Additive Models with Sparse Convexity Patterns

Yo Joong (YJ) Choe*
University of Chicago
yjchoe@uchicago.edu

Abstract

In nonparametric statistics, additive modeling is an efficient and interpretable technique for multivariate models, even when the true model is not additive. Shape constraints, such as requiring functions to be convex or monotone, also allow tractable methods that apply to numerous examples.

Here, we propose a novel estimation technique that combines 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*. This model extends the idea of sparse additive models (SpAM) proposed in [17].

We first show that the problem can be expressed as a 0-1 mixed-integer second-order cone program (MISOCP), for which there exist solvers based on heuristics. Then, we describe our recently discovered quadratic program (QP) formulation which extends the idea of Lasso [20]. We present examples as well as simulations comparing the different formulations.

*This work is done as part of the Chicago Center for the Theory of Computing and Allied Areas ("Theory Center") Summer 2014 Research Education for Undergraduates (REU) Program, which is jointly run by the University of Chicago and Toyota Technical Institute at Chicago. This paper contains both original research and reviews of established ideas studied during the program duration. The research is a joint work with Sabyasachi Chatterjee, Min Xu, and Professor John Lafferty.

Contents

Notations and Assumptions

Throughout this paper, we work in a regression setting with sample data. That is, we assume that we have $(X_1, Y_1), \ldots, (X_n, Y_n) \in \mathbb{R}^p \times \mathbb{R}$, where $X_i = (X_{i1}, \ldots, X_{ip})^T$ for each $i = 1, \ldots, n$, which comes from a true regression function $m : \mathbb{R}^p \to \mathbb{R}$ with a Gaussian noise $\varepsilon_i \overset{IID}{\sim} \mathcal{N}(0, \sigma^2)$ according to the following model:

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

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

We say that the model is **additive** if the true function m is a sum of univariate **component functions** $f_1, \ldots, f_p : \mathbb{R} \to \mathbb{R}$ in respective components:

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

for any $x = (x_1, \ldots, x_p)^T \in \mathbb{R}^p$.

In general, we are interested in finding the least-squares fit, or the function that minimizes the **mean squared error (MSE)**, within a specified function space $\mathcal{F}$:

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

1 Introduction

In this paper, we consider a regression problem for an additive model in which each component function is *either convex, concave, or identically zero*. In [17], Ravikumar et al. describe an additive model where each component is either identically zero or not, and they call such pattern a **sparsity pattern**. Here, accordingly, we call the pattern of having either convex, concave, or identically zero component in each dimension as a **(sparse) convexity pattern**. Note that there are 2^p sparsity patterns and 3^p convexity patterns. We may write this model as

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

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. We call this regression problem the **convexity pattern problem**.

The reason behind studying such a nonparametric model is to combine the benefits of additive models and shape (e.g. convexity) constraints, both of which allow *efficient and interpretable* methods for estimation in models with less stringent assumptions than parametric models such as linear and polynomial ones. Additive models give a way to decompose complex high-dimensional structures into sets of univariate components that are easier to fit and interpret. Also, convexity vastly generalizes linearity and yet it is reasonably well understood – in particular, there is a finite-dimensional expression of convexity in the statistical estimation setting that gives tractable reformulations.

In Section 2, we introduce our primitive formulation of the convexity pattern problem. We use some basic results regarding convexity and additive modeling as well as some ideas from sparse additive models in [17] to formulate the problem into a mixed-integer second-order cone program (MISOCP). In Section 3, we give a review of methods for solving MISOCPs, such as branch-and-cut methods with lift-and-project cuts, which are discussed in detail in [7] and [19].

In Section 4, we introduce a new formulation of the problem using ℓ^1-regularization. In particular, this version is a quadratic program (QP) that resembles the lasso in [20]. We first give such a formulation of the isotonic pattern problem, which is a simpler problem where each component is either monotone increasing, monotone decreasing, or identically zero. Then, we give an analogous formulation of the convexity pattern problem and explain its similarities to the isotonic pattern problem and the lasso. In Section 5, we show empirically that this version of the problem works and give an experimental result that suggests the pattern consistency of our formulation.

The lasso formulation is a recent one, and we are currently working to prove consistency – analogous to *sparsistency* in [17] – in terms of finding the correct isotonic and convexity patterns.

2 Formulating the Problem

The convexity pattern problem is a regression problem with an additive model where each component function is either *convex, concave, or identically zero*:

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

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.

We will first introduce some of the simpler and previously studied problems that eventually lead to our initial formulation of the convexity pattern problem, which we will present at the end of this section.

2.1 Convexity as a Set of Affine Constraints

We first introduce the regression problem where the true function is convex. That is,

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

This problem is, in fact, equivalent to the following finite-dimensional quadratic program (QP):

$$\begin{aligned} \underset{f,\beta}{\text{minimize}} \quad & \frac{1}{n}\sum_{i=1}^{n}(Y_i - f_i)^2 \\ \text{s.t.} \quad & f_{i'} \geq f_i + \beta_i^T(X_{i'} - X_i) \\ & i, i' = 1, \ldots, n. \end{aligned} \tag{3}$$

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. More precisely, the solution to (3) can be viewed as a piecewise linear convex function whose slopes are precisely these β_i's:

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

Because of the inequality constraint in (3), $\hat{m}(X_i) = f_i$ for $i = 1, \ldots, n$. The following lemma and proposition formally establish the equivalence.

Lemma 2.1 ([4], p.338)**.** *Let $(x_1, y_1), \ldots, (x_n, y_n)$ be n points such that $x_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$ for $i = 1, \ldots, n$. There exists a convex function $f : \mathbb{R}^p \to \mathbb{R}$ such that*

$$f(x_i) = y_i \qquad \forall\, i = 1, \ldots, n$$

if and only if there exist $\beta_1, \ldots, \beta_n \in \mathbb{R}^p$ such that

$$y_{i'} \geq y_i + \beta_i^T(x_{i'} - x_i) \qquad \forall\, i, i' = 1, \ldots, n.$$

Proof. First, suppose that we have a convex function f that interpolates the n points. At each point x_i, since f is convex, by the supporting hyperplane theorem applied to the epigraph of f at $(x_i, f(x_i))$, there exists a subgradient β_i such that $f(x_{i'}) \geq f(x_i) + \beta_i^T(x_{i'} - x_i)$ for all $i' = 1, \ldots, n$. This proves one half of the statement, because $f(x_i) = y_i$ for all $i = 1, \ldots, n$.

To prove the other half, suppose that there exist $\beta_1, \ldots, \beta_n \in \mathbb{R}^p$ such that $y_{i'} \geq y_i + \beta_i^T(x_{i'} - x_i)$ for all $i, i' = 1, \ldots, n$. Define f such that

$$f(x) = \max_{i=1,\ldots,n} \left(y_i + \beta_i^T(x - x_i)\right)$$

for all $x \in \mathbb{R}^p$. Then, f is a (piecewise linear) convex function. Also, for any i', we have $y_{i'} \geq y_i + \beta_i^T(x_{i'} - x_i)$ for all i, so

$$f(x_{i'}) = \max_{i=1,\ldots,n} \left(y_i + \beta_i^T(x_{i'} - x_i)\right) = y_{i'}.$$

□

Proposition 2.2. *Problems* (2) *and* (3) *are equivalent.*

Proof. By Lemma 2.1, if there is a solution to (2) then there is one achieving the same MSE for (3), and vice versa. □

Proposition 2.2 is crucial, because *convexity is now expressed as a finite set of affine constraints.* One can also express concavity by reversing the inequality.

2.2 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, we can *sort* the points. Then, convexity is equivalent to saying that the subgradients are nondecreasing. Checking if the slopes are nondecreasing is, of course, checking only $n-1$ linear inequalities.

Specifically, we may first assume that our data $(X_1, Y_1), \ldots, (X_n, Y_n) \in \mathbb{R} \times \mathbb{R}$ is sorted by X, i.e.

$$X_i < X_{i+1}$$

for $i = 1, \ldots, n-1$. *We assume here that no points are exactly the same.*

Then, the $\binom{n}{2}$ convexity constraints in (3) are reduced to just $n-1$ of them:

$$\frac{f_i - f_{i-1}}{X_i - X_{i-1}} \leq \frac{f_{i+1} - f_i}{X_{i+1} - X_i}$$

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

We can bring back the auxiliary variables $\beta_1, \ldots, \beta_n$ for these slopes in order to be consistent with the previous formulation. (Also, in practice, this is more numerically stable and does not increase the computation cost at all. See the documentation for [14].)

$$\begin{aligned} \underset{f,\beta}{\text{minimize}} \quad & \frac{1}{n}\sum_{i=1}^{n}(Y_i - f_i)^2 \\ \text{s.t.} \quad & f_{i+1} = f_i + \beta_i(X_{i+1} - X_i) \\ & \beta_i \leq \beta_{i+1} \\ & \text{for } i = 1, \ldots, n-1. \end{aligned} \tag{4}$$

2.3 Additive Modeling

With this formulation of convexity in mind, we now introduce the additive version in higher dimensions. In general, additive modeling is an effective way to deal with multivariate data, because it breaks down the joint multivariate structure into univariate components, which are easier to fit. Additive models are also more interpretable, because it gives one function for each covariate. In practice, additive models often perform well even when the true model is not additive, given that p is not large when compared to n.

First, if we assume that each component function of a model is convex, we can write the regression problem for that model as the following QP:

$$\begin{aligned} \underset{f,\beta}{\text{minimize}} \quad & \frac{1}{n}\sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p} f_{ij})^2 \\ \text{s.t.} \quad & f_{(i+1)_j,j} = f_{(i)_j,j} + \beta_{(i)_j,j}(X_{(i+1)_j,j} - X_{(i)_j,j}) \\ & \beta_{(i)_j,j} \leq \beta_{(i+1)_j,j} \\ & \text{for } i = 1, \ldots, n-1 \text{ and } j = 1, \ldots, p \\ & \sum_{i=1}^{n} f_{ij} = 0 \\ & \text{for } j = 1, \ldots, p \end{aligned} \tag{5}$$

where $(i)_j$ indicates the ith rank statistic with respect to the jth component values of the data. That is, $X_{(i)_j,j}$ is the ith largest value among $X_{1j}, \ldots, X_{nj}$. This notation allows us to *sort* the points as in (4). f_{ij} is the fitted value of the component function f_j at point X_i, and β_{ij} is the subgradient of f_j at X_i.

Note that we have $\sum_{i=1}^n f_{ij} = 0$ for $j = 1, \ldots, p$. These are often called **identifiability constraints**. Assuming that the outputs 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 output. That is, we lose uniqueness and components become non-identifiable. In fact, there are more complicated identifiability issues in our convexity pattern problem, depending on the design (X). We are currently working on solving some of them.

2.4 Convexity Pattern Problem

We are now ready to formulate the convexity pattern problem. Recall from (1) 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.

The biggest issue here is to somehow encode the constraint that at most one of f_j and g_j is nonzero for each component. One natural way to deal with this is to introduce two binary variables that encode whether or not f_j and g_j are each identically zero. This way, we no longer have a convex program but a mixed-integer (0-1) convex program.

Also, recall that $f_j = (f_{1j}, \ldots, f_{nj})^T$ and $g_j = (g_{1j}, \ldots, g_{nj})^T$ are actually n-vectors. We can consider f_j (and g_j) as a group, because we would want to zero out all entries of f_j at once if we are removing the jth component entirely. Thus, we can impose a grouped penalty on each of these vectors, in a similar manner to that in the group lasso [22].

With these in mind, we give a mixed-integer second-order cone program (MISOCP) formulation of the convexity pattern problem:

$$\begin{aligned}
\underset{f,g,\beta,\gamma,z,w}{\text{minimize}} \quad & \frac{1}{n}\sum_{i=1}^n (Y_i - \sum_{j=1}^p (f_{ij} + g_{ij}))^2 + \lambda \sum_{j=1}^p (z_j + w_j) \\
\text{s.t.} \quad & f_{(i+1)_j,j} = f_{(i)_j,j} + \beta_{(i)_j,j}(X_{(i+1)_j,j} - X_{(i)_j,j}) \\
& g_{(i+1)_j,j} = g_{(i)_j,j} + \gamma_{(i)_j,j}(X_{(i+1)_j,j} - X_{(i)_j,j}) \\
& \beta_{(i)_j,j} \leq \beta_{(i+1)_j,j} \\
& \gamma_{(i)_j,j} \geq \gamma_{(i+1)_j,j} \\
& \text{for } i = 1, \ldots, n-1 \text{ and } j = 1, \ldots, p \\
& \sum_{i=1}^n f_{ij} = 0; \; \sum_{i=1}^n g_{ij} = 0 \\
& \|f_j\|_2 \leq z_j B; \; \|g_j\|_2 \leq w_j B \\
& z_j + w_j \leq 1 \\
& z_j, w_j \in \{0, 1\} \\
& \text{for } j = 1, \ldots, p
\end{aligned} \tag{6}$$

where $f_j = (f_{1j}, \ldots, f_{nj})^T$ and $g_j = (g_{1j}, \ldots, g_{nj})^T$ denote the vectors of fitted values from the convex and the concave components, respectively, in each component $j = 1, \ldots, p$. Again, $(i)_j$ indicates the first index of the ith largest number among $X_{1j}, \ldots, X_{nj}$.

$\|\cdot\|_2$ is the Euclidean vector norm, so in particular $f_j = 0$ if $z_j = 0$, and $g_j = 0$ if $w_j = 0$. The use of the Euclidean norm to each 'group' here is analogous to that in the group lasso [22]. Since at most one of z_j and w_j is 1, at most one of f_j and g_j is nonzero. B is some smoothness parameter, which in practice can just be some large number.

λ is a sparsity parameter, which plays a much more important role that B does. Its role is intuitive: if λ is large, any component j, whose fitted values are close to zero, will be completely zeroed out because otherwise $z_j = 1$ or $w_j = 1$ so that the objective will increase by λ. *We can think of this regularization as ℓ^0-regularization, because $\sum_{j=1}^{p}(z_j + w_j)$ is precisely the number of non-zero components.* Since we are already using integer variables, this type of regularization (unlike in the convex case) does not add any more computational cost.

Other parts of the program are self-explanatory, e.g. the first four constraints establish the convexity of f and the concavity of g, and the next two are the identifiability constraints for f_j and g_j.

(6) is perhaps the most natural way to express the convexity pattern problem, but we may replace the quadratic objective so that it looks more like a convex problem (except, of course, that it has integer variables). In particular, we replace the quadratic objective with a quadratic constraint by introducing an auxiliary scalar variable t. By doing this, the objective becomes a linear function of the program variables. Thus, we can restate (6) as the following:

$$\begin{aligned}
\underset{f,g,\beta,\gamma,z,w,t}{\text{minimize}} \quad & \frac{t}{n} + \lambda \sum_{j=1}^{p}(z_j + w_j) \\
\text{s.t.} \quad & \sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p}(f_{ij} + g_{ij}))^2 \leq t \\
& f_{(i+1)_j,j} = f_{(i)_j,j} + \beta_{(i)_j,j}(X_{(i+1)_j,j} - X_{(i)_j,j}) \\
& g_{(i+1)_j,j} = g_{(i)_j,j} + \gamma_{(i)_j,j}(X_{(i+1)_j,j} - X_{(i)_j,j}) \\
& \beta_{(i)_j,j} \leq \beta_{(i+1)_j,j} \\
& \gamma_{(i)_j,j} \geq \gamma_{(i+1)_j,j} \\
& \text{for } i = 1, \ldots, n-1 \text{ and } j = 1, \ldots, p \\
& \sum_{i=1}^{n} f_{ij} = 0; \; \sum_{i=1}^{n} g_{ij} = 0 \\
& \|f_j\|_2 \leq z_j B; \; \|g_j\|_2 \leq w_j B \\
& z_j + w_j \leq 1 \\
& z_j, w_j \in \{0, 1\} \\
& \text{for } j = 1, \ldots, p
\end{aligned} \tag{7}$$

Note that there are $O(np)$ variables, $O(p)$ conic constraints, and $O(np)$ linear equalities/inequalities in this program. There are precisely $2p$ 0-1 integer variables.

This formulation is quite nice. Without the 0-1 integer variables, this program is a second-order cone program (SOCP)[1]: it consists of a linear objective, affine equalities/inequalities, and second-order cones of the form $\|f_j\|_2 \leq z_j B$, $\|g_j\|_2 \leq w_j B$, and $\sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p}(f_{ij} + g_{ij}))^2 \leq t$. In particular, without the integer variables, it is a convex program and there are efficient algorithms for solving it.

Even with the integers, there is still hope. It is perhaps a well-known fact that the 0-1 integer linear programming is NP-complete, and general mixed-integer convex programming is NP-hard. However, there are a few different algorithms that attempt to obtain or approximate the solution as efficiently as possible, using the fact that the program is convex without the integers. (See [9], [1], [18], [12], [19], and [11].) Such algorithms have been studied in the contexts of mixed-integer linear, quadratic, and conic programming. Our interest, MISOCP, is a special case of the latter.

[1] [4] gives a detailed explanation of SOCPs in general. For our purposes, it suffices to know that a SOCP is a convex program.

3 Solving the Full MISOCP with Branch-and-Cut

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

$$\begin{array}{ll} \underset{x\in\mathbb{R}^l}{\text{minimize}} & c^T x \\ \text{s.t.} & Ax = b \\ & \|P_i x + q_i\|_2 \le r_i^T x + s_i \quad \forall\, i = 1, \ldots, m \\ & x_j \in \{0,1\} \qquad\qquad\qquad\quad \forall\, j \in J \subseteq [l] \end{array} \tag{8}$$

where l is the total number of program variables and $[l] = \{1, \ldots, l\}$. Just like a second-order cone program (SOCP), which is a convex program, the 0-1 MISOCP has a linear objective and a set of linear equalities as well as second-order cones. (Note that a linear inequality is a second-order cone with $P_i = O$ and $q_i = 0$.) However, it also allows a subset of its program variables to be 0-1 integer variables.

A survey on MISOCP is given in [3]; a more comprehensive treatment of MISOCPs is given in [7]. The methods described below are in fact general methods for mixed-integer convex programs, including those with integer variables beyond 0 and 1. Most of these were developed first in the context of approximating solutions to mixed-integer linear programs (MILPs), which have been of great interest in theoretical computer science and combinatorics (e.g. the traveling salesman problem) among other areas. (See [6] for a survey.) Here, however, we want to find the actual optimimum (at the cost of more computation) rather than an approximate.

3.1 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**. That is, we replace the integer constraint $x_j \in \{0,1\}$ with

$$x_j \in [0,1] \tag{9}$$

for $j \in J$ in (8). The relaxed problem is then convex.

Since any optimal solution to (8) is also in the feasible set of the relaxed problem, the optimum of the relaxed problem is a lower bound on that of the original problem. The ratio of the original optimum to the relaxed optimum is often called the integrality gap – our goal is then to devise a way to go from the relaxed optimum to the original optimum, thereby closing the integrality gap.

3.2 Branch-and-Bound

Branch-and-bound is a simple yet useful technique that finds the original optimum given the relaxed one. Let $x^* = (x_1^*, \ldots, x_l^*)^T$ be an optimal solution to the relaxed problem as in (9). For any $j \in J$, if $x_j^* \notin \{0,1\}$, then we generate two **subproblems**, one with $x_j = 0$ and the other with $x_j = 1$. We can think of this as "branching" out of the jth covariate – and if we repeat this for each subproblem, at the end we will have something analogous to a binary tree with at most $2^{|J|}$ leaves, corresponding to the $2^{|J|}$ different configurations of the integer variables. Each of its nodes represents a subproblem with some of the integer variables fixed to be either 0 or 1. (For general integers, we "branch" out to $x_j \le \lfloor x_j^* \rfloor$ and $x_j \ge \lceil x_j^* \rceil$.)

Essentially, we are considering the brute-force search of a binary tree that has exponentially many nodes in the number of integer variables. Recall that the problem itself is NP-hard and there are no efficient algorithms; but we do have various heuristics with which we can hope to decrease the running time slightly, such as the most infeasible branching method, which chooses to branch out of $j \in J$ such that x_j^* is the closest to 0.5 among all entries of x^*.

Another trick is to add a pruning step as in the usual tree search. Note that, as we go down the tree, the optimum of the program at each node will only increase, because we are restricting the feasible set every time. Thus, the current optimum gives a lower bound on any optima of the children. Because of this, we can stop branching out of a node if an optimal solution at the node is already an integer solution, i.e. $x_j^* \in \{0,1\}$ for all $j \in J$. In other words, we can store the current integer

optimum at each node and stop branching out of any node that gives an optimum worse than the current integer optimum.

These steps are detailed in Algorithm 1.

Algorithm 1 Branch-and-Bound (with Most Infeasible Branching)

Initialize $x^* \leftarrow NULL$; $OPT \leftarrow \infty$; $\mathcal{P} \leftarrow \{\text{Problem (8)}\}$
while $\mathcal{P}$ not empty **do**
  Remove a problem P from $\mathcal{P}$
  if relaxation of P is infeasible **then**
    Continue to next iteration of the loop
  end if
  Solve the relaxed version of P and obtain its solution x_P and optimum ν_P
  if $x_P \in \{0,1\}^{|J|}$ **and** $\nu_P < OPT$ **then**
    $x^* \leftarrow x_P$; $OPT \leftarrow \nu_P$
  else if $\nu_P < OPT$ **then**
    Find $j = \text{argmin}_{j \in J} |(x_P)_j - 0.5|$ # Most Infeasible Branching
    Define $P_0 \leftarrow (P \text{ with } x_j = 0)$; $P_1 \leftarrow (P \text{ with } x_j = 1)$
    Add P_0 and P_1 to $\mathcal{P}$
  end if
end while
return x^* and OPT

3.3 Cuts

One can infer from the above that what matters the most in improving the efficiency is to prune the tree more efficiently than just bounding. One important type of pruning is called the **cutting plane method**.

If x^* is a non-integral solution to some relaxation, then the cutting plane method attempts to find a linear hyperplane that separates x^* from *all* of the feasible integer points. Such hyperplane is called a **cut**[2]. The cutting plane method is then to find a cut if an optimal solution is not integral, restrict the feasible set to the other side of the cut, and repeat. This procedure can be put another way: the cutting plane method approximates the feasible set by a sequence of linear inequalities generated by cuts.

The first thing to note about cuts is that they need not exist in every case. Thus, in general, we need to find some construction schema which allows us to *generate* a cut in each step. We discuss one such construction in Sections 3.5 and 3.6.

3.4 Branch-and-Cut

The branch-and-cut method combines the traditional branch-and-bound algorithm with additional pruning by cuts. At each node of the branch-and-bound tree, the algorithm tries to find a cut and add the affine inequality to the relaxed problem at the node as an additional constraint. This allows us to restrict the feasible set even further, so that the desired integral solution is (hopefully) found more efficiently. We describe the procedure in Algorithm 2.

As an illustrative example, we present a simple mixed-integer linear program (MILP) in (10). For MILPs, the algorithm is exactly the same, except that the underlying convex program is now a linear program (LP) and the integer variables can take all values in $\mathbb{N} \cup \{0\}$. Given some fractional solution

[2]The idea of cuts was first suggested by Gomory for general integer variables in [9]. Here, we are more interested in lift-and-project cuts in terms of convex relaxations, as suggested in terms of MILPs by Balas et al. in [1]. In the paper, Balas et al. explain the connections between their method and those by Sherali and Adams in [18] and by Lovász and Schrijver in [12]. Unlike Gomory cuts, lift-and-project cuts work more naturally with 0-1 integer variables. Here, we follow the construction by Drewes in [7], which describes the general method developed by Stubbs and Mehrotra in the context of SOCPs in [19].

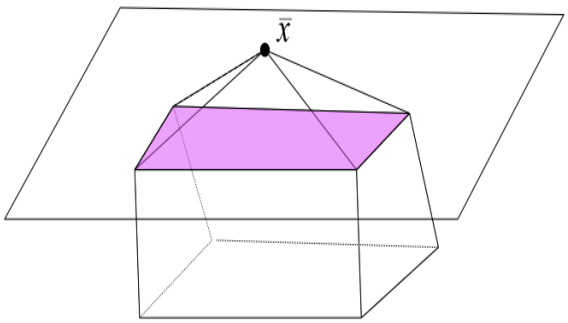

Figure 1: A synthetic example of a cut. The polyhedron represents the feasible set of a convex (in fact, linear) program, and $\overline{x}$ optimizes the objective. The lower, rectangular part of the polyhedron is the convex hull of the zero and one solutions. Thus, the horizontal hyperplane, which intersects with the polyhedron on the purple area, separates the optimal solution to the relaxed problem from all of the integer solutions, and thus it is a cut. (Figure from [5].)

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

Initialize $x^* \leftarrow NULL$; $OPT \leftarrow \infty$; $\mathcal{P} \leftarrow \{\text{Problem (8)}\}$
while $\mathcal{P}$ not empty **do**
  Remove a problem P from $\mathcal{P}$
  if relaxation of P is infeasible **then**
    Continue to next iteration of the loop
  end if
  Solve the relaxed version of P and obtain its solution x_P and optimum ν_P
  if $x_P \in \{0,1\}^{|J|}$ **and** $\nu_P < OPT$ **then**
    $x^* \leftarrow x_P$; $OPT \leftarrow \nu_P$
  else if $\nu_P < OPT$ **then**
    if there is a cut for x_P **then**
      Add the cut to P and insert P to $\mathcal{P}$
      Continue to the next iteration of the loop
    else
      Find $j = \text{argmin}_{j \in J} |(x_P)_j - 0.5|$ # Most Infeasible Branching
      Define $P_0 \leftarrow (P \text{ with } x_j = 0)$; $P_1 \leftarrow (P \text{ with } x_j = 1)$
      Add P_0 and P_1 to $\mathcal{P}$
    end if
  end if
end while
return x^* and OPT

with $x_j^* \notin \mathbb{Z}$, we now branch on a variable x_j into $x_j \leq \lfloor x_j^* \rfloor$ and $x_j \geq \lceil x_j^* \rceil$ instead of $x_j = 0$ and $x_j = 1$. This example is borrowed from [13].

$$\begin{aligned} \underset{x_1,x_2}{\text{minimize}} \quad & -6x_1 - 5x_2 \\ \text{s.t.} \quad & 3x_1 + x_2 \leq 11 \\ & -x_1 + 2x_2 \leq 5 \\ & x_1, x_2 \geq 0 \\ & x_1, x_2 \in \mathbb{Z} \end{aligned} \tag{10}$$

Figure 2 shows the feasible sets of (10) and its relaxed program. First, in the relaxed version of (10), the objective is minimized at a fractional point $(2\frac{3}{7}, 3\frac{5}{7})$ with optimal value $-33\frac{1}{7}$. Then, we branch out of the variable x_1 into two subproblems: one with $x_1 \geq 3$ and the other with $x_1 \leq 2$. The former

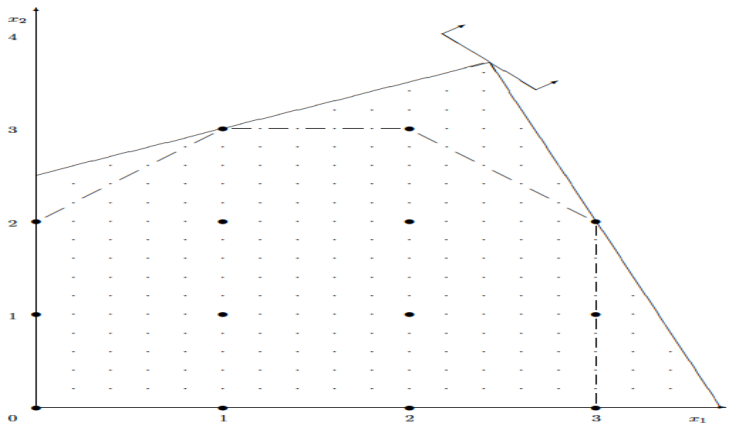

Figure 2: A two-dimensional mixed integer linear program (MILP) (10). Figure from [13].

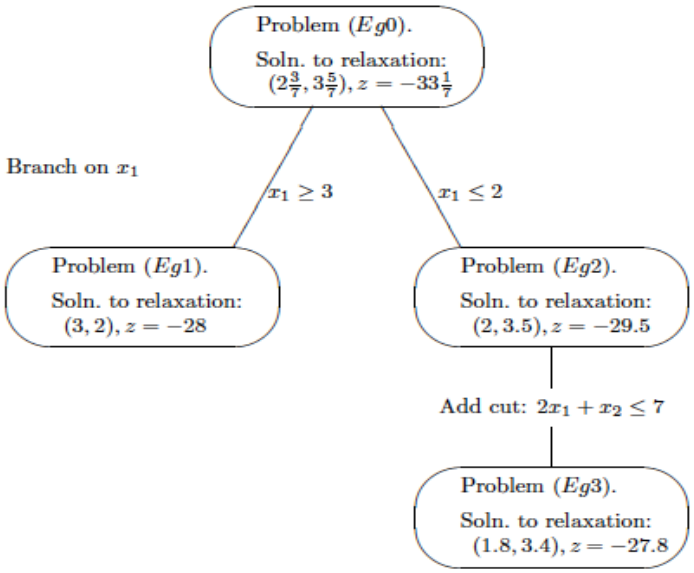

Figure 3: A diagram showing the branch-and-cut tree search procedure for the MILP. Figure from [13].

gives an integer solution $(3, 2)$ with objective -28, which becomes the current best integer solution, and the latter gives yet another fractional solution $(2, 3.5)$ with objective -29.5.

Since -29.5 is less than the current optimum -28, we proceed with the children of this node. But before that, we find that there exists a cut which separates $(2, 3.5)$ from all other integer solutions in the feasible set of the node. This cut is $2x_1 + x_2 \leq 7$, so we add it to the set of constraints and solve the new problem. This gives $(1.8, 3.4)$ with -27.8, which is larger (i.e. worse) than the current optimum -28. Thus, we can stop searching further down and $(3, 2)$ with -28 becomes the final solution. Figure 3 describes this procedure as a tree.

This is a good example where finding a cutting plane reduces the total number of nodes in the tree (i.e. the number of convex programs to be solved): the branch-and-bound algorithm would have branched out of x_2 into two subproblems, whereas the branch-and-cut only created one subproblem.

3.5 Lift-and-Project Construction

Here, we introduce one way to *generate* cuts in the context of the branch-and-cut algorithm. In broad terms, it "lifts" the feasible set into a higher-dimensional space in which we can always generate a cut, and "project" the solution from the cut problem back to the original space.

Speficically, we first define the following sets from (8) and (9):

$$C_0 = \{x \in \mathbb{R}^l : Ax = b,\ \|P_i x + q_i\|_2 \leq r_i^T x + s_i,\ x_j \in \{0,1\}\ \ \forall j \in J\}$$

$$C = \{x \in \mathbb{R}^l : Ax = b,\ \|P_i x + q_i\|_2 \leq r_i^T x + s_i,\ x_j \in [0,1]\ \ \forall j \in J\}$$

Note that C_0 is the feasible set of (8) and C is that of the relaxed program. Then, the ultimate goal of cutting plane methods is *to find affine inequalities whose intersection with C approximates C_0*. This is equivalent to approximating $\text{conv}\,(C_0)$, since no affine inequality can exclude a convex combination of any two integer points and still include the two.

We attempt to do this by introducing a hierarchy of sets from C that eventually leads to C_0. In particular, we construct the hierarchy in a way that corresponds exactly to the branch-and-cut algorithm. For the first branching, we define

$$C_j = \{x \in C : x_j \in \{0,1\}\}$$

for $j \in J$. Note that $\text{conv}\,(C_0) \subseteq \text{conv}\,(C_j)$. Thus, we first attempt to approximate $\text{conv}\,(C_j)$ by lifting C_j into a higher-dimensional space by defining

$$\begin{aligned} M_j(C) = \{\ & (x, u^0, u^1, \lambda^0, \lambda^1) \in \mathbb{R}^l \times \mathbb{R}^l \times \mathbb{R}^l \times \mathbb{R} \times \mathbb{R} : \\ & x = \lambda^0 u^0 + \lambda^1 u^1 \\ & \lambda^0 + \lambda^1 = 1;\ \lambda^0, \lambda^1 \geq 0 \\ & Au^0 = b;\ Au^1 = b \\ & \left\|P_i u^0 + q_i\right\|_2 \leq r_i^T u^0 + s_i \quad \forall\, i = 1, \ldots, m \\ & \left\|P_i u^1 + q_i\right\|_2 \leq r_i^T u^1 + s_i \quad \forall\, i = 1, \ldots, m \\ & u_j^0 = 0;\ u_j^1 = 1 \\ & u_k^0, u_k^1 \in [0,1] \quad \forall\, k \in J \setminus \{j\}\ \} \end{aligned}$$

The intuition behind this construction is that *we consider each point x to be a convex combination of a zero solution u^0 and a one solution u^1*. The zero/one solutions correspond to feasible points with the jth covariate set to be exactly zero/one. λ^0 and λ^1 are the convex coefficients.

As we will see right away, we want $M_j(C)$ to be a convex set. The only constraint that prevents $M_j(C)$ from being convex is the first one: $x = \lambda^0 u^0 + \lambda^1 u^1$, which involves a product term. Fortunately, we can remove this non-linearity by introducing replacement variables $v^0 = \lambda^0 u^0$ and $v^1 = \lambda^1 u^1$. Then, we obtain a convex set $\tilde{M}_j(C)$:

$$\begin{aligned} \tilde{M}_j(C) = \{\ & (x, v^0, v^1, \lambda^0, \lambda^1) \in \mathbb{R}^l \times \mathbb{R}^l \times \mathbb{R}^l \times \mathbb{R} \times \mathbb{R} : \\ & x = v^0 + v^1 \\ & \lambda^0 + \lambda^1 = 1;\ \lambda^0, \lambda^1 \geq 0 \\ & Av^0 = \lambda^0 b;\ Av^1 = \lambda^1 b \\ & \left\|P_i v^0 + \lambda^0 q_i\right\|_2 \leq r_i^T v^0 + \lambda^0 s_i \quad \forall\, i = 1, \ldots, m \\ & \left\|P_i v^1 + \lambda^1 q_i\right\|_2 \leq r_i^T v^1 + \lambda^1 s_i \quad \forall\, i = 1, \ldots, m \\ & v_j^0 = 0;\ v_j^1 = 1 \\ & v_k^0 \in [0, \lambda^0], v_k^1 \in [0, \lambda^1] \quad \forall\, k \in J \setminus \{j\}\ \} \end{aligned}$$

Now, define the natural projection to be

$$P_j(C) = \{x : (x, v^0, v^1, \lambda^0, \lambda^1) \in \tilde{M}_j(C)\}.$$

Then, it is clear that $x \in P_j(C)$ is a convex combination of a zero solution and a one solution. This directly implies that

$$P_j(C) = \operatorname{conv}(C_j).$$

Again, this corresponds to the first node of the branch-and-cut tree that branches out in the jth integer variable. For any intermediate node, let the set of branched-out variables be $B = \{j_1, \ldots, j_{|B|}\} \subseteq J$, in the order starting from the root node. Then, we can analogously define the convex set $\tilde{M}_B(C)$ and the corresponding projection $P_B(C)$. x is now the output of a repeated sequence of convex combinations, in the order of branching out. The only additional constraint is a symmetry condition that we do not state here, and as a result we obtain the projection $P_B(C)$, which is a convex set. By construction, it satisfies

$$P_B(C) \subseteq \bigcup_{j \in B} \operatorname{conv}(C_j).$$

(See pp. 50-51 in [7] for details.)

Recall that the ultimate goal was to build a hierarchy that approximates $\operatorname{conv}(C_0)$. This is established by the following statements, both of which are proved in [19]:

$$P_{j_r}(\ldots(P_{j_2}(P_{j_1}(C)))) = \operatorname{conv}(C_0)$$

for $j_1, \ldots, j_{|J|} \in J$, and

$$(P_J)^{|J|}(C) = \operatorname{conv}(C_0).$$

3.6 Subgradient Cuts

Now that we defined the underlying sets, we can properly generate the cuts. In [19], Stubbs and Mehrotra describe how a cut can be generated in the above lift-and-project hierarchy by solving a simple minimum distance problem: in each node of the tree, represented by the set B, the problem is

$$\underset{x \in P_B(C)}{\text{minimize}} \ \|x - x^*\|_2 \tag{11}$$

where x^* is the optimal point at the node.

Since $P_B(C)$ is a convex set, this is a convex problem with a unique solution. Denote the solution to the above problem by $\hat{x}$. Then, $\hat{x} = x^*$ if and only if $x^* \in P_B(C)$. Now, recall that $P_B(C) \subset \bigcup_{j \in B} \operatorname{conv}(C_j)$. This implies that there can*not* exist a cut of x^* if $x^* \in P_B(C)$. So we are only interested in the case where $x^* \notin P_B(C)$, in which case the objective of (11) is strictly positive.

The following proposition gives us what we naturally call as **subgradient cuts**.

Proposition 3.1. *Let $B \subseteq J$ and x^* be the optimal solution to the relaxed problem at the node corresponding to B. Define $f(x) = \|x - x^*\|$ and let $\hat{x} = \operatorname{argmin}_{x \in P_B(C)} f(x)$. If $x^* \notin P_B(C)$, then there exists a subgradient ξ of f at $\hat{x}$ such that*

$$\xi^T(x - \hat{x}) \geq 0 \tag{12}$$

for all $x \in P_B(C)$ and, at the same time, $\xi^T(x^* - \hat{x}) < 0$. In other words, (12) *cuts off x^* from $P_B(C)$.*

Proof. Since f is convex on $P_B(C)$ and $\hat{x}$ is a minimizer of f in the convex set, by Theorem 3.4.3 in [2], there exists a subgradient ξ of f at $\hat{x}$ such that

$$\xi^T(x - \hat{x}) \geq 0$$

for all $x \in P_B(C)$. Since ξ is a subgradient and $\hat{x} \neq x^*$, we get

$$\xi^T(x^* - \hat{x}) \leq f(x^*) - f(\hat{x}) = 0 - \|\hat{x} - x^*\| < 0.$$

□

The only subgradient of f at $\hat{x}$, at which f is differentiable, is

$$\nabla f(\hat{x}) = \frac{\hat{x} - x^*}{\|\hat{x} - x^*\|}$$

with $\|\hat{x} - x^*\| > 0$. Plugging this in to (12), we obtain the subgradient cut

$$(\hat{x} - x^*)^T x \geq (\hat{x} - x^*)^T \hat{x} \tag{13}$$

for all $x \in \mathbb{R}^l$. This means that we can immediately (and very efficiently) compute a subgradient cut, once we compute the optimal solution to the relaxed convex problem and the solution to the minimum-distance problem (11).

Note that (11) is an SOCP with $O(n|B|)$ variables and $O(m|B|)$ conic constraints. Although this problem is convex, it is not very small, so it is often inefficient to generate a cut at every node. As a result, there are variants such as the cut-and-branch method which generates a cut only for the root node and not for the children nodes.

3.7 Limitations

Although the MISOCP formulation is intuitive and does work for small n and p, it clearly does not scale, even with the branch-and-cut methods. After all, it is still an NP-hard problem, and getting the exact solution is nearly impossible for large n or p. With Rmosek on a personal computer, for example, a synthetic (i.e. nice) data with $n = 500$ and $p = 8$ amounted to about $2,000$ branches and 200 cuts, taking around 5 minutes. A backfitting version, exploiting the additive structure, does not perform much better – although it is faster, it rarely converges for $p \geq 20$. It should also be noted that backfitting algorithms are more difficult to analyze theoretically.

Another concern with the formulation, not necessarily that of mixed-integer programs, is that the subgradients are indeterminate if any two points are identical or very close. Such case is not uncommon in real-world applications. This happens because we have

$$\beta_i = \frac{f_{i+1} - f_i}{X_{i+1} - X_i}$$

as our constraint. If two points are too close, then these subgradients are left undefined (or defined to be arbitrarily large/small), leading to non-convex/concave fits. Unlike the scalability issue, this problem persists even in our new formulation given in the next section.

4 Lasso and a Convex Program Formulation

More recently, we came up with a different formulation of the convexity pattern problem using an interesting equivalence to the idea of lasso [20]. Although we have yet proved pattern consistency, experiments indicate that the new formulation indeed does its job. The most important improvement from the previous MISOCP formulation is that *we removed all the integer constraints and formed a convex program that finds the convexity patterns nearly as correctly.*

4.1 A Review of the Lasso

In [20], Tibshirani introduced the least absolute shrinkage and selection operator, or the **lasso**. The lasso, in its simplest case, can be viewed as regularized linear regression with an ℓ^1 penalty. The general technique of imposing an ℓ^1-norm penalty is called ℓ^1**-regularization** (and analogously ℓ^p-regularization for $p \geq 0$).

In general, regularized least-squares minimizes a linear combination of the MSE and a convex penalty function J of the model parameters: in linear regression, this would be

$$\frac{1}{n} \|Y - X\beta\|_2^2 + \lambda J(\beta) \tag{14}$$

where X is an $n \times p$ design matrix, Y is an $n \times 1$ response vector, β is a p-vector of linear coefficients, and λ is a nonnegative scalar often called the **regularization parameter**. It is important to note that this is precisely the Lagrangian of the convex optimization problem

$$\begin{aligned} \underset{\beta}{\text{minimize}} \quad & \frac{1}{n} \|Y - X\beta\|_2^2 \\ \text{s.t.} \quad & J(\beta) \leq t \end{aligned} \tag{15}$$

for some parameter $t \geq 0$ that is inversely related to λ. Since J is a convex function of β, we can think of this optimization as *fitting the data as much as we can, but within some bound on the complexity and/or the size of the model.*

In the case of the lasso, the penalty term is set to be the ℓ^1-norm of the coefficient vector β. That is, the lasso minimizes

$$\frac{1}{n} \|Y - X\beta\|_2^2 + \lambda \|\beta\|_1 \tag{16}$$

This is the Lagrangian of the optimization problem

$$\begin{aligned} \underset{\beta}{\text{minimize}} \quad & \frac{1}{n} \|Y - X\beta\|_2^2 \\ \text{s.t.} \quad & \|\beta\|_1 \leq t \end{aligned} \tag{17}$$

for some fixed (and inversely related) nonnegative scalars λ and t. Note that this becomes a nice QP simply by introducing auxiliary variables for the absolute values.

The single most remarkable thing about the lasso is that it induces **sparsity**, i.e. it yields a sparse solution $\hat{\beta}$ that contains (many) entries that are *exactly zero.* A mathematical intuition is that the ℓ^1-norm is a convex approximation to the ℓ^0-norm, which is the number of nonzero coefficients. Note that the ℓ^0-norm is neither a norm nor a convex function.

A perhaps more intuitive way to explain the idea is to understand Figure 4 (borrowed from [10]). These plots are based on a 2-dimensional regularized linear regression setting. In both plots, the red ellipses are the contour lines of the objective function, i.e. the mean squared error. $\hat{\beta}$ here is the minimizer of the objective *without* any constraints.

The blue areas are two different feasible sets representing the constraints $J(\beta) := \|\beta\|_1 \leq t$ (left) and $J(\beta) := \|\beta\|_2^2 \leq t$ (right). The left plot corresponds to the lasso, or more generally ℓ^1-regularization; the right corresponds to ridge regression[3], or ℓ^2-regularization.

[3]Ridge regression (or more generally ℓ^2-regularization) minimizes the MSE (or some convex objective) within the round ball as in the second plot in Figure 4. It brings an effect of *shrinking* the size of the coefficients. In contrast to the lasso, however, it does not give a corner solution in general.

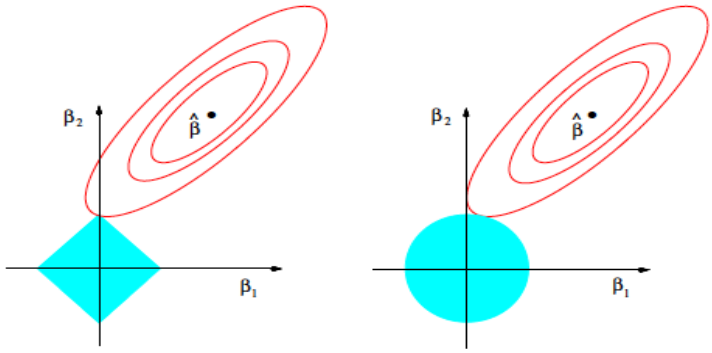

Figure 4: A visualization of the optimization for the lasso (left) and ridge regression (right) in two dimensions. Figure from [10].

The lasso plot explains how having the ℓ^1-norm as the penalty induces a sparse solution. Observe that, drawing contours from the unconstrained minimum $\hat{\beta}$, the first contour line that touches the feasible set (blue diamond) always touches a *corner* of the feasible set, except in degenerate cases. In other words, a corner solution is always the closest feasible point to the unconstrained minimum and thus minimizes the objective within the feasible set. This idea naturally extends to p dimensions for $p > 2$, and the p-dimensional lasso solution is always a corner solution containing zeros. The idea of ℓ^1-regularization further extends to other estimation techniques with different models and/or objective functions, and it is the key to our new formulation of the convexity pattern problem.

*** The role of regularization parameters for the lasso.** If we were to obtain corner solutions, specifically which variables would be zeroed out? Because the objective is the mean squared error, less relevant variables will more likely have zero coefficients.

Specifically, for each data, the actual number of zero coefficients is determined by the regularization parameter λ. Recall that the objective for the lasso (in its Lagrangian form) is

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

Denote the solution as $\hat{\beta}$. If $\lambda = 0$, then $\hat{\beta}$ is the usual least-squares fit. As we increase λ, more weight is given on the penalty term, so we penalize $\|\beta\|_1 = \sum_{j=1}^{p} |\beta_j|$ more and more. Note that this can be thought of as "shrinking" the model: the absolute values of β_j's are forced to be smaller. If $\lambda = \infty$, then every coefficient of $\hat{\beta}$ has to be 0.

Because the solution to the lasso is always a corner solution, we obtain more and more zero coefficients as we increase λ. The path of each coefficient β_j as λ decreases from ∞ to 0 is often called the **regularization path** of the variable j. (We start with ∞ and decrease to 0 so that every entry of $\hat{\beta}$ starts from zero and the next most relevant one becomes nonzero.)

Figure 5, borrowed from [10], is an example of regularization paths of the variables in a prostrate cancer data. Intuitively, smaller shrinkage factor (x-axis) corresponds to larger λ. As we increase s from 0 (i.e. decrease λ from ∞), previously zero coefficients become active. The variables that emerge earlier give a fit with a smaller MSE than other variables.

We can choose an appropriate value of the regularization by cross-validation. In this case, the red dotted line signifies the optimal shrinkage factor, leading to all but three variables having zero coefficients.

4.2 The Isotonic Pattern Problem

In order to explain how the lasso plays its crucial role in our problem, we first introduce a slightly simpler problem which we call the **isotonic pattern problem** or the **monotonicity pattern problem**. This is the same problem as the convexity pattern problem except that we assume each nonzero

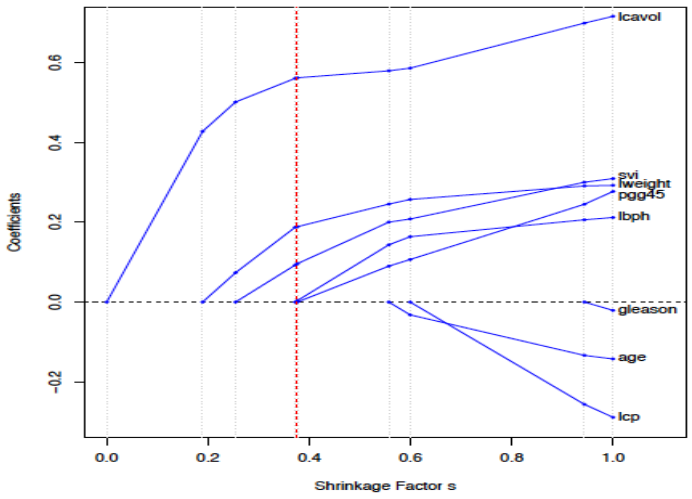

Figure 5: The regularization path of the lasso applied to a prostrate cancer data. Figure and data from [10].

component function to be either monotone increasing or decreasing, instead of either convex or concave.

In the univariate case, assuming sorted distinct points (i.e. $X_i < X_{i+1}$), the isotonic pattern problem corresponds to the following problem:

$$\begin{array}{ll} \underset{f,g}{\text{minimize}} & \frac{1}{n}\sum_{i=1}^{n}(Y_i - (f_i + g_i))^2 \\ \text{s.t.} & f_i \leq f_{i+1} \\ & g_i \geq g_{i+1} \\ & \text{for } i = 1, \ldots, n-1 \\ & \sum_{i=1}^{n} f_i = 0; \quad \sum_{i=1}^{n} g_i = 0 \\ & \textit{at most one of } f \textit{ and } g \textit{ is nonzero.} \end{array} \tag{18}$$

Here, f and g correspond to increasing and decreasing components, respectively. The affine constraints $\sum_{i=1}^{n} f_i = 0$ and $\sum_{i=1}^{n} g_i = 0$ are the usual identifiability constraints for additive models. (See Section 2.3 for explanation.) Just like the solution to the convexity pattern problem is a piecewise linear convex/concave fit, the solution to this problem is a piecewise constant increasing/decreasing fit, i.e. a step function.

(18) looks straightforward, yet we need some way to express the constraint that *"at most one of f and g is nonzero"* in affine/convex terms. To do this, we first define the *differences*, i.e.

$$\begin{aligned} \Delta f_i &= f_{i+1} - f_i \\ \Delta g_i &= g_{i+1} - g_i \end{aligned}$$

for $i = 1, \ldots, n-1$. The monotonicity constraints in (18) ensure that Δf_i are *nonnegative* and Δg_i are *nonpositive*. (Note that the analogues of these difference variables in the convexity pattern problem were the subgradients β_i and γ_i.)

One important observation is that, because the points are centered ($\sum_{i=1}^{n} f_i = 0$ and $\sum_{i=1}^{n} g_i = 0$), *we can recover the original points f_i and g_i exactly by just knowing the differences Δf_i and Δg_i.* In other words, any model in this problem can be parameterized by $\Delta f = (\Delta f_1, \ldots, \Delta f_{n-1})^T$ and $\Delta g = (\Delta g_1, \ldots, \Delta g_{n-1})^T$ only.

At this point, we can make some interesting observations on how these variables relate to the characteristics of the resulting fit. First, the absolute sizes of Δf and Δg correspond to the overall variance of the fit: more concretely, the height of each step in the step function solution. There are two different things that we want about variance. One is that, if we want the step function fit to have low staircases, so as to prevent overfitting, then we need to *shrink* the entries of Δf and Δg. The other is that, if we want the fit to have only few steps, we want many entries of Δf and Δg to be zero and *select* only a few entries to be nonzero.

Second and more importantly, we need at least one of Δf and Δg to be exactly a zero vector, in order to ensure that the fit is either monotone increasing or decreasing. This would mean that if an entry of Δf is nonzero, then every entry of Δg must be zeroed out, and vice versa.

In summary, we want shrinkage and selection of the differences Δf and Δg! Thus, our idea is to use ℓ^1-regularization as in the lasso (least absolute *shrinkage and selection* operator) on this characterization of the model.

Define the penalty as the ℓ^1-norm of the difference vectors:

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

Note that the penalty only involves four terms because Δf is nonnegative and Δg is nonpositive. Now, we apply this penalty for the desired regularization as in (14):

$$\begin{aligned} \underset{f,g}{\text{minimize}} \quad & \frac{1}{n}\sum_{i=1}^{n}(Y_i - (f_i + g_i))^2 + \lambda\{(f_n - f_1) + (g_1 - g_n)\} \\ \text{s.t.} \quad & f_i \le f_{i+1} \\ & g_i \ge g_{i+1} \\ & \text{for } i = 1, \ldots, n-1 \\ & \sum_{i=1}^{n} f_i = 0; \quad \sum_{i=1}^{n} g_i = 0 \end{aligned} \tag{19}$$

As is the lasso, this is a convex QP involving only $O(n)$ linear constraints. We describe a perhaps striking phenomenon in the following (informally stated) conjecture:

Conjecture 4.1 (informally stated)**.** *Suppose we have* (19) *with a true function that is monotone increasing (decreasing). Under "good" conditions, as λ goes from ∞ to 0, the entries of Δf (Δg) will become nonzero* before *those of Δg (Δf). In particular, with an appropriately chosen λ, only the correct pattern will emerge.*

Another way to state this conjecture is that, in the regularization path of the entries of $\begin{bmatrix}\Delta f \\ \Delta g\end{bmatrix}$, the most significant entries from the correct component will become nonzero first.

Figure 6 shows an illustrative example of our approach. The three points – $(X_1, Y_1) = (1, 1)$, $(X_2, Y_2) = (2, 2)$, and $(X_3, Y_3) = (3, 5)$ – are the data from a true model which is monotone increasing. The height of the red solid slope between X_2 and X_3 is Δf_2, and the height of the orange and less steep one between X_1 and X_2 is Δf_1. The heights (drawn as zero in the figure) of the dotted lines between X_1 and X_2 and between X_2 and X_3 are Δg_1 and Δg_2, respectively. Note first that the slopes are not the actual fit but only the heights of the steps in the step function output.

Suppose we solve (19) on these points repeatedly with decreasing λ. As we gradually decrease λ from ∞, Δf_2 (red) will emerge first, *because increasing Δf_2 first will decrease the MSE more than increasing Δf_1 (orange) or decreasing Δg_1 (skyblue) or Δg_2 (blue) by the same amount.* The next difference to become relevant would be the less steep Δf_1 (orange). We stop decreasing λ before one of the other two monotone decreasing components, Δg_1 (skyblue) and Δg_2 (blue), also become nonzero.

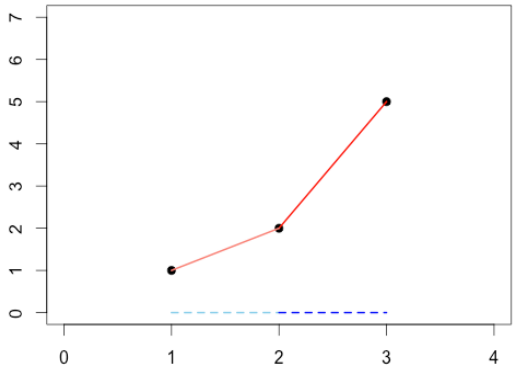

Figure 6: An illustrative example of solving the isotonic pattern problem as in (19). The lines are drawn only to indicate the height, and they do not correspond to the fit itself. The actual fit is a step function.

Finally, we can easily extend this idea to the p-dimensional isotonic problem for $p > 1$. The multivariate extension is formed analogously to (5). This is given by

$$\begin{aligned}
\underset{f,g}{\text{minimize}} \quad & \frac{1}{n}\sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p}(f_{ij} + g_{ij}))^2 \\
& + \lambda \sum_{j=1}^{p}\{(f_{(n)_j,j} - f_{(1)_j,j}) + (g_{(1)_j,j} - g_{(n)_j,j})\} \\
\text{s.t.} \quad & f_{(i)_j,j} \leq f_{(i+1)_j,j} \\
& g_{(i)_j,j} \geq g_{(i+1)_j,j} \\
& \text{for } i = 1,\ldots,n-1 \text{ and } j = 1,\ldots,p \\
& \sum_{i=1}^{n} f_{ij} = 0; \quad \sum_{i=1}^{n} g_{ij} = 0 \\
& \text{for } j = 1,\ldots,p
\end{aligned} \tag{20}$$

where $(i)_j$ again is the first index of the ith largest element in $\{X_{1j},\ldots,X_{nj}\}$. Figure 7 is a sample fit obtained by solving this QP. Notice that the solution is a step function.

Naturally, we can define

$$\begin{aligned}
\Delta f_{(i)_j,j} &= f_{(i+1)_j,j} - f_{(i)_j,j} \\
\Delta g_{(i)_j,j} &= g_{(i+1)_j,j} - g_{(i)_j,j} \\
\Delta f_j &= (\Delta f_{(1)_j,j},\ldots,\Delta f_{(n-1)_j,j})^T \\
\Delta g_j &= (\Delta g_{(1)_j,j},\ldots,\Delta g_{(n-1)_j,j})^T
\end{aligned}$$

for $i = 1,\ldots,n-1$ and $j = 1,\ldots,p$. We can then pose the analogous conjecture:

Conjecture 4.2 (informally stated)**.** *Suppose we have* (20) *with a true function that is monotone increasing (decreasing). Under "good" conditions, as λ goes from ∞ to 0, the entries of Δf_j (Δg_j) will become nonzero* before *those of Δg_j (Δf_j) w.h.p. for each $j = 1,\ldots,p$. In particular, with an appropriately chosen λ, only the correct pattern will emerge.*

We believe that we have a proof of Conjecture 4.1, i.e. the univariate case, and we are currently working on the multivariate case. Note that, here, these conjectures are stated informally; in fact, what we are working to prove is the **pattern consistency** of our algorithm.

Conjecture 4.3 (convexity pattern consistency)**.** *Under certain conditions on λ, the convexity pattern given by solving* (20) *is* consistent*, i.e. the probability that the estimated pattern is correct converges to 1 as $n \to \infty$.*

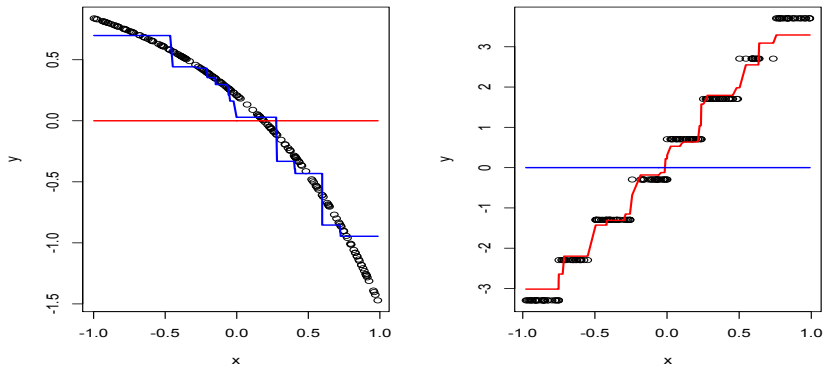

Figure 7: A sample solution to an isotonic pattern problem with $n = 200$ and $p = 8$. λ was chosen at 15. The two plots shown here are two of the eight components, the left with a monotone decreasing true function and the right with a monotone increasing one. The red and blue lines are the increasing and decreasing fitted values, respectively – notice that the other component is completely zeroed out. Original code by Sabyasachi Chatterjee.

4.3 The Convexity Pattern Problem with ℓ^1-Regularization

Now, we return to our original problem in which each nonzero component is either convex or concave. Is there an analogous lasso formulation?

The p-dimensional convexity pattern problem can be stated the following way:

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

Considering how the isotonic (monotonicity) pattern problem turned into the lasso, one observation is crucial:

the subgradients are monotone!

Shrinking and selecting the subgradients β and γ correspond to 1) shrinking the overall fit and 2) forcing either the convex or the concave component to emerge before the other component.

Most of the ideas that we discussed in the previous section apply here, except that knowing the subgradients and the usual identifiability constraints is *not* enough to recover the fitted values. Unlike in the isotonic pattern problem, here we have the differences in subgradients, which are not necessarily centered at zero. Nevertheless, we believe that one additional set of identifiablity constraints will allow us to get around this issue.

We re-state the convexity pattern problem as the following:

$$\begin{aligned}
\underset{f,g,\beta,\gamma}{\text{minimize}} \quad & \frac{1}{n}\sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p}(f_{ij} + g_{ij}))^2 \\
& + \lambda \sum_{j=1}^{p}\{(\beta_{(n)_j,j} - \beta_{(1)_j,j}) + (\gamma_{(1)_j,j} - \gamma_{(n)_j,j})\} \\
\text{s.t.} \quad & f_{(i+1)_j,j} = f_{(i)_j,j} + \beta_{(i)_j,j}(X_{(i+1)_j,j} - X_{(i)_j,j}) \\
& g_{(i+1)_j,j} = g_{(i)_j,j} + \gamma_{(i)_j,j}(X_{(i+1)_j,j} - X_{(i)_j,j}) \\
& \beta_{(i)_j,j} \leq \beta_{(i+1)_j,j} \\
& \gamma_{(i)_j,j} \geq \gamma_{(i+1)_j,j} \\
& \text{for } i = 1,\ldots,n-1 \text{ and } j = 1,\ldots,p \\
& \sum_{i=1}^{n} f_{ij} = 0; \quad \sum_{i=1}^{n} g_{ij} = 0 \qquad \text{for } j = 1,\ldots,p.
\end{aligned} \tag{21}$$

The idea is clearer in the univariate case. Assuming the data is sorted, we have

$$\begin{aligned}
\underset{f,g}{\text{minimize}} \quad & \frac{1}{n}\sum_{i=1}^{n}(Y_i - (f_i + g_i))^2 + \lambda\{(\beta_n - \beta_1) + (\gamma_1 - \gamma_n)\} \\
\text{s.t.} \quad & f_{i+1} = f_i + \beta_i(X_{i+1} - X_i) \\
& g_{i+1} = g_i + \gamma_i(X_{i+1} - X_i) \\
& \beta_i \leq \beta_{i+1} \\
& \gamma_i \geq \gamma_{i+1} \\
& \text{for } i = 1,\ldots,n-1 \\
& \sum_{i=1}^{n} f_i = 0; \quad \sum_{i=1}^{n} g_i = 0.
\end{aligned} \tag{22}$$

As examples and experiments show in the following section, this formulation works well and much more efficiently than the MISOCP formulation in practice. We believe that this is the case because of the the same reasoning behind using ℓ^1-regularization in the isotonic pattern problem.

4.4 Limitations

The lasso formulation indeed does remove the integer variables in our previous MISOCP formulation. However, it still does not remove the instability caused by close or identical points, since it does not change the definition of the subgradients:

$$\beta_i = \frac{f_{i+1} - f_i}{X_{i+1} - X_i}$$

which does not behave well if $X_i \approx X_{i+1}$.

Also, the quality of the lasso fit (i.e. how well it minimizes the mean squared error) is not as good as that of the MISOCP fit. This is largely due to the fact that one global λ affects both the pattern (convex, concave, or zero) and the size of the differences, unlike in the MISOCP formulation where B controls the size and λ only chooses the pattern. However, once we recover the correct pattern efficiently, we may use the standard backfitting algorithm, where for each component we perform a univariate convex/concave regression as in (4). After all, the entire procedure is still much more efficient than solving a MISOCP of similar size.

5 Examples and Experiments

5.1 Examples on Synthetic Data

5.1.1 The MISOCP Formulation

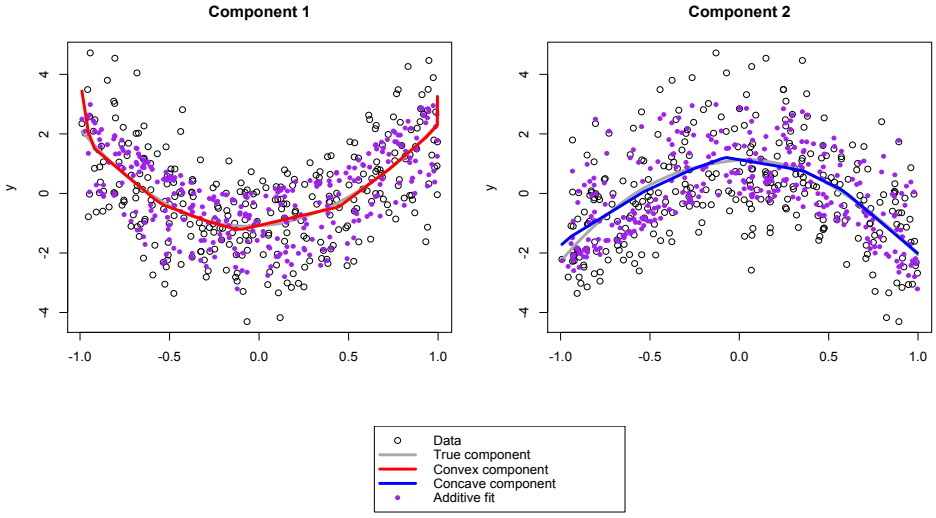

Figure 8: A sample fit using the MISOCP formulation on a 2-dimensional data generated from a true model with one convex and one concave components. The two plots show the true and fitted component functions (must be either concave or convex), as well as the additive data and fitted values. Note that each fitted component is a piecewise-linear function, because of the formulation in Lemma 2.1 – in fact, the number of pieces in the fit is inversely proportional to the degree of regularization (λ).

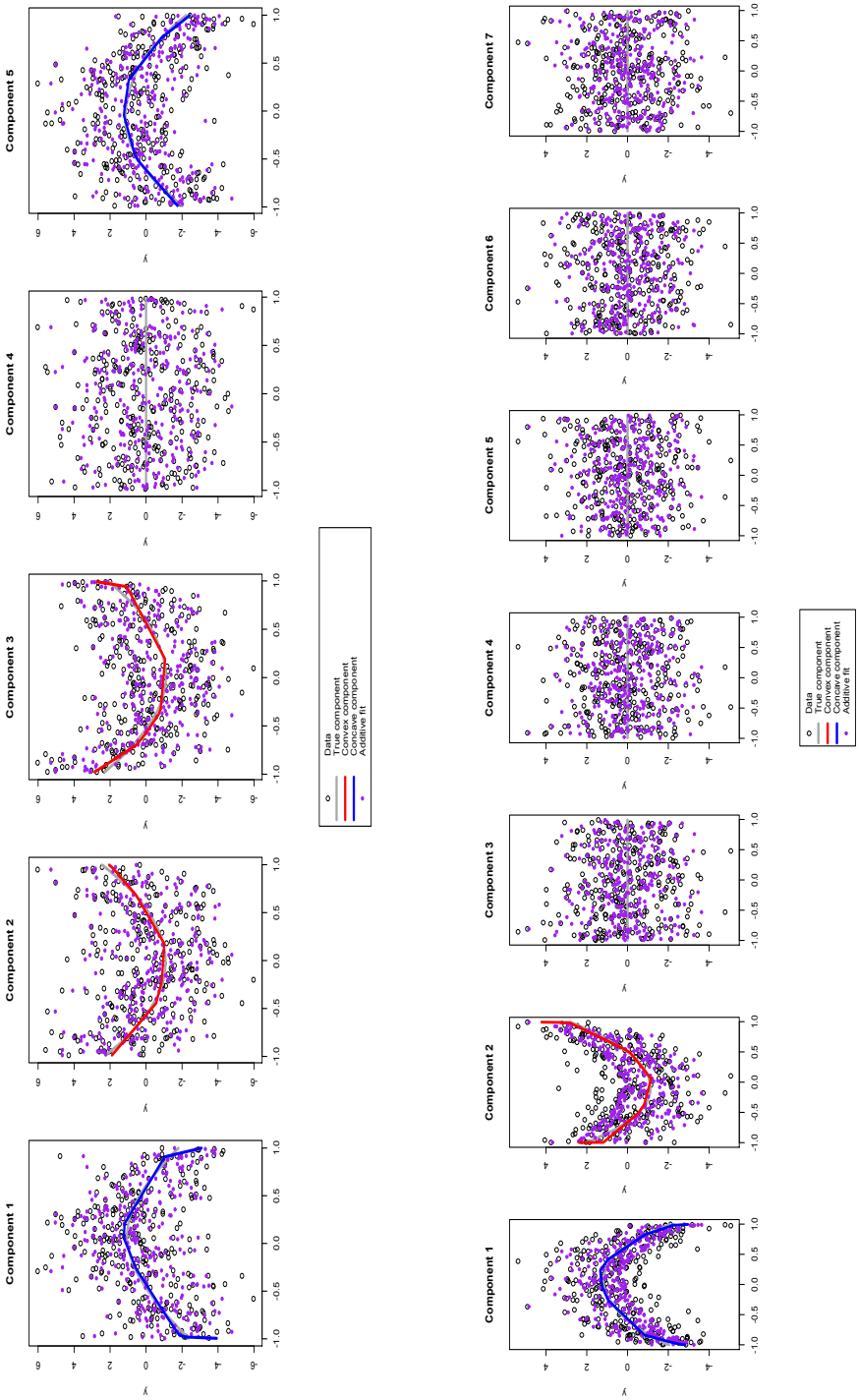

Figure 9: Two more sample fits using the MISOCP formulation on a 5-dimensional data including a sparse component (left) and on a 7-dimensional data with all but two (one convex and one concave) components identically zero (right). A sparse fitted component is not drawn, as in component 4 on the left, for example. Both patterns are recovered exactly.

5.1.2 The Lasso Formulation

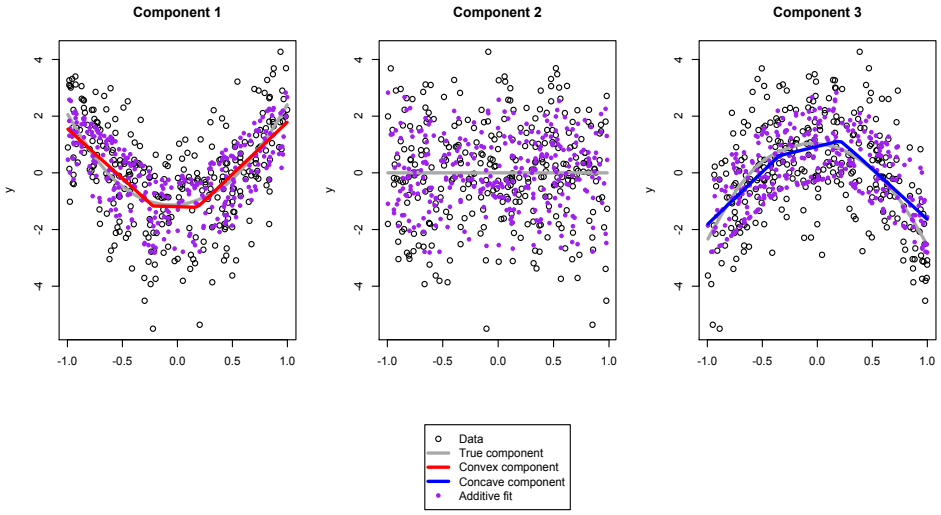

Figure 10: A sample fit using the lasso formulation on a 3-dimensional data with one convex, one sparse, and one concave components. Note that the fits have less number of fits than the MISOCP fits and thus underfit the data. However, as discussed in Section 4.4, once we have the correct pattern, we can use the backfitting algorithm with univariate convex/concave regression. The entire procedure is still efficient, unlike solving the MISOCP.

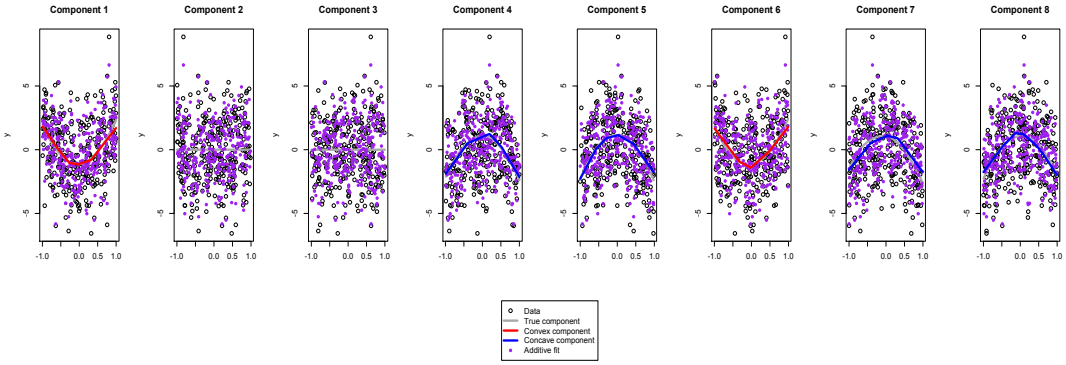

Figure 11: A sample fit using the lasso formulation of the problem on an 8-dimensional data including sparse components. A sparse fitted component is not drawn, as in components 2 and 3. The pattern is recovered correctly, as with all other figures presented above.

5.2 The Effects of Regularization for the Lasso Formulation

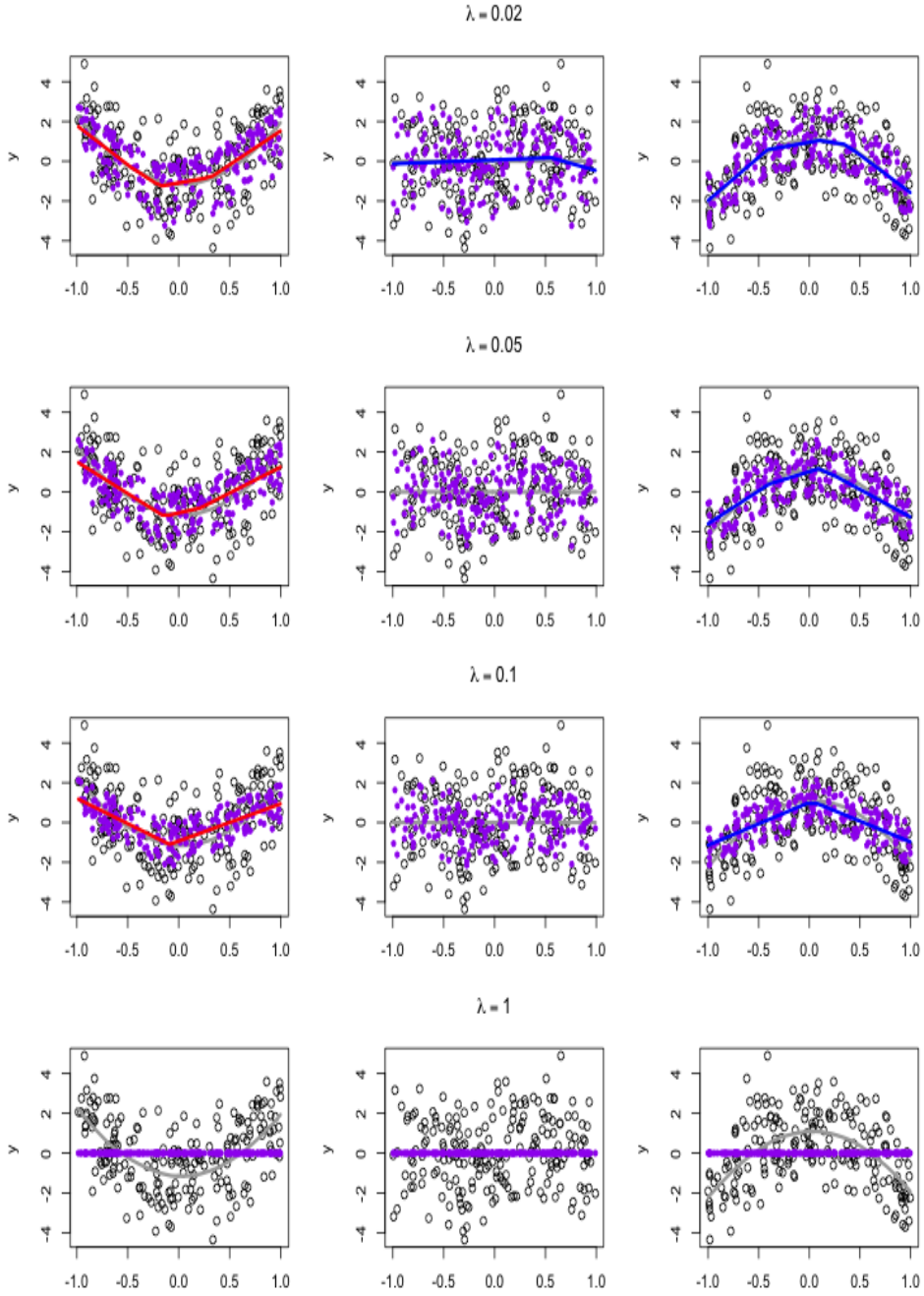

Figure 12: Four test runs on the same 3-dimensional synthetic data using the lasso formulation with different choice of λ's: 0.02, 0.05, 0.1, and 1.0. *The usual notion of regularizataion is embodied in the number of pieces in the piecewise-linear fits.* A less regularized fit (small λ) contains many pieces and often appears to be smooth and strictly convex/concave; a more regularized fit (large λ) has only few. Note that if λ is too small (0.02) then a sparse component is not found; if too large (1) then every component is a single piece, i.e. identically zero.

5.3 Experiments on Pattern Recovery (Consistency)

In [21], Wainwright demonstrates a thresholding phenomenon in the lasso between sample size and the probability of successfully recovering the correct sparsity pattern. In particular, he gives specific choice of λ which he shows is necessary and sufficient for the lasso to be *consistent*, i.e. the probability of recovering the correct sparsity pattern converges to 1 as $n \to \infty$.

Here, we attempt to empirically demonstrate a similar phenomenon with our lasso formulation, which we think is very similar to the standard lasso itself. As in [21], we computed the rate at which the algorithm successfully recovers the convexity pattern for each sample size n and dimension p. This is done by repeatedly generating random convex/concave additive data and counting the number of fits that find the correct pattern. We analogously chose $\sigma = 0.5$, $\lambda = \sigma\sqrt{\frac{\log p}{n}}$ and $k = \lceil 0.40 p^{0.75} \rceil$ where k is the number of sparse components. Each nonzero component was chosen to be convex or concave at random with probability 0.5 each.

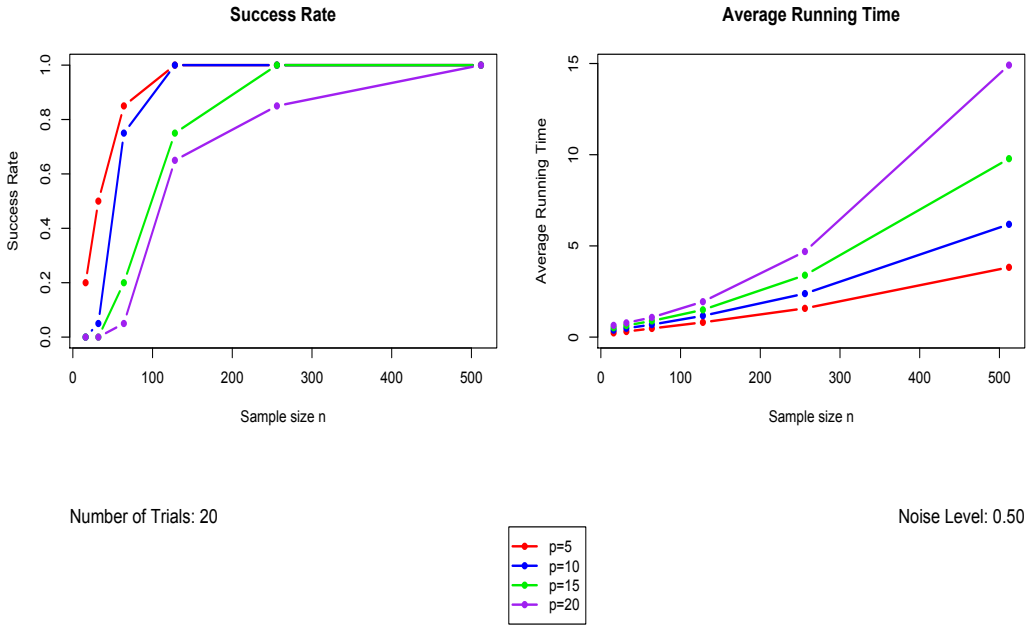

Figure 13: *Left:* A plot of success rate for recovering the convexity pattern using the lasso formulation for $n = 16, 32, 64, 128, 256, 512$ and $p = 5, 10, 15, 20$. Each point represents the ratio of success to the total number of trials (20) for each given n and p. *Right:* The average running time for the trials.

It does seem that the thresholding phenomenon continues to appear here, such as in the $p = 20$ curve with the threshold between 64 and 128. This suggests, again, that the problem is nearly equivalent to the lasso.

As a baseline, we also ran the same experiment on our MISOCP formulation. Figure 14 shows the result. Note the huge difference in the scale of the running times, even though the testing sample sizes and dimensions were reduced by more than half.

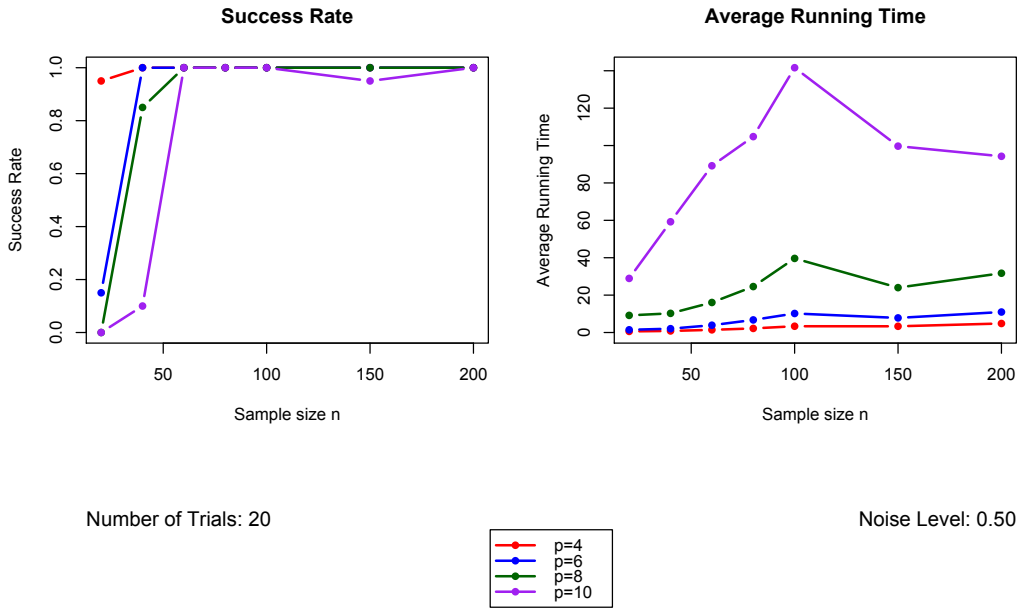

Figure 14: *Left:* A plot of success rate for recovering the convexity pattern using the MISOCP formulation. Each point represents the ratio of success to the total number of trials (20) for given n and p. *Right:* The average running time for the trials. (The sudden spike in running time seems to be an issue with the system rather than the algorithm.)

Acknowledgments

First and foremost, I am very grateful to Professor John Lafferty for all his support and guidance into academic career and research in machine learning. It is impossible to imagine my experiences at the University of Chicago as a machine learning student without his excellent academic and personal mentorship.

Next, I would like to thank Sabyasachi Chatterjee and Min Xu for putting much effort into this work and being great colleagues and mentors to learn from. Sabyasachi, in particular, made this project all the more exciting by coming up with the lasso formulation.

Further, I thank Max Cytrynbaum and Wei Hu for not only their contributions to this problem but for being such wonderful friends to go through the summer REU program together. Special thanks to Max for his insightful talk on SOS-convexity and the theory of positive polynomials and moments.

Finally, I would like to thank Professors László Babai and Stuart Kurtz for hosting and organizing the Chicago Theory Center REU Program. I was fortunate to be a part of the very first summer REU program on computer science at the University of Chicago.

References

[1] 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.

[2] Bazaraa, M. S., Sherali, H. D., & Shetty, C. M. (2013). *Nonlinear programming: theory and algorithms.* John Wiley & Sons.

[3] Benson, H. Y., & Saglam, U. (2005). Mixed-integer second-order cone programming: A survey. *Tutorials in Operations Research*, 13-36.

[4] Boyd, S., & Vandenberghe, L. (2009). *Convex optimization.* Cambridge University Press.

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

[6] Chlamtac, E., & Tulsiani, M. (2012). Convex relaxations and integrality gaps. *In Handbook on semidefinite, conic and polynomial optimization* (pp. 139-169). Springer US.

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

[8] 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`

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

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

[11] Lasserre, J. B. (2001). An explicit exact SDP relaxation for nonlinear 0-1 programs. *In Integer Programming and Combinatorial Optimization* (pp. 293-303). Springer Berlin Heidelberg.

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

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

[14] 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/`

[15] Qi, Y., Xu, M., & Lafferty, J. (2014). Learning High-Dimensional Concave Utility Functions for Discrete Choice Models. *to appear.*

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

[17] 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.

[18] 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.

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

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

[21] 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.

[22] Yuan, M., & Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1), 49-67.