Minimizing a Sum of Clipped Convex Functions

Shane Barratt Guillermo Angeris Stephen Boyd

October 27, 2019

Abstract

We consider the problem of minimizing a sum of clipped convex functions; applications include clipped empirical risk minimization and clipped control. While the problem of minimizing the sum of clipped convex functions is NP-hard, we present some heuristics for approximately solving instances of these problems. These heuristics can be used to find good, if not global, solutions and appear to work well in practice. We also describe an alternative formulation, based on the perspective transformation, which makes the problem amenable to mixed-integer convex programming and yields computationally tractable lower bounds. We illustrate one of our heuristic methods by applying it to various examples and use the perspective transformation to certify that the solutions are relatively close to the global optimum. This paper is accompanied by an open-source implementation.

1 Introduction

Suppose $f : \mathbf{R}^n \to \mathbf{R}$ is a convex function, and $\alpha \in \mathbf{R}$. We refer to the function $\min\{f(x), \alpha\}$ as a *clipped convex function*. In this paper we consider the problem of minimizing a sum of clipped convex functions,

$$\text{minimize} \quad f_0(x) + \textstyle\sum_{i=1}^m \min\{f_i(x), \alpha_i\}, \tag{1}$$

with variable $x \in \mathbf{R}^n$, where $f_0 : \mathbf{R}^n \to \mathbf{R} \cup \{+\infty\}$ and $f_i : \mathbf{R}^n \to \mathbf{R}$ for $i = 1, \ldots, m$ are closed proper convex functions, and $\alpha_i \in \mathbf{R}$ for $i = 1, \ldots, m$. We use infinite values of f_0 to encode constraints on x, *i.e.*, to constrain $x \in \mathcal{X}$ for a closed convex set $\mathcal{X}$ we let $f_0(x) = +\infty$ for all $x \notin \mathcal{X}$. When $f_i(x) > \alpha_i$, the value of the ith term in the sum is *clipped* to α_i, which limits how large each term in the objective can be. Many practical problems can be formulated as instances of (1); we describe a few in §2.

NP-hardness. In general, problem (1) is nonconvex and as a result can be very difficult to solve. Indeed, (1) is NP-hard. We show this by giving a reduction of the subset sum problem to an instance of (1).

The subset sum problem involves determining whether or not there exists a subset of a given set of integers $a_1, \ldots, a_n$ that sum to zero. The optimal value of the problem

$$\begin{array}{ll} \text{minimize} & (a^T x)^2 - n/4 + \sum_{i=1}^n \min\{x_i^2, 1/4\} + \min\{(x_i - 1)^2, 1/4\} \\ \text{subject to} & \mathbf{1}^T x \geq 1, \end{array}$$

which has the form (1), is zero if and only if $x_i \in \{0, 1\}$, at least one of $x_i = 1$, and $a^T x = 0$; in other words, the set $\{a_i \mid x_i = 1\}$ sums to zero. Since the subset sum problem can be reduced to an instance of (1), we conclude that in general our problem is at least as hard as difficult problems like the subset sum problem.

Global solution. There is a simple (exhaustive) method to solve (1) globally: for each subset Ω of $\{1, \ldots, m\}$, we solve the convex problem

$$\begin{array}{ll} \text{minimize} & f_0(x) + \sum_{i \in \Omega} f_i(x) + \sum_{i \notin \Omega} \alpha_i \\ \text{subject to} & f_i(x) \leq \alpha_i, \quad i \in \Omega, \end{array} \tag{2}$$

with variable $x \in \mathbf{R}^n$. The solution to (2) with the lowest optimal value is the solution to (1). This general method is not practical unless m is quite small, since it requires the solution of 2^m convex optimization problems.

In some specific instances of problem (1), we can cut down the search space if we know that a specific choice of $\Omega \subseteq \{1, \ldots, m\}$ implies

$$\{x \mid f_i(x) \leq \alpha_i, \, i \in \Omega\} = \emptyset,$$

which means that the optimal value of (2) is $+\infty$. In this case, we do not have to solve problem (2) for this choice of Ω, as we know it will be infeasible. One simple example where this happens is when the α_i-sublevel sets of f_i are pairwise disjoint, which implies that we only have to solve m convex problems (as opposed to 2^m) to find the global solution. This idea is used in [10] to guide their proposed search algorithm.

Related work. The general problem of minimizing a sum of clipped convex functions was recently considered in [10]. In their paper, they also show that the problem is NP-hard via a reduction to 3-SAT and give a global solution method in a few special cases whenever n is small. They also provide a heuristic method based on cyclic coordinate descent, leveraging the fact that one-dimensional problems are easy to solve.

The idea of using clipped convex functions has appeared in multiple application areas, the most prominent being statistics. For example, the sum of clipped absolute values (often referred to as the *capped* ℓ_1-norm) has been used as a sparsity-inducing regularizer [24, 25, 12]. In particular, [24, 12] make use of the fact that problem (1) can be written as a difference-of-convex (DC) problem and can be approximately minimized via the convex-concave procedure [9] (see Appendix A). The clipped square function (also known as the *skipped-mean* loss) was

also used in [20] to estimate view relations, and in [13] to perform robust image restoration. Similar approaches have been taken for clipped loss functions, where they have been used for robust feature selection [8], regression [22, 16], classification [18, 15, 21], and robust principal component analysis [17].

Summary. We begin by presenting some applications of minimizing a sum of clipped convex functions in §2 to empirical risk minimization and control. We then provide some simple heuristics for approximately solving (1) in §3, which we have found to work well in practice. In §4, we describe a method for converting (1) into a mixed-integer convex program, which is amenable to solvers for mixed-integer convex programs. Finally, we describe an open-source Python implementation of the ideas described in this paper in §5 and apply our implementation to a few illustrative examples in §6.

2 Applications

In this section we describe some possible applications of minimizing a sum of clipped convex functions.

2.1 Clipped empirical risk minimization

Suppose we have data

$$x_1, \ldots, x_N \in \mathbf{R}^n, \quad y_1, \ldots, y_N \in \mathcal{Y}.$$

Here x_i is the ith feature vector, y_i is its corresponding output (or label), and $\mathcal{Y}$ is the output space.

We find parameters $\theta \in \mathbf{R}^n$ of a linear model given the data by solving the *empirical risk minimization* (ERM) problem

$$\text{minimize} \quad \frac{1}{N}\sum_{i=1}^{N} l(x_i^T\theta, y_i) + r(\theta), \tag{3}$$

with variable θ, where $l : \mathbf{R}\times\mathcal{Y} \to \mathbf{R}$ is the loss function, and $r : \mathbf{R}^n \to \mathbf{R}$ is the regularization function. Here the objective is composed of two parts: the loss function, which measures the accuracy of the predictions, and the regularization function, which measures the complexity of θ. We assume that l is convex in its first argument and that r is convex, so the problem (3) is a convex optimization problem.

For a given $x \in \mathbf{R}^n$, our prediction of y is

$$\hat{y} = \operatorname*{argmin}_{y\in\mathcal{Y}} l(x^T\theta^\star, y),$$

where $\theta^\star$ is optimal for (3). For example, in linear regression, $\mathcal{Y} = \mathbf{R}$, $l(z, w) = (z-w)^2$, and $\hat{y} = x^T\theta^\star$; in logistic regression, $\mathcal{Y} = \{-1, 1\}$, $l(z, w) = \log(1 + e^{-wz})$, and $\hat{y} = \mathbf{sign}(x^T\theta^\star)$, where $\mathbf{sign}(z)$ is equal to 1 if $z \geq 0$ and -1 otherwise.

While ERM often works well in practice, it can perform poorly when there are outliers in the data. One way of fixing this is to clip the loss for each data point to a value $\alpha \in \mathbf{R}$, leading to the *clipped ERM* problem,

$$\text{minimize} \quad \frac{1}{N}\sum_{i=1}^{N} \min\{l(x_i^T\theta, y_i), \alpha\} + r(\theta). \tag{4}$$

After solving (or approximately solving) the clipped problem, we can label data points (x_i, y_i) where $l(x_i^T\theta^\star, y_i) \geq \alpha$ as outliers. The clipped ERM problem is an instance of what is referred to in statistics as a *redescending M-estimator* [7, §4.8], since the derivative of the clipped loss goes to 0 as the magnitude of its input goes to infinity. In this terminology, the clip value α is referred to as the *minimum rejection point*.

In §6.1, we show an example where the normal empirical risk minimization problem fails, while its clipped variant has good performance.

2.2 Clipped control

Suppose we have a linear system with dynamics given by

$$x_{t+1} = Ax_t + Bu_t, \quad t = 0, \ldots, T-1,$$

where $x_t \in \mathbf{R}^n$ is the state of the system and $u_t \in \mathbf{R}^p$ denotes the input to the system, at time period t. The dynamics matrix $A \in \mathbf{R}^{n\times n}$ and the input matrix $B \in \mathbf{R}^{n\times m}$ are given.

We are given stage cost functions $g_t : \mathbf{R}^n \times \mathbf{R}^p \to \mathbf{R}$, and an initial state $x^{\text{init}} \in \mathbf{R}^n$. The standard optimal control problem is

$$\begin{array}{ll} \text{minimize} & \sum_{t=0}^{T} g_t(x_t, u_t) \\ \text{subject to} & x_{t+1} = A_t x_t + B_t u_t, \quad t = 0, \ldots, T-1, \\ & x_t \in \mathcal{X}_t, \quad u_t \in \mathcal{U}_t, \quad t = 0, \ldots, T, \\ & x_0 = x^{\text{init}}, \end{array}$$

where, at time t, $\mathcal{X}_t \subseteq \mathbf{R}^n$ is the convex set of allowable states and $\mathcal{U}_t \subseteq \mathbf{R}^m$ is the convex set of allowable inputs. The variables in this problem are the states and inputs, x_t and u_t. If the stage cost function g_t are convex, the optimal control problem is a convex optimization problem.

We define a *clipped optimal control* problem as an optimal control problem in which the stage costs can be expressed as sums of clipped convex functions, *i.e.*,

$$g_t(x, u) = g_t^0(x, u) + \sum_{i=1}^{K} \min\{g_t^i(x, u), \alpha_t^i\},$$

where, for all t and $i = 1, \ldots, K$, the functions $g_t^i : \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}$ are convex and $\alpha_t^i \in \mathbf{R}$. This gives another instance of our general problem (1).

A simple but practical example of a clipped control problem is described in §6.3. The problem is to design a lane change trajectory for a vehicle; the stage cost is small when the vehicle is centered in either lane, which we express as a sum of two clipped convex functions.

3 Heuristic methods

There are many methods for approximately solving (1). In this section we describe a few heuristic methods that we have observed to work well in practice.

Bi-convex formulation. Throughout this section, we will make use of a simple reformulation of (1) as the bi-convex problem

$$\begin{array}{ll} \text{minimize} & L(x,\lambda) = f_0(x) + \sum_{i=1}^m \lambda_i f_i(x) + (1-\lambda_i)\alpha_i \\ \text{subject to} & 0 \leq \lambda \leq \mathbf{1}, \end{array} \tag{5}$$

with variables $\lambda \in \mathbf{R}^m$ and $x \in \mathbf{R}^n$. (We note that this reformulation was also pointed out in [22, §3].) The equivalence follows immediately from the fact that

$$\min\{a, b\} = \min_{0 \leq \lambda \leq 1} \left(\lambda a + (1-\lambda)b\right).$$

Nonlinear programming. When f_i are all smooth functions and $\mathbf{dom}\, f_0$ is representable as the sublevel set of a smooth function, it is possible to use general nonlinear solvers to (approximately) solve (5).

Alternating minimization. Another possibility is to perform alternating minimization on (5), since each respective minimization is a convex optimization problem. In alternating minimization, at iteration k, we solve (5) while fixing $\lambda = \lambda^{k-1}$, resulting in x^k. We then solve (5) while fixing $x = x^k$, resulting in λ^k. It can be shown that

$$(\lambda^k)_i = \begin{cases} 1 & f_i(x^k) \leq \alpha_i \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

is a solution for minimization over λ with fixed $x = x^k$.

Inexact alternating minimization. Although alternating minimization often works well, we have found that inexact minimization over λ works better in practice. Instead of fully minimizing over λ, we instead compute the gradient of the objective with respect to λ,

$$g_i = (\nabla_\lambda L(x^k, \lambda))_i = f_i(x^k) - \alpha_i.$$

We then perform a signed projected gradient step on λ with a fixed step size $\beta > 0$ (we have found $\beta = 0.1$ works well in practice, though a range of values all appear to work equally as well). This results in the update

$$\lambda^k = \Pi_{[0,1]^m}(\lambda^k - \beta \mathbf{sign}(g)),$$

where **sign** is applied elementwise to g, and $\Pi_{[0,1]^m}$ denotes the projection onto the unit box, given by

$$(\Pi_{[0,1]^m}(z))_i = \begin{cases} 1 & z_i \geq 1, \\ z_i & 0 < z_i < 1, \\ 0 & \text{otherwise.} \end{cases}$$

The final algorithm is described below.

Algorithm 3.1 *Inexact alternating minimization.*

given initial $\lambda^0 = (1/2)\mathbf{1}$, step size $\beta = 0.1$, and tolerance $\epsilon > 0$.
for $k = 1, \ldots, n_{\text{iter}}$

1. *Minimize over x.* Set x^k to the solution of the problem

$$\text{minimize} \quad f_0(x) + \textstyle\sum_{i=1}^m \lambda_i^{k-1} f_i(x) + (1 - \lambda_i^{k-1})\alpha_i.$$

2. *Compute the gradient.* Set $g_i = f_i(x^k) - \alpha_i$.
2. *Update λ.* Set $\lambda^k = \Pi_{[0,1]^m}(\lambda^{k-1} - \beta \mathbf{sign}(g))$.
3. *Check stopping criterion.* Terminate if $\|\lambda^k - \lambda^{k-1}\|_1 \leq \epsilon$.

end for

Algorithm 3.1 is a descent algorithm in the sense that the objective function of (5) decreases after every iteration. It is also guaranteed to terminate in a finite amount of time, since there is a finite number of possible values of λ. We also note that alternating minimization can be thought of as a special case of algorithm 3.1 where $\beta \geq 1$. In practice, we have found that algorithm 3.1 often finds the global optimum in simple problems and appears to work well on more complicated cases. We use algorithm 3.1 in our generic `cvxpy` implementation (see §5).

4 Perspective formulation

In this section we describe the perspective formulation of (1). The perspective formulation is a mixed-integer convex program (MICP), for which specialized solvers with reasonable practical performance exist. The perspective formulation can also be used to compute a lower bound on the original objective by relaxing the integral constraints, as in [11], as well to obtain good initializations for any of the procedures described in §3.

Perspective. Following [14, §8], we define the perspective (or recession) of the closed convex function f with $0 \in \mathbf{dom}\, f$ as[1]

$$f^{\text{p}}(x,t) = \begin{cases} tf(x/t) & t > 0, \\ \lim_{\gamma \downarrow 0} \gamma f_0(x/\gamma) & t = 0, \\ +\infty & \text{otherwise,} \end{cases} \tag{7}$$

[1] If $0 \notin \mathbf{dom}\, f$, replace $\gamma f_0(x/\gamma)$ with $\gamma f_0(y + x/\gamma)$ for any $y \in \mathbf{dom}\, f$. See [14, Thm. 8.3] for more details.

for $(x,t) \in \mathbf{R}^n \times \mathbf{R}_+$. We will use the fact that the resulting function f^{p} is convex [3, §3.2.6].

Superlinearity assumption. If f is superlinear, *i.e.*, if for all $x \in \mathbf{R}^n \setminus \{0\}$, we have

$$\lim_{t\to\infty} \frac{f(tx)}{t} = +\infty, \tag{8}$$

then

$$f^{\mathrm{p}}(x,t) = \begin{cases} tf(x/t) & t > 0 \\ 0 & t = 0,\ x = 0, \\ +\infty & \text{otherwise}, \end{cases} \tag{9}$$

since the limit in (7) is equal to the limit in (8) unless $x = 0$.

There are many convex functions that satisfy this superlinearity property. Some examples are the sum of squares function and the indicator function of a compact convex set. Since we will make heavy use of property (9) in this section, we will assume that f_0 is superlinear for the remainder of this section. If f_0 is not superlinear, then it can be made superlinear by adding, *e.g.*, a small positive multiple of the sum of squares function.

Conic representation of the perspective. We note that representing the epigraph of the perspective of a function is often simple if the function has a conic representation [6]. More specifically, if f has a conic representation

$$f(x) \le v \iff Ax + bv + c \in \mathcal{K},$$

for some closed convex cone $\mathcal{K}$, then the perspective of f has a conic representation given by

$$f^{\mathrm{p}}(x,t) \le v \iff Ax + bv + tc \in \mathcal{K}.$$

This fact allows us to use a conic representation of the perspective and avoid issues of non-differentiability and division-by-zero that we might encounter with direct numerical implementations of the perspective [11, §2].

Perspective formulation. We define the *perspective formulation* of (1) as the following MICP:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^m f_i^{\mathrm{p}}(z_i,t_i) + (1-t_i)\alpha_i + \frac{1}{m}\left(f_0^{\mathrm{p}}(z_i,t_i) + f_0^{\mathrm{p}}(x - z_i, 1-t_i)\right) \\ \text{subject to} & t \in \{0,1\}^m, \end{array} \tag{10}$$

with variables $x, z_i \in \mathbf{R}^n$ for $i = 1,\ldots,m$ and $t \in \mathbf{R}^m$. Any MICP solver that can handle the functions f_i^{p} for $i = 0,\ldots,m$ can be used to solve (10).

Proof of equivalence. To show that (10) is equivalent to the original problem (1), first take (x, t, z_i) that are feasible for (10). Since t is Boolean, for each i we have $t_i = 0$ or $t_i = 1$. Since $f_0^{\mathrm{p}}(z_i, t_i)$ must be finite (as this point is feasible), then $t_i = 0$ implies that $z_i = 0$ (due to (9)). Similarly, when $t_i = 1$ we must have $z_i = x$. Therefore the ith term in the sum becomes

$$t_i f_i(x) + (1 - t_i)\alpha_i + \frac{1}{m} f_0(x).$$

Summing over the index i yields that problem (10) is equivalent to

$$\begin{array}{ll} \text{minimize} & f_0(x) + \sum_{i=1}^m t_i f_i(x) + (1 - t_i)\alpha_i \\ \text{subject to} & t \in \{0, 1\}^m. \end{array} \tag{11}$$

Partially minimizing (11) over t, we find that x is a feasible point for (1) with the same objective value.

Now take x feasible for (1). Let

$$t_i = \begin{cases} 1 & f_i(x) \leq \alpha_i \\ 0 & \text{otherwise,} \end{cases} \quad i = 1, \ldots, m,$$

and $z_i = t_i x$. Then (x, t, z_i) is feasible for (10) and has the same objective value, and the problems are equivalent.

Lower bound via relaxation. Since the perspective formulation is equivalent to the original problem, relaxing the Boolean constraint in (10) and solving the resulting convex optimization problem

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^m f_i^{\mathrm{p}}(z_i, t_i) + (1 - t_i)\alpha_i + \frac{1}{m}\left(f_0^{\mathrm{p}}(z_i, t_i) + f_0^{\mathrm{p}}(x - z_i, 1 - t_i)\right) \\ \text{subject to} & 0 \leq t \leq \mathbf{1}, \end{array} \tag{12}$$

with variables z_i, t, and x, yields a lower bound on the objective value of (1). That is, given any approximate solution of (1) with objective value p, the optimal value $q^\star$ of (12) yields a certificate guaranteeing that the approximate solution is suboptimal by at most $p - q^\star$. Additionally, a solution of the relaxed problem can be used as an initial point for any of the heuristic methods described in §3.

Efficiently solving the relaxed problem. We note that (12) has $m + 1$ times as many variables as the original problem, so it is worth considering faster solution methods. To do so, we can convert the problem to *consensus form* [2, §7.1]; *i.e.*, we introduce additional variables $y_i \in \mathbf{R}^n$ for $i = 1, \ldots, m$, and constrain $y_i = x$, resulting in the equivalent problem

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^m f_i^{\mathrm{p}}(z_i, t_i) + (1 - t_i)\alpha_i + \frac{1}{m}\left(f_0^{\mathrm{p}}(z_i, t_i) + f_0^{\mathrm{p}}(y_i - z_i, 1 - t_i)\right) \\ \text{subject to} & y_i = x, \quad i = 1, \ldots, m, \\ & 0 \leq t \leq \mathbf{1}. \end{array} \tag{13}$$

Since the objective is separable in (y_i, z_i, t_i) over i, there exist many efficient distributed algorithms for solving this problem, *e.g.*, the alternating direction method of multipliers (ADMM) [2, 5, 4].

5 Implementation

Our Python package `sccf` approximately solves generic problems of the form (1) provided all f_i can be represented as valid `cvxpy` expressions and constraints. We provide a method `sccf.minimum`, which can be applied to a `cvxpy` Expression and a scalar to create a `sccf.MinExpression`. The user then forms an objective as a sum of `sccf.MinExpression`s, passes this objective and (possibly) constraints to a `sccf.Problem` object, and then calls the `solve` method, which implements algorithm 3.1. We take advantage of the fact that the only parameter changing between problems is λ by caching the canonicalization procedure [1]. Here is an example of using `sccf` to solve a clipped least squares problem:

```
import cvxpy as cp
import sccf

A, b = get_data(m, n)

x = cp.Variable(n)
objective = 0.0
for i in range(m):
    objective += sccf.minimum(cp.square(A[i]@x-b[i]), 1.0)
objective += 0.01 * cp.sum_squares(x)

prob = sccf.Problem(objective)
prob.solve()
```

6 Examples

All experiments were conducted on a single core of an Intel i7-8700K CPU clocked at 3.7 GHz.

6.1 Clipped regression

In this example we compare clipped regression (§2.1) with standard linear regression and Huber regression [**?**] (a well known technique for robust regression) on a one-dimensional dataset with outliers. We generated data by sampling 20 data points (x_i, y_i) according to

$$x_i \sim \mathcal{N}(0,1), \quad y_i = x_i + (0.1)z_i, \quad z_i \sim \mathcal{N}(0,1), \quad i = 1, \ldots, 20.$$

We introduced outliers in our data by flipping the sign of y_i for 5 random data points.

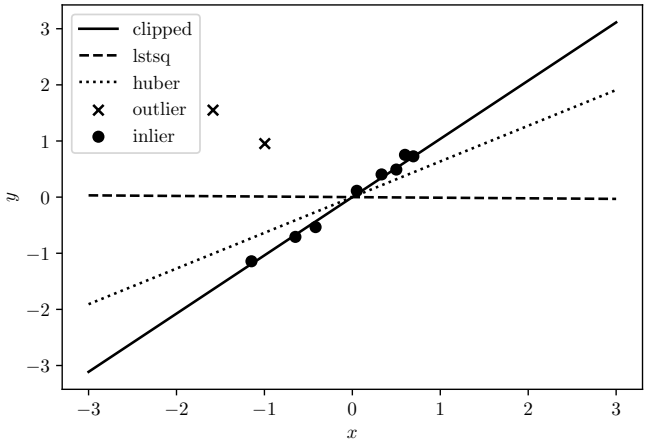

Figure 1: Clipped regression, linear regression, and Huber regression on a one-dimensional dataset with outliers. The outliers affect the linear regression and Huber regression models, while the clipped regression model appears to be minimally affected.

The problems all have the form

$$\text{minimize} \quad L(\theta) = \textstyle\sum_{i=1}^{20} \phi(x_i\theta - y_i) + (0.2)\theta^2, \tag{14}$$

where $\phi : \mathbf{R} \to \mathbf{R}$ is a penalty function. In clipped regression, $\phi(z) = \min\{z^2, 0.5\}$. In linear regression, $\phi(z) = z^2$. In Huber regression,

$$\phi(z) = \begin{cases} z^2 & |z| \leq 0.5 \\ 0.5(2|z| - 0.5) & \text{otherwise.} \end{cases}$$

Let θ^{clip} be the clipped regression model; we deem points where $(x_i\theta^{\text{clip}} - y_i)^2 \geq 0.5$ as outliers and the remaining points as inliers. In figure 1 we visualize the data points and the resulting models along with the outliers/inliers identified by the clipped regression model. In this figure, the clipped regression model clearly outperforms the linear and Huber regression models since it is able to fully ignore the outliers. Algorithm 3.1 terminated in 0.13 seconds and took 8 iterations on this instance.

Lower bound. The relaxed version of the perspective formulation (12) can be used to efficiently find a lower bound on the objective value for the clipped version of (14). The objective value of (14) for clipped regression was 1.147, while the lower bound we calculated was 0.533, meaning our approximate solution is suboptimal by at most 0.614.

In figure 2 we plot the clipped objective (14) for various values of θ; note that the function is highly nonconvex and that θ^{clip} is the (global) solution. We also plot the objective of the

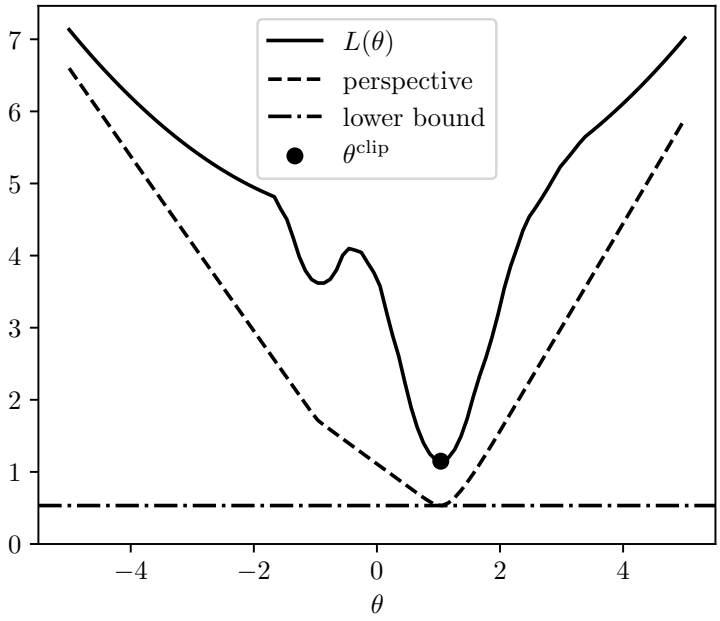

Figure 2: The clipped regression loss and its perspective relaxation.

perspective relaxation as a function of θ, found by partially minimizing (12) over z_i and t; note that the function is convex and a surprisingly good approximation of the true convex envelope. We note that the minimum of the perspective relaxation and the true minimum are surprisingly close, leading us to believe that the solution to the perspective relaxation could be a good initialization for heuristic methods.

6.2 Clipped logistic regression

In this example we apply clipped logistic regression (§2.1) to a dataset with outliers. We generated data by sampling 1000 data points (x_i, y_i) from a mixture of two Gaussian distributions in $\mathbf{R}^5$. We randomly partitioned the data into 100 training data points and 900 test data points and introduced outliers by flipping the sign of y_i for 20 random training data points.

We (approximately) solved the *clipped logistic regression* problem

$$\text{minimize} \quad \tfrac{1}{1000}\textstyle\sum_{i=1}^{1000} \min\{\log(1+e^{-y_i(x_i^T\theta+b)}), \alpha\} + (0.1)\|\theta\|_2^2,$$

with variables θ and b, for various values of $\alpha \in [10^{-1}, 10^1]$. We also solved the problem for $\alpha = +\infty$, *i.e.*, the *standard logistic regression problem.* Over the α values we tried, on average, algorithm 3.1 took 6.37 seconds and terminated in 9.64 iterations.

Figure 3 displays the test loss and fraction of outliers over the range of values of α we approximately minimized. Figure 4 shows the trajectory of the entries of λ during each step

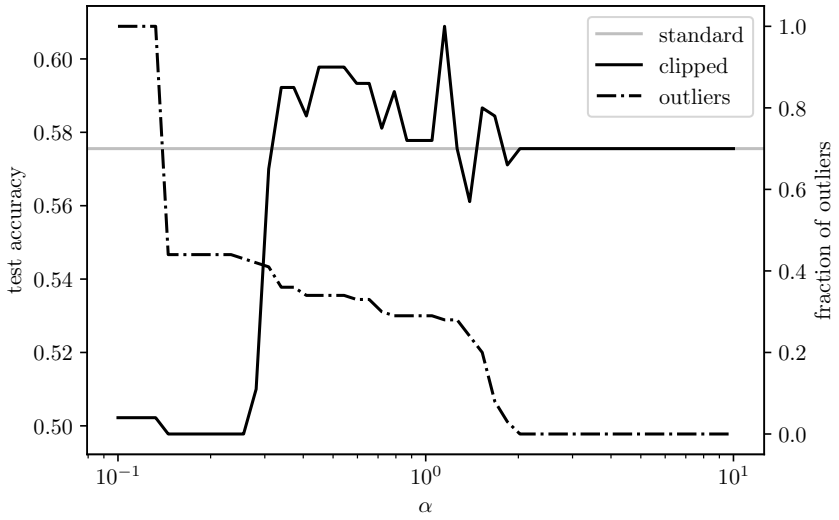

Figure 3: Test accuracy of clipped logistic regression (solid), test accuracy of standard logistic regression (gray), and fraction of outliers (dotted dashed) for varying clip values α. Note that the fraction of detected outliers goes down as α goes up. Between roughly $\alpha = 10^{-.5}$ and $\alpha = 10^{0.05}$, the test accuracy of clipped logistic regression is higher than standard logistic regression. Clipped logistic regression converges to standard logistic regression as $\alpha \to \infty$.

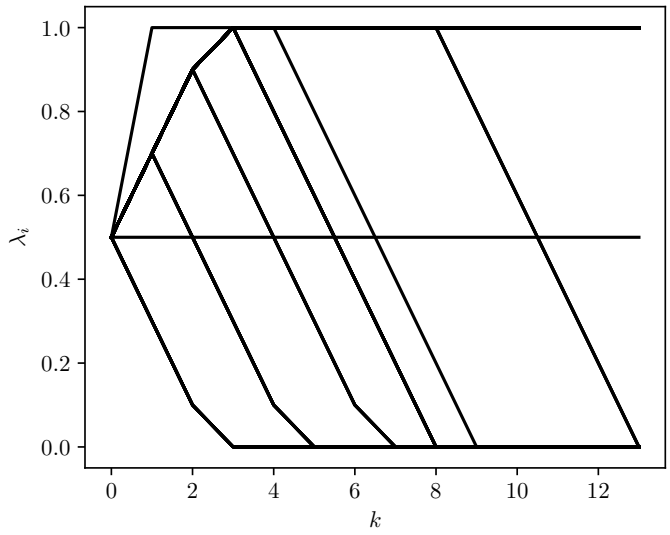

Figure 4: A plot of λ throughout the course of algorithm 3.1 for the clipped logistic regression example. Note that at some of the iterations (*e.g.*, $k = 1$, 2, or 3), the gradient of the loss with respect to a certain λ_i changes sign, causing λ_i to be updated in the opposite direction.

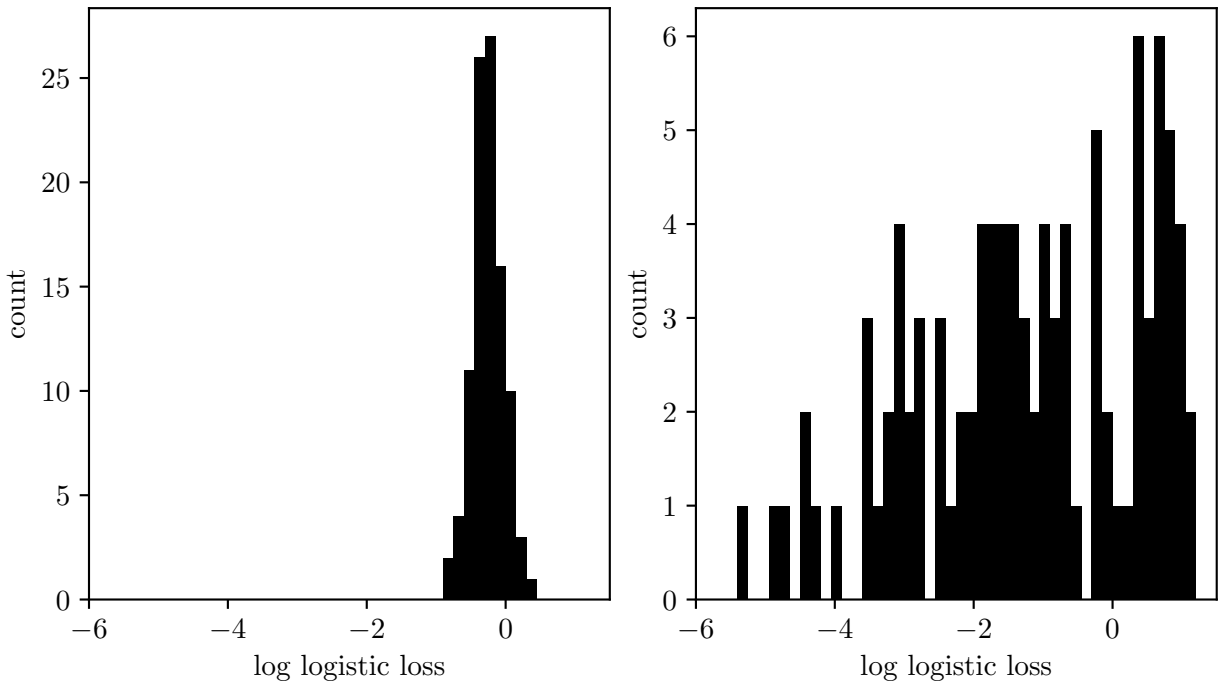

Figure 5: Left: histogram of log logistic loss for each data point in standard logistic regression; right: histogram of log logistic loss for each data point in clipped logistic regression. Note that standard logistic regression attempts to make the loss small for all data points, while its clipped counterpart allows the loss to be high for some of the data points.

of the execution of algorithm 3.1 for the α with the highest test accuracy, while figure 4 plots the histogram of the logistic loss for each of the available data points for this same α.

6.3 Lane changing

In this example, we consider a control problem where a vehicle traveling down a road at a fixed speed must avoid obstacles, stay in one of two lanes, and provide a comfortable ride. We let $x_t \in \mathbf{R}$ denote the lateral position of the vehicle at time $t = 0, \ldots, T$ (T is the time horizon).

The obstacle avoidance constraints are given as vectors $x^{\min}, x^{\max} \in \mathbf{R}^T$ that represent lower and upper bounds on x_t at time t.

We can split the objective into the sum of two functions described below.

- *Lane cost.* Suppose the two lanes are centered at $x = -1$ and $x = 1$. The lane cost is

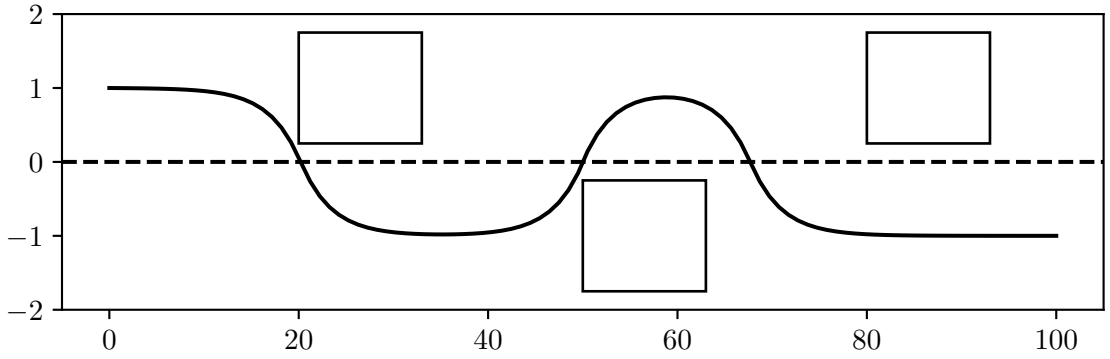

Figure 6: Trajectory of a vehicle looking to avoid obstacles (represented by boxes) while optimizing for comfort and lane position.

given by

$$g^{\text{lane}}(x) = \sum_{t=0}^{T} \min\{(x_t - 1)^2, 1\} + \min\{(x_t + 1)^2, 1\}.$$

The lane cost incentivizes the vehicle to be in the center of one of the two lanes. The lane cost is evidently a sum of clipped convex functions.

- *Comfort cost.* The comfort cost is given by

$$g^{\text{comfort}}(x) = \rho_1 \|Dx\|_2^2 + \rho_2 \|D^2 x\|_2^2 + \rho_3 \|D^3 x\|_2^2,$$

where D is the difference operator and $\rho_1, \rho_2, \rho_3 > 0$ are weights to be chosen. The comfort cost is a weighted sum of the squared lateral velocity, acceleration, and jerk.

To find the optimal lateral trajectory we solve the problem

$$\begin{array}{ll} \text{minimize} & g^{\text{lane}}(x) + g^{\text{comfort}}(x) \\ \text{subject to} & x_0 = x^{\text{start}}, \quad x_T = x^{\text{end}}, \\ & x^{\min} \leq x \leq x^{\max}, \end{array} \tag{15}$$

where $x^{\text{start}}, x^{\text{end}} \in \mathbf{R}$ are given starting and ending points of the trajectory.

Numerical example. We use $T = 100$, $\rho_1 = 10$, $\rho_2 = 1$, $\rho_3 = .1$, $x^{\text{start}} = 1$, and $x^{\text{end}} = -1$. In figure 6 we show the trajectory resulting from an approximate solution to (15) with three obstacles. For this example, algorithm 3.1 terminated in 1.2 seconds and took 4 iterations. We are able to find a comfortable trajectory that avoid the obstacles and spends as little time as possible in between the lanes.

Lower bound and global solution. Using the relaxed version of the perspective formulation (12), we can compute a lower bound on the objective value of the clipped control problem (15). We found a lower bound value of around 103.55, while the approximate solution we found had an objective value of 119.07, indicating that the solution we found was suboptimal by at most 15.52.

Acknowledgments

S. Barratt is supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1656518.

References

[1] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and Z. Kolter. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*, 2019.

[2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[3] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[4] D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, 2(1):17–40, 1976.

[5] R. Glowinski and A. Marroco. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires. *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique*, 9(R2):41–76, 1975.

[6] M. Grant and S. Boyd. Graph implementations for nonsmooth convex programs. In *Recent Advances in Learning and Control*, pages 95–110. Springer, 2008.

[7] P. Huber and E. Ronchetti. *Robust Statistics*. John Wiley & Sons, 2009.

[8] G. Lan, C. Hou, and D. Yi. Robust feature selection via simultaneous capped 2-norm and 2, 1-norm minimization. In *IEEE Intl. Conf. on Big Data Analysis (ICBDA)*, pages 1–5. IEEE, 2016.

[9] T. Lipp and S. Boyd. Variations and extension of the convex–concave procedure. *Optimization and Engineering*, 17(2):263–287, 2016.

[10] T. Liu and H. Jiang. Minimizing sum of truncated convex functions and its applications. *Journal of Computational and Graphical Statistics*, 28(1):1–10, 2019.

[11] N. Moehle and S. Boyd. A perspective–based convex relaxation for switched-affine optimal control. *Systems & Control Letters*, 86:34–40, 2015.

[12] C. Ong and L. An. Learning sparse classifiers with difference of convex functions algorithms. *Optimization Methods and Software*, 28(4):830–854, 2013.

[13] J. Portilla, A. Tristan-Vega, and I. Selesnick. Efficient and robust image restoration using multiple-feature l2-relaxed sparse analysis priors. *IEEE Transactions on Image Processing*, 24(12):5046–5059, 2015.

[14] T. Rockafellar. *Convex analysis*. Princeton University Press, 1970.

[15] A. Safari. An e–E–insensitive support vector regression machine. *Computational Statistics*, 29(6):1447–1468, 2014.

[16] Y. She and A. Owen. Outlier detection using nonconvex penalized regression. *Journal of the American Statistical Association*, 106(494):626–639, 2011.

[17] Q. Sun, S. Xiang, and J. Ye. Robust principal component analysis via capped norms. In *Proc. Intl. Conf. on Knowledge Discovery and Data Mining*, pages 311–319. ACM, 2013.

[18] S. Suzumura, K. Ogawa, M. Sugiyama, and I. Takeuchi. Outlier path: A homotopy algorithm for robust SVM. In *Intl. Conf. on Machine Learning*, pages 1098–1106, 2014.

[19] P. Tao and L. An. Convex analysis approach to DC programming: Theory, algorithms and applications. *Acta Mathematica Vietnamica*, 22(1):289–355, 1997.

[20] P. Torr and A. Zisserman. Robust computation and parametrization of multiple view relations. In *Intl. Conf. on Computer Vision*, pages 727–732. IEEE, 1998.

[21] G. Xu, B.-G. Hu, and J. Principe. Robust C-loss kernel classifiers. *IEEE Transactions on Neural Networks and Learning Systems*, 29(3):510–522, 2016.

[22] Y.-l. Yu, M. Yang, L. Xu, M. White, and D. Schuurmans. Relaxed clipping: A global training method for robust regression and classification. In *Advances in Neural Information Processing Systems*, pages 2532–2540, 2010.

[23] A. Yuille and A. Rangarajan. The concave–convex procedure. *Neural Computation*, 15(4):915–936, 2003.

[24] T. Zhang. Multi-stage convex relaxation for learning with sparse regularization. In *Advances in Neural Information Processing Systems*, pages 1929–1936, 2009.

[25] T. Zhang. Analysis of multi-stage convex relaxation for sparse regularization. *Journal of Machine Learning Research*, 11(Mar):1081–1107, 2010.

A Difference of convex formulation

In this section we make the observation that (1) can be expressed as a difference of convex (DC) programming problem.

Let $h_i(x) = \max(f_i(x) - \alpha_i, 0)$. This (convex) function measures how far $f_i(x)$ is above α_i. We can express the ith term in the sum as

$$\min\{f_i(x), \alpha_i\} = f_i(x) - h_i(x),$$

since when $f_i(x) \leq \alpha_i$, we have $h_i(x) = 0$, and when $f_i(x) > \alpha$, we have $h_i(x) = f_i(x) - \alpha_i$. Since f_i and h_i are convex, (1) can be expressed as the DC programming problem

$$\text{minimize} \quad f_0(x) + \textstyle\sum_{i=1}^m f_i(x) - \sum_{i=1}^m h_i(x), \tag{16}$$

with variable x. We can apply then well-known algorithms like the convex-concave procedure [19, 23] to (approximately) solve (16).

B Minimal convex extension

If we replace each f_i with any function $\tilde{f}_i$ such that $\tilde{f}_i(x) = f_i(x)$ when $f_i(x) \leq \alpha_i$, we get an equivalent problem. One such $\tilde{f}_i$ is the *minimal convex extension* of f_i, which is given by

$$\tilde{f}_i(x) \coloneqq \sup\{f_i(z) + g^T(x - z) \mid g \in \partial f_i(z), f_i(z) \leq \alpha_i, z \in \mathbf{R}^n\}.$$

In general, the minimal convex extension of a function is often hard to compute, but it can be represented analytically in some (important) special cases. For example, if $f_i(x) = (a^T x - b)^2$, the minimal convex extension is the Huber penalty function, or

$$\tilde{f}_i(x) = \begin{cases} (a^T x - b)^2 & |a^T x - b| \leq \alpha_i \\ \alpha_i(2|a^T x - b| - \alpha_i) & \text{otherwise.} \end{cases}$$

Using the minimal convex extension leads to an equivalent problem, but, depending on the algorithm, replacing f_i with $\tilde{f}_i$ can lead to better numerical performance.