_i) + \sum_{i=1}^{n} (g_i - g_{i+1}) \\ &= (f_n - f_1) + (g_1 - g_n). \end{aligned}$$

The Magic

$$
\begin{aligned}
\underset{f,g}{\text{minimize}} \quad & \frac{1}{n}\sum_{i=1}^{n}(Y_i - (f_i + g_i))^2 + \lambda\{(f_n - f_1) + (g_1 - g_n)\} \\
\text{s.t.} \quad & f_i \leq f_{i+1} \\
& g_i \geq g_{i+1} \\
& \text{for } i = 1, \ldots, n-1 \\
& \sum_{i=1}^{n} f_i = 0; \quad \sum_{i=1}^{n} g_i = 0
\end{aligned}
$$

Claim: With this penalty, only the right pattern will emerge!

Example: The Isotonic Pattern Problem



(Code by Sabyasachi Chatterjee)

The p-Dimensional Isotonic Pattern Problem

$$
\begin{aligned}
\underset{f,g}{\text{minimize}} \quad & \frac{1}{n}\sum_{i=1}^{n}\Big(Y_i - \sum_{j=1}^{p}(f_{ij}+g_{ij})\Big)^2 \\
& + \lambda \sum_{j=1}^{p}\{(f_{(n)_j,j} - f_{(1)_j,j}) + (g_{(1)_j,j} - g_{(n)_j,j})\} \\
\text{s.t.} \quad & f_{(i)_j,j} \le f_{(i+1)_j,j} \\
& g_{(i)_j,j} \ge g_{(i+1)_j,j} \\
& \text{for } i = 1,\ldots,n-1 \text{ and } j = 1,\ldots,p \\
& \sum_{i=1}^{n} f_{ij} = 0; \quad \sum_{i=1}^{n} g_{ij} = 0 \\
& \text{for } j = 1,\ldots,p.
\end{aligned}
$$

Convexity Pattern Problem with ℓ^1-Regularization

Is there an analogous lasso formulation for convexity?

...almost.

A Second Look at the Convexity Pattern Problem

$$
\begin{aligned}
\underset{f,g,\beta,\gamma}{\text{minimize}} \quad & \frac{1}{n}\sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p}(f_{ij}+g_{ij}))^2 + \lambda\{penalty\} \\
\text{s.t.} \quad & f_{(i+1)_j,j} = f_{(i)_j,j} + \beta_{(i)_j,j}(X_{(i+1)_j,j} - X_{(i)_j,j}) \\
& g_{(i+1)_j,j} = g_{(i)_j,j} + \gamma_{(i)_j,j}(X_{(i+1)_j,j} - X_{(i)_j,j}) \\
& \beta_{(i)_j,j} \leq \beta_{(i+1)_j,j} \\
& \gamma_{(i)_j,j} \geq \gamma_{(i+1)_j,j} \\
& \text{for } i = 1,\ldots,n-1 \text{ and } j = 1,\ldots,p \\
& \ldots
\end{aligned}
$$

Where can we induce sparsity?

The subgradients are monotone!

The Convexity Pattern Problem with ℓ^1-Regularization

$$\begin{aligned}
\underset{f,g,\beta,\gamma}{\text{minimize}} \quad & \frac{1}{n}\sum_{i=1}^{n}\Big(Y_i - \sum_{j=1}^{p}(f_{ij}+g_{ij})\Big)^2 \\
& + \lambda \sum_{j=1}^{p}\{(\beta_{(n)_j,j} - \beta_{(1)_j,j}) + (\gamma_{(1)_j,j} - \gamma_{(n)_j,j})\} \\
\text{s.t.} \quad & f_{(i+1)_j,j} = f_{(i)_j,j} + \beta_{(i)_j,j}(X_{(i+1)_j,j} - X_{(i)_j,j}) \\
& g_{(i+1)_j,j} = g_{(i)_j,j} + \gamma_{(i)_j,j}(X_{(i+1)_j,j} - X_{(i)_j,j}) \\
& \beta_{(i)_j,j} \le \beta_{(i+1)_j,j} \\
& \gamma_{(i)_j,j} \ge \gamma_{(i+1)_j,j} \\
& \text{for } i = 1,\ldots,n-1 \text{ and } j = 1,\ldots,p \\
& \sum_{i=1}^{n} f_{ij} = 0; \quad \sum_{i=1}^{n} g_{ij} = 0 \qquad \text{for } j = 1,\ldots,p.
\end{aligned}$$

The 1-D Version

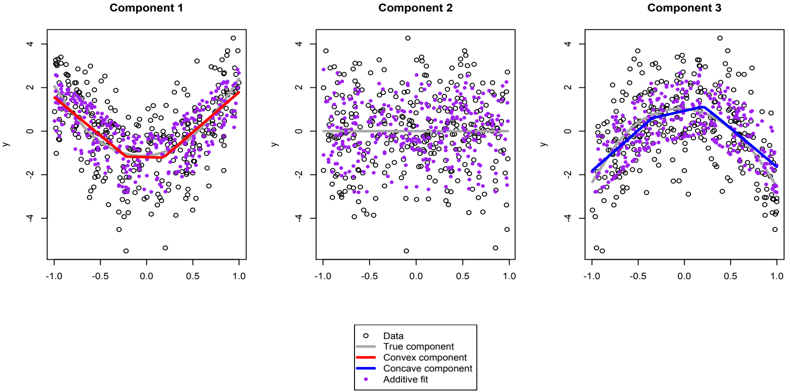
$$\begin{aligned}
\underset{f,g}{\text{minimize}} \quad & \frac{1}{n}\sum_{i=1}^{n}(Y_i - (f_i + g_i))^2 + \lambda\{(\beta_n - \beta_1) + (\gamma_1 - \gamma_n)\} \\
\text{s.t.} \quad & f_{i+1} = f_i + \beta_i(X_{i+1} - X_i) \\
& g_{i+1} = g_i + \gamma_i(X_{i+1} - X_i) \\
& \beta_i \leq \beta_{i+1} \\
& \gamma_i \geq \gamma_{i+1} \\
& \text{for } i = 1, \ldots, n-1 \\
& \sum_{i=1}^{n} f_i = 0; \quad \sum_{i=1}^{n} g_i = 0
\end{aligned}$$

Is it exactly the same?

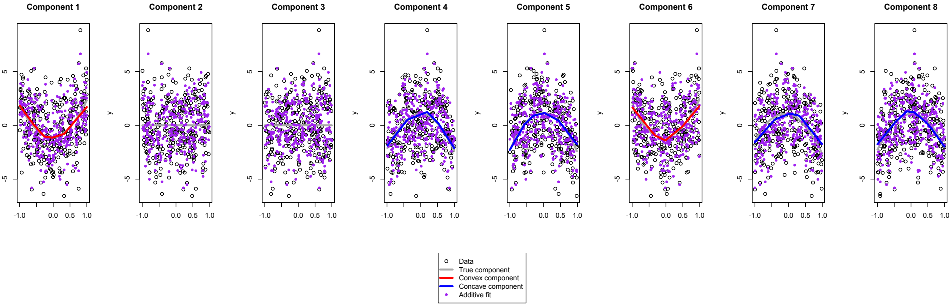
One issue: the fitted values are centered, but the *subgradients* are not.

But it seems to work exactly as it should.

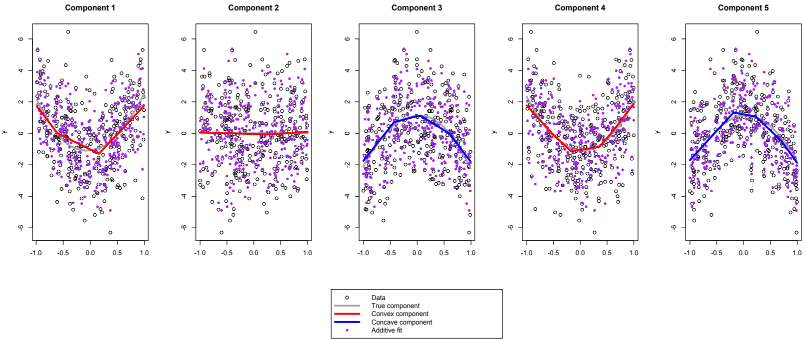
Lasso Example: 3 Components



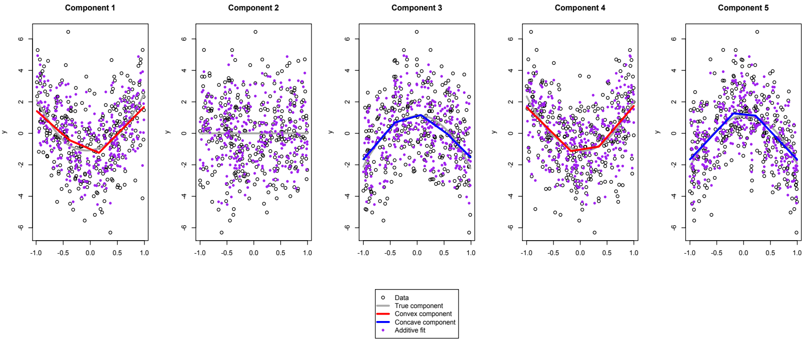
Lasso Example: 8 Components



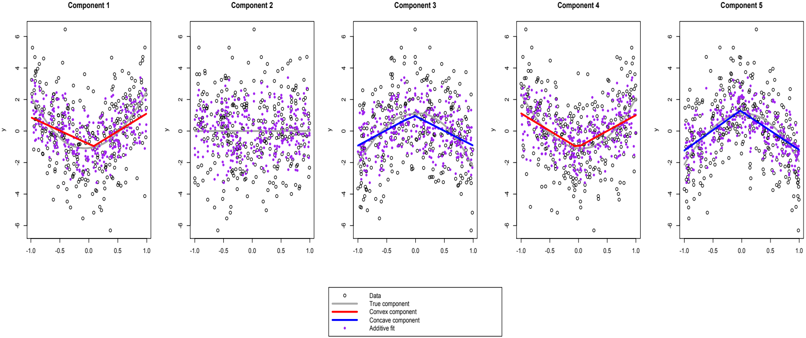
ℓ^1-Regularization Example: $\lambda = 0.01$



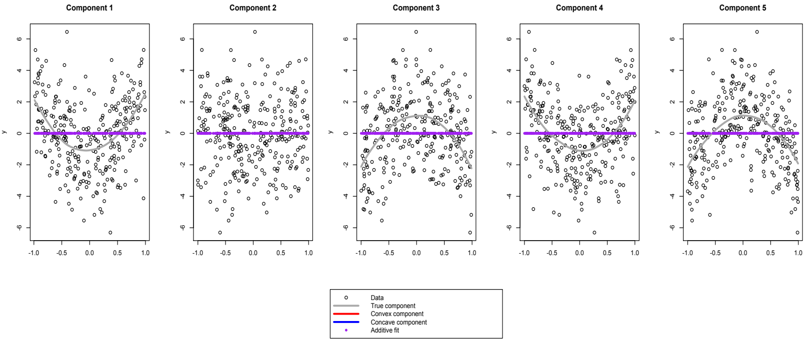
ℓ^1-Regularization Example: $\lambda = 0.02$



ℓ^1-Regularization Example: $\lambda = 0.1$



ℓ^1-Regularization Example: $\lambda = 1.0$



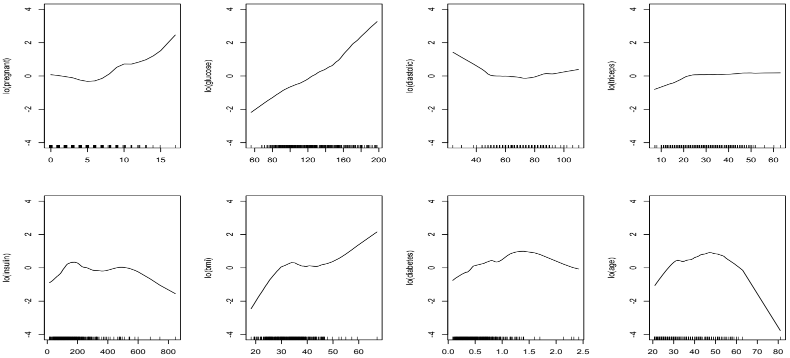
Limitations

- Quality of fit: We may need to re-fit in 1-D (or backfitting) once we have the pattern.
- Global penalty: Less freedom on choice of smoothness for each component.
- Close/identical points: $\beta_i = \dfrac{f_{i+1} - f_i}{X_{i+1} - X_i}$.

Section 4

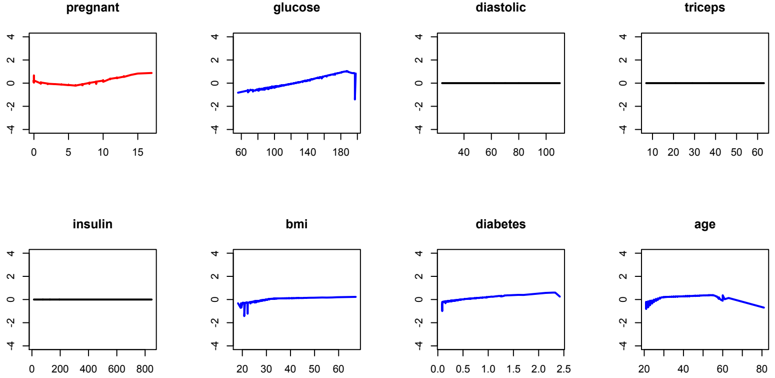
Demo

Logistic Regression: Generalized Additive Models

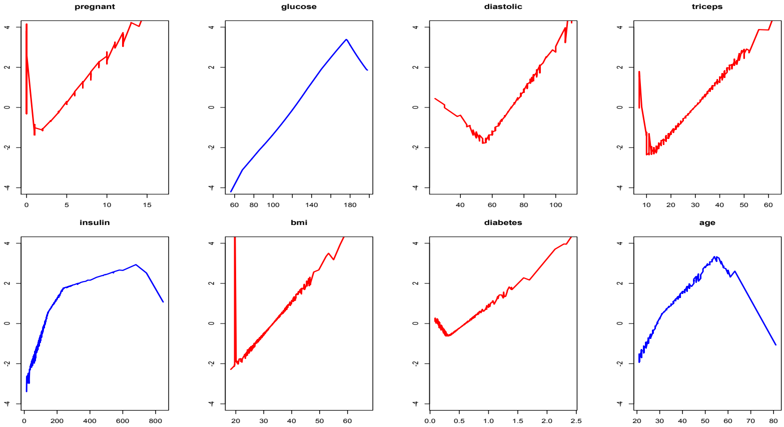


(Data: `pima` from [Faraway, 2014] in `R` package `faraway`)

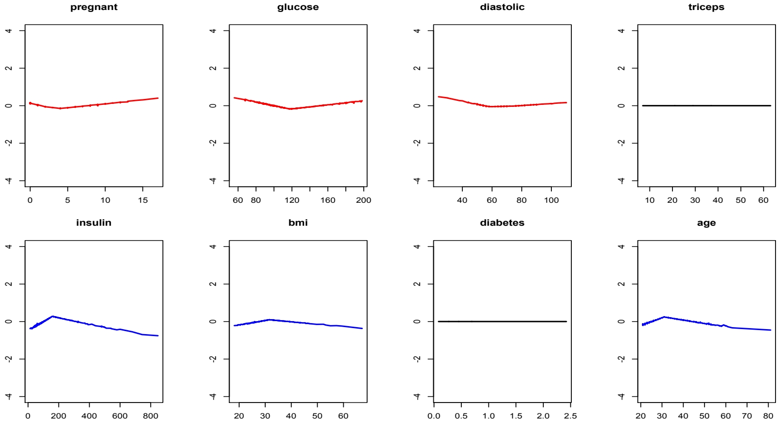
Logistic Regression: Convexity Pattern (Full MISOCP)



Logistic Regression: Convexity Pattern (Backfitting)



Logistic Regression: Convexity Pattern (Lasso)



Sparsity Pattern Recovery: Parametric Lasso

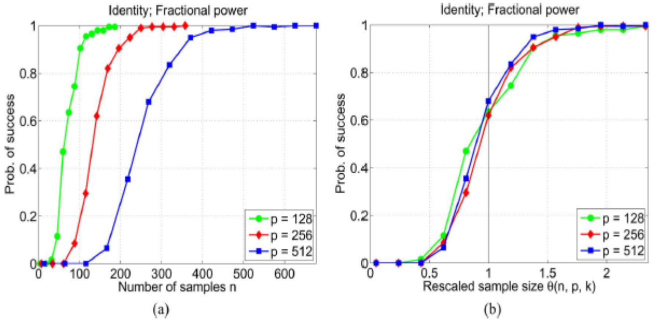
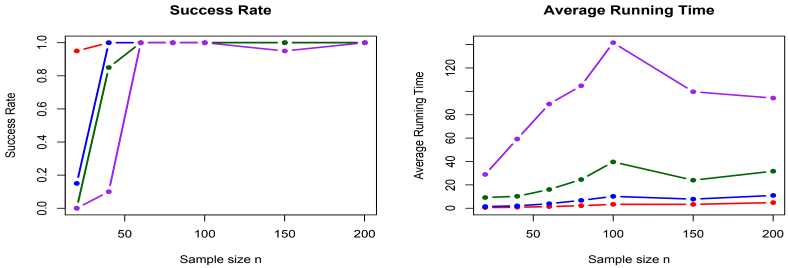


Fig. 1. (a) Plots of the success probability $\mathbb{P}[\mathbb{S}_\pm(\widehat{\beta}) = \mathbb{S}_\pm(\beta^*)]$ of obtaining the correct signed support versus the sample size n for three different problem sizes p, in all cases with sparsity $k = \lceil 0.40 p^{0.75} \rceil$. (b) Same simulation results with success probability plotted versus the rescaled sample size $\theta(n, p, k) = n / [2k \log(p-k)]$. As predicted by Theorems 3 and 4, all the curves now lie on top of one another. See Section VII for further simulation results.

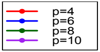
[Wainwright, 2009]

Convexity Pattern Recovery: Full MISOCP

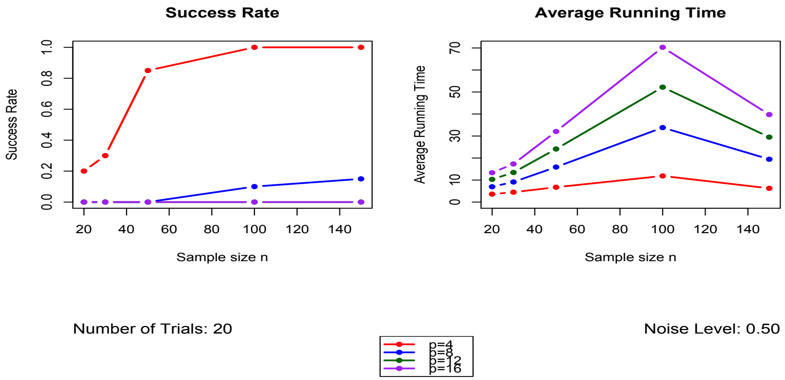


Number of Trials: 20

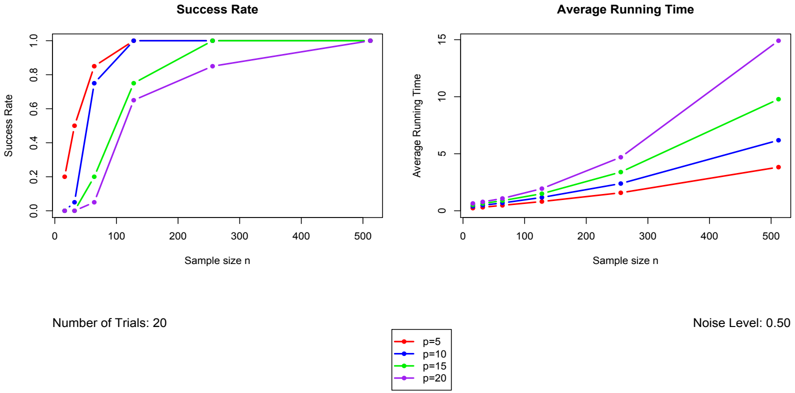
Noise Level: 0.50



Convexity Pattern Recovery: MISOCP with Backfitting



Convexity Pattern Recovery: Lasso



References I

Balas, E., Ceria, S., & Cornuéjols, G. (1993).
A lift-and-project cutting plane algorithm for mixed 0–1 programs.
Mathematical programming, 58(1-3), 295-324.

Boyd, S., & Vandenberghe, L. (2009).
Convex optimization.
Cambridge University Press.

Ceria, S.
Lift-and-project cuts: An efficient solution method for mixed integer programs.
http://www.cs.cmu.edu/~ACO/dimacs/ceria.html.

References II

Drewes, S. (2009).
Mixed integer second order cone programming.
Verlag Dr. Hut.

Julian Faraway (2014)
faraway: Functions and datasets for books by Julian Faraway.
R package version 1.0.6.
http://CRAN.R-project.org/package=faraway

Gomory, R. E. (1958).
Outline of an algorithm for integer solutions to linear programs.
Bulletin of the American Mathematical Society, 64(5), 275-278.

References III

Hastie, T., Tibshirani, R., & Friedman, J. (2009).
The elements of statistical learning (Vol. 2, No. 1).
New York: Springer.

Lovász, L., & Schrijver, A. (1991).
Cones of matrices and set-functions and 0-1 optimization.
SIAM Journal on Optimization, 1(2), 166-190.

Mitchell, J. E. (2002).
Branch-and-cut algorithms for combinatorial optimization problems.
Handbook of Applied Optimization, 65-77.

References IV

MOSEK
Rmosek: The R to MOSEK optimization interface.
R package version 7.0.5. http://rmosek.r-forge.r-project.org/, http://www.mosek.com/

R Core Team (2014).
R: A language and environment for statistical computing.
R Foundation for Statistical Computing, Vienna, Austria. http://www.R-project.org/.

Ravikumar, P., Lafferty, J., Liu, H., & Wasserman, L. (2009).
Sparse additive models.
Journal of the Royal Statistical Society: Series B (Statistical Methodology), 71(5), 1009-1030.

References V

Sherali, H. D., & Adams, W. P. (1990).
A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems.
SIAM Journal on Discrete Mathematics, 3(3), 411-430.

Stubbs, R. A., & Mehrotra, S. (1999).
A branch-and-cut method for 0-1 mixed convex programming.
Mathematical Programming, 86(3), 515-532.

Tibshirani, R. (1996).
Regression shrinkage and selection via the lasso.
Journal of the Royal Statistical Society. Series B (Methodological), 267-288.

References VI

Wainwright, M. J. (2009).
Sharp thresholds for high-dimensional and noisy sparsity recovery using ℓ^1-constrained quadratic programming (Lasso).
Information Theory, IEEE Transactions on, 55(5), 2183-2202.

Thank You!