Using Benson's Algorithm for Regularization Parameter Tracking

Joachim Giesen, Sören Laue, Andreas Löhne
Friedrich-Schiller-Universität Jena
Faculty of Mathematics and Computer Science
Ernst-Abbe-Platz 2
07743 Jena, Germany

Christopher Schneider
Ernst-Abbe-Hochschule Jena
Fachbereich Grundlagenwissenschaften
Carl-Zeiss-Promenade 2
07745 Jena, Germany

Abstract

Regularized loss minimization, where a statistical model is obtained from minimizing the sum of a loss function and weighted regularization terms, is still in widespread use in machine learning. The statistical performance of the resulting models depends on the choice of weights (regularization parameters) that are typically tuned by cross-validation. For finding the best regularization parameters, the regularized minimization problem needs to be solved for the whole parameter domain. A practically more feasible approach is covering the parameter domain with approximate solutions of the loss minimization problem for some prescribed approximation accuracy. The problem of computing such a covering is known as the approximate solution gamut problem. Existing algorithms for the solution gamut problem suffer from several problems. For instance, they require a grid on the parameter domain whose spacing is difficult to determine in practice, and they are not generic in the sense that they rely on problem specific plug-in functions. Here, we show that a well-known algorithm from vector optimization, namely the Benson algorithm, can be used directly for computing approximate solution gamuts while avoiding the problems of existing algorithms. Experiments for the Elastic Net on real world data sets demonstrate the effectiveness of Benson's algorithm for regularization parameter tracking.

1 Introduction

Regularized optimization problems of the form

$$\begin{array}{ll} \min\limits_{x\in\mathbb{R}^n} & \ell(x) + \sum_{i=1}^{q} \alpha_i r_i(x) \\ \text{s.t.} & g(x) \le 0 \end{array} \tag{P}$$

are still used extensively in the day-to-day practice of machine learning. Here, $\ell\colon \mathbb{R}^n \to \mathbb{R}$ is a *loss function*, the $r_i\colon \mathbb{R}^n \to \mathbb{R}$, $i = 1, \dots, q$ are *regularization terms*, and the $\alpha_i \ge 0$ are the corresponding *regularization parameters*. Sometimes, the problem is constrained, and here the constraints are given by a function $g\colon \mathbb{R}^n \to \mathbb{R}^m$.

In many machine learning applications Problem (P) is convex, i.e., all the functions ℓ, r_i, g are convex. The optimal solution x^α of Problem (P) describes some machine learning model, for instance the weights of a support vector machine, and depends on the regularization parameters $\alpha = (\alpha_1, \dots, \alpha_q)$. Thus it is important to choose *good* values for these parameters. The regularization parameters are typically optimized using some measure for the generalization error of the model on validation data, while x^α is computed from training data. Optimizing the regularization parameters is almost always a highly *non-convex* problem, even if Problem (P) is convex. This, essentially, leaves only searching for optimal parameters over the whole parameter domain. An exhaustive search would require computing x^α for all possible parameter values. The gamut of these solutions is called the *full solution gamut*. Since it is almost never tractable to compute the full solution gamut, approximation methods and heuristics are typically used in practice. In a search heuristic, the parameter domain is sampled with a finite number $\alpha^1, \dots, \alpha^k$ of parameter vectors, an optimal solution x^j, $j = 1, \dots, k$ is computed for all these vectors, and the best among them is chosen. A naive search approach, called *Grid Search*, samples the parameter domain in a structured way on a grid. As (Bergstra and Bengio 2012) have pointed out, it is highly beneficial to sample the parameter domain randomly in an unstructured fashion, if the different parameter dimensions are not equally important, see Figure 1. In this case, *Random Search* needs significantly fewer samples for achieving the same *approximation error*. The main disadvantage of both search approaches is

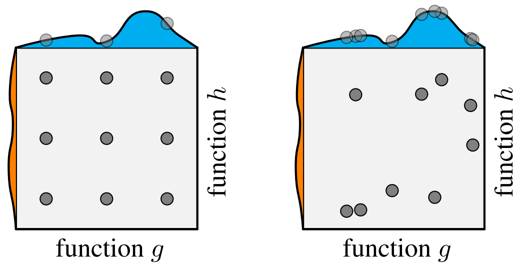


Figure 1: Optimization of $f(x, y) = g(x) + h(y) \approx g(x)$. Above each square, the function g is shown in blue, left to each square, h is shown in orange. With Grid Search (left), nine trial parameters only check three values of g. With Random Search (right), nine trial parameters check nine different values of the more important function g (cf. (Bergstra and Bengio 2012, Figure 1)).

that no guarantees on the approximation error can be given, i.e., there exists no *stopping rule* that determines a good grid spacing or number of random samples.

Related Work. The problem of missing approximation guarantees for the generic search methods led to a research focus on special instances of Problem (P) that we briefly review here. The special case of only one regularization term, i.e., the case $q = 1$, was first studied in the seminal work of (Efron et al. 2004) who observed that the full solution gamut of the Lasso is a piecewise linear function. The full solution gamut for problems with only one regularization term, i.e., the optimal solution x^α as a function of the single regularization parameter α, is traditionally called the regularization path. In (Rosset and Zhu 2007), a fairly general theory of piecewise linear regularization paths has been developed and exact path following algorithms have been devised. Important special cases are support vector machines whose regularization paths have been studied in (Hastie et al. 2004; Zhu et al. 2003), support vector regression (Wang, Yeung, and Lochovsky 2006), and the generalized Lasso (Tibshirani and Taylor 2011). Early on it was known, see for example (Allgower and Georg 1993; Bach, Thibaux, and Jordan 2004; Hastie et al. 2004), that exact regularization path following algorithms suffer from numerical instabilities as they repeatedly need to invert a matrix whose condition number can be poor, especially when using kernels. It also turned out (Gärtner, Jaggi, and Maria 2012; Mairal and Yu 2012) that the combinatorial (and thus also computational) complexity of exact regularization paths can be exponential in the number of data points. These shortcomings that also show up in practice sparked interest in the development of more robust and efficient approximate path algorithms (Friedman et al. 2007; Rosset 2004). By now numerically robust, approximate regularization path following algorithms are known for many problems including support vector machines (Giesen, Jaggi, and Laue 2012a; Giesen et al. 2012), the Lasso (Mairal and Yu 2012), and regularized matrix factorization and completion problems (Giesen, Jaggi, and Laue 2012b; Giesen et al. 2012). For a prescribed accuracy $\varepsilon > 0$, these algorithms compute a piecewise constant approximation of the solution path, which is called an *ε-approximate solution gamut*.

The idea of approximate solution gamuts was carried further and extended to higher dimensions, i.e., $q > 1$ in Problem (P), by (Blechschmidt, Giesen, and Laue 2015). The basic algorithmic idea of this work is computing a solution x^α to Problem (P) for some parameter vector α and determining the region in the parameter domain, where x^α is at least an ε-approximate solution. The algorithm then iterates over the complement of the union of the regions that are already covered by some ε-approximate solution. At every iteration, one element of the ε-approximate solution gamut is computed. The algorithm stops once the whole parameter domain is covered. Key features of this *Solution Gamut method* are a well-defined *stopping criterion* for a desired approximation guarantee, and its efficiency that results from *adapting* to the complexity of the solution space. Fewer solutions are computed in parameter regions where the solution does not change much. Unfortunately, the Solution Gamut method still needs a grid on the parameter domain. In contrast to Grid Search, here the grid is only used for testing whether a given solution is still an ε-approximate solution at the grid point. Therefore, it is enough to compute function values at the grid points which, in general, is much cheaper than solving optimization problems. For providing the guarantee that the whole parameter domain is covered by ε-approximate solutions, the grid spacing has to be chosen carefully. In theory, a sufficient spacing can be derived from smoothness properties, i.e., Lipschitz constants, of Problem (P), but in practice it is non-trivial to get the grid spacing right. Another drawback is that for checking if a solution is an ε-approximate solution, the algorithm needs to compute not only solutions to Problem (P), but also solutions to its Lagrangian dual problem. Furthermore, it requires problem-specific plug-in functions.

Contributions In this paper, we show that a variant of Benson's vector optimization algorithm (Benson 1998; Löhne, Rudloff, and Ulus 2014) naturally allows computing ε-approximate solution gamuts in a way that combines the advantages of previous approaches. Benson's algorithm comes with a clear *stopping criterion* and also *adapts* to the problem structure like the Solution Gamut method, but is *grid/lattice-free* in contrast to the latter. Furthermore, it works out of the box for all instances of the generic Problem (P) and does not rely on problem-specific plug-in functions. It is easy to *parallelize* (Bücker et al. 2018) and does not need to solve dual problems. We summarize the advantages of Benson's algorithm in Table 1 and compare it to the established approaches.

Table 1: Comparison of methods.

	Grid Search	Random Search	Solution Gamut	Benson Alg.
lattice-free	✗	✓	✗	✓
adaptive	✗	✓	✓	✓
parallelizable	✓	✓	✗	✓
stopping rule	✗	✗	✓	✓

Outline In the next section, we study the parameterized Problem (P) in a vector optimization setting, allowing us to apply a simplified variant of Benson's algorithm that we introduce in Secton 3. Thereafter, in Section 4, we report on experimental results for the Elastic Net on various real world data sets that corroborate the effectiveness of Benson's algorithm. Experimentally, our implementation of Benson's algorithm matches the asymptotic lower bound on the solution gamut complexity that has been proven by (Blechschmidt, Giesen, and Laue 2015). But our approach yields a much better constant. In fact, it needs to solve about an order of magnitude fewer optimization problems than the Solution Gamut method for the same approximation guarantees. Similar results also hold true for Ridge Regression and the Lasso that both feature only one regularization term.

2 The Solution Gamut and its Relation to Vector Optimization

We begin by recapitulating the definition of the *solution gamut* for the considered class of optimization problems and show its direct relation to the *solution concept* in the field of vector optimization

Solution Gamut. By re-scaling the objective function of Problem (P), we can assume that (P) is given as

$$\min_{x\in\mathcal{X}} \quad f_w(x) = w_0\,\ell(x) + \sum_{i=1}^{q} w_i\, r_i(x)\,, \tag{P$_w$}$$

with feasible set $\mathcal{X} = \{x \in \mathbb{R}^n \mid g(x) \le 0\}$, and regularization parameters $w = (w_0, w_1, \ldots, w_q) \in \mathcal{S} \subseteq \mathbb{R}^{q+1}$, where

$$\mathcal{S} = \left\{ w \in \mathbb{R}^{q+1} \;\middle|\; w_i \ge 0 \text{ for all } i\,,\ \textstyle\sum_{i=0}^{q} w_i = 1 \right\}$$

defines the q-dimensional standard simplex. For a given parameter $w \in \mathcal{S}$, we denote by x^w a global optimal solution to Problem (P$_w$).

Definition 2.1. Let $\varepsilon > 0$ be given. We call some function $\bar{x}\colon \mathcal{S} \to \mathbb{R}^n$ an *ε-approximative solution gamut* of Problem (P$_w$), if for all $w \in \mathcal{S}$

$$g(\bar{x}(w)) \le 0 \quad \text{and} \quad f_w(\bar{x}(w)) - f_w(x^w) \le \varepsilon\,. \qquad \diamond$$

Remark 2.2. The ε-approximative solution gamut in (Blechschmidt, Giesen, and Laue 2015) is piecewise constant. $\diamond$

Remark 2.3. For $\varepsilon = 0$, Definition 2.1 yields the so-called *full solution gamut*. Notice that the computation of full solution gamuts is only possible for special cases (e.g. LARS algorithm for the Lasso (Efron et al. 2004)). $\diamond$

Vector Optimization. For our approach, we consider the convex vector optimization problem related to (P$_w$), namely

$$\begin{array}{ll} \min\limits_{x\in\mathcal{X}} & F(x) = \Big[\ell(x), r_1(x), \ldots, r_q(x)\Big]^{\mathsf{T}} \\ \text{w.r.t.} & \le_{\mathbb{R}^{q+1}_+} \end{array} \tag{VP}$$

whose objective function $F\colon \mathbb{R}^n \to \mathbb{R}^{q+1}$ is vector-valued and minimized w.r.t. the component-wise partial ordering $\le_{\mathbb{R}^{q+1}_+}$ on $\mathbb{R}^{q+1}$:

$$y_1 \le_{\mathbb{R}^{q+1}_+} y_2 \quad \text{if and only if} \quad y_2 - y_1 \in \mathbb{R}^{q+1}_+\,,$$

where $\mathbb{R}^{q+1}_+ = \{y \in \mathbb{R}^{q+1} \mid y_i \ge 0,\ i = 1, \ldots, q+1\}$. In the following, the feasible set $\mathcal{X}$ of (VP) is assumed to be non-empty.

Connection to the Solution Gamut Problem. Our main contribution is the observation that the full solution gamut of Problem (P$_w$) is the set of so-called *weak minimizers* of Problem (VP).

Definition 2.4. A point $x^* \in \mathcal{X}$ is called a *weak minimizer* of Problem (VP) if $(\{F(x^*)\} - \operatorname{int}\mathbb{R}^{q+1}_+) \cap F(\mathcal{X}) = \emptyset$, where $F(\mathcal{X}) = \{F(x) \in \mathbb{R}^{q+1} \mid x \in \mathcal{X}\}$ is the image of the feasible set. $\diamond$

The connection between the full solution gamut and the set of weak minimizers becomes intuitively clear, if one considers the geometry of the problems. The *upper image* of Problem (VP) is the set

$$\mathcal{P} = \operatorname{closure}(F(\mathcal{X}) + \mathbb{R}^{q+1}_+)\,.$$

An optimal solution to Problem (P$_w$) with parameters $w \in \mathcal{S}$ is then just a weak minimizer of Problem (VP) and vice versa (see e.g., (Jahn 2011)). Let x^w be an optimal solution of Problem (P$_w$) for parameters $w \in \mathcal{S}$, then $F(x^w)$ is a boundary point of the convex and closed set $\mathcal{P}$ and there exists a supporting hyperplane in $F(x^w)$ with normal vector w (see also Figure 3 (left)). Conversely, the image of any weak minimizer of Problem (VP) is a boundary point of $\mathcal{P}$ with some normal vector $w \in \mathcal{S}$.

Approximate Solutions. Motivated by our application in regularization parameter tracking, we are aiming at a finite representation of the full solution gamut of Problem (P$_w$) or, equivalently, the set of weak minimizers of Problem (VP). Finite representations are possible for special cases, see (Hamel, Löhne, and Rudloff 2014), but not for a general convex vector optimization problem. Therefore, we will consider ε-approximate solutions.

Definition 2.5. Let $c \in \operatorname{int}\mathbb{R}^{q+1}_+$ be arbitrary but fixed and assume that Problem (VP) is bounded, i.e., $\mathcal{P} \subseteq \{y\} + \mathbb{R}^{q+1}_+$ holds for some $y \in \mathbb{R}^{q+1}$. Then a nonempty, finite set $\mathcal{X}^* \subseteq \mathcal{X}$ is called an *ε-infimizer* if

$$\operatorname{conv} F(\mathcal{X}^*) + \mathbb{R}^{q+1}_+ - \varepsilon\, c \supseteq \mathcal{P}\,.$$

An ε-infimizer $\mathcal{X}^*$ of (VP) is called a *weak ε-solution* to (VP) if it only consists of weak minimizers. $\diamond$

An ε-solution $\mathcal{X}^*$ provides both, an inner and an outer polyhedral approximation of the upper image $\mathcal{P}$ by finitely many minimizers. Setting $\mathcal{P}_\varepsilon = \operatorname{conv} F(\mathcal{X}^*) + \mathbb{R}^{q+1}_+$, we have

$$\mathcal{P}_\varepsilon - \varepsilon\, c \supseteq \mathcal{P} \supseteq \mathcal{P}_\varepsilon\,.$$

Figure 2 illustrates these inclusions.

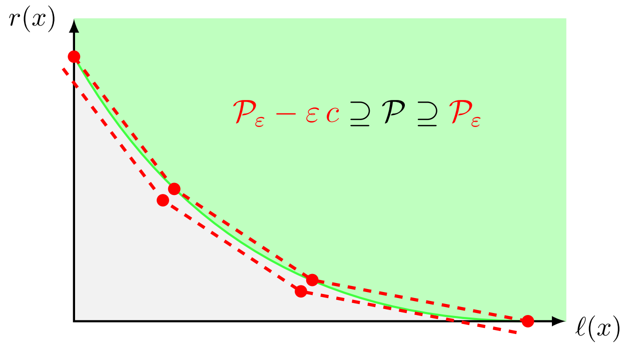

$r(x)$
$\mathcal{P}_\varepsilon - \varepsilon\, c \supseteq \mathcal{P} \supseteq \mathcal{P}_\varepsilon$
$\ell(x)$

Figure 2: Approximate Solutions.

In the following, we focus on computing ε-solutions of Problem (VP) for a prescribed accuracy $\varepsilon > 0$. By similar arguments as for the relation between weak minimizers and the full solution gamut, such an ε-solution directly provides an ε-approximative solution gamut for Problem (P$_w$).

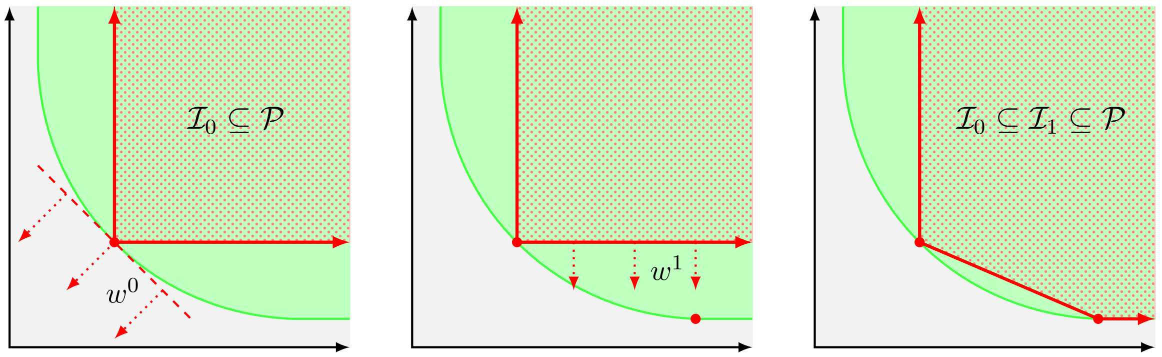

$\mathcal{I}_0 \subseteq \mathcal{P}$
w^0
w^1
$\mathcal{I}_0 \subseteq \mathcal{I}_1 \subseteq \mathcal{P}$

Figure 3: **Left:** Initialization of Algorithm 1 with weighted-sum scalarization (P_w) for $w = w^0$, where the dashed red line is defined by $\{y \mid w^\top y = F(x^w)\}$. **Middle and right:** First iteration of Algorithm 1 with regularization parameter $w = w^1$.

3 Benson's Algorithm

In this section, we present an algorithm for solving Problem (VP), whose initial ideas go back to (Benson 1998). Since the solution gamut problem does not require the most general formulation of Problem (VP), we will adapt the ideas behind Benson's algorithm and end up with a particularly simple and easy to implement variant.

Remark 3.1. Benson's algorithm essentially exists in two variants—a primal and a dual formulation. While in the original work (Benson 1998), the primal algorithm has been derived, we will use and refine a dual scheme, whose idea goes back to (Ehrgott, Löhne, and Shao 2012). More details for primal and dual variants can be found, e.g., in (Hamel, Löhne, and Rudloff 2014) and (Löhne, Rudloff, and Ulus 2014). $\diamond$

Benson's algorithm is especially well-suited for problems with $q \ll n$, i.e., like in our context, when there are many more variables than regularization terms, because it operates in the image space of the problem. The algorithm is iterative and, under the following conditions, returns a weak ε-solution on termination—cf. (Löhne, Rudloff, and Ulus 2014, Theorems 4.9 and 4.14): (i) Problem (P_w) has an optimal solution for all $w \in \mathcal{S}$ and (ii) the feasible set $\mathcal{X}$ has a non-empty interior (Slater's condition). These conditions clearly are fulfilled for the problems we consider in the following, where all functions ℓ, r_i are defined by norms (coercive and continuous) and $\mathcal{X} = \mathbb{R}^{q+1}_+$.

Since for the algorithm polyhedra play a crucial role, remember that each nonempty convex, polyhedral set $A \subseteq \mathbb{R}^{q+1}$ either can be defined as the intersection of finitely many half-spaces, i.e.,

$$A = \bigcap_{i=1}^{r} \{y \in \mathbb{R}^{q+1} \mid (w^i)^\top y \geq b_i\} \qquad (1)$$

for some $r \in \mathbb{N}$, $w^i \in \mathbb{R}^{q+1} \setminus \{0\}$, and $b_i \in \mathbb{R}$, or by

$$A = \operatorname{conv}\{v^1, \dots, v^s\} + \operatorname{cone}\{d^1, \dots, d^t\}\,, \qquad (2)$$

with $s \in \mathbb{N} \setminus \{0\}$, $t \in \mathbb{N}$, points $v^i \in \mathbb{R}^{q+1}$, and directions $d^j \in \mathbb{R}^{q+1} \setminus \{0\}$. We call (1) an *H-representation* and (2) a *V-representation* of A, respectively.

The Algorithm. The now discussed (dual) variant of Benson's algorithm approximates the upper image $\mathcal{P}$ of Problem (VP) by computing a growing sequence of *inner approximation polyhedra* $\mathcal{I}_j = \{y \in \mathbb{R}^{q+1} \mid (w^j)^\top y \geq b_j\}$. After an initial inner approximation $\mathcal{I}_0$ of $\mathcal{P}$ has been computed, the algorithm iteratively improves this approximation such that

$$\mathcal{I}_0 \subseteq \mathcal{I}_1 \subseteq \ldots \subseteq \mathcal{I}_j \subseteq \ldots \subseteq \mathcal{P}\,.$$

Thereby, the algorithm relies on *scalarizations* of Problem (VP), where the vector optimization problem is replaced by a suitable scalar optimization problem. This will be done by using the so-called *weighted-sum scalarization* (P_w) (see Section 2) with good choices for the regularization parameter w. How those w are chosen is described in Algorithm 1. An illustration of the initialization and the first iteration of the algorithm is given by Figure 3.

Algorithm 1 Benson Algorithm

```
    Input: Problem data $(F, g)$, direction $c$, accuracy $\varepsilon$
    Output: V-rep. $\mathcal{I}_{\mathrm{poi}}$ and H-rep. $\mathcal{I}$ of $\mathcal{P}_\varepsilon$

 1: function BENSONALGORITHM
 2:     $T \leftarrow \emptyset$
 3:     $w^0 \leftarrow (\frac{1}{q+1}, \dots, \frac{1}{q+1})^\top$
 4:     $x^0 \leftarrow \arg\min(P_{w^0})$
 5:     $\mathcal{I}_{\mathrm{poi}} \leftarrow \{F(x^0)\}$
 6:     compute H-representation $\mathcal{I}$ of $\mathcal{I}_{\mathrm{poi}}$
 7:     repeat
 8:         choose $w \in \mathcal{I} \setminus T$
 9:         $x^w \leftarrow \arg\min(P_w)$
10:         compute $d_c(x^w)$ as in Equation (3)
11:         if $d_c(x^w) > \varepsilon$ then
12:             $\mathcal{I}_{\mathrm{poi}} \leftarrow \mathcal{I}_{\mathrm{poi}} \cup \{F(x^w)\}$
13:             update the H-representation $\mathcal{I}$ of $\mathcal{I}_{\mathrm{poi}}$
14:         else
15:             $T \leftarrow T \cup \{w\}$
16:         end if
17:     until $\mathcal{I} \setminus T = \emptyset$
18: end function
```

Table 2: Comparison of the minimum test MSE and number of solved optimization problems for the considered methods.

Data Set	**Grid Search**		**Benson Algorithm**		**Random Search**		**Solution Gamut**	
	# Opt.	MSE	# Opt.	MSE	# Opt.	MSE	# Opt.	MSE
`ALLAML`	5151	0.0530	26	0.0544	168	0.0557	1141	0.0546
`arcene`	5151	0.6385	53	0.6460	71	0.7126	180	0.6565
`GLI-85`	5151	0.1030	55	0.1041	70	0.1057	562	0.1051
`GLIOMA`	5151	0.1919	57	0.1943	125	0.1992	575	0.1964
`Prostate-GE`	5151	0.0698	39	0.0701	2736	0.0708	1026	0.0702
`SMK-CAN-187`	5151	0.1600	51	0.1628	290	0.1668	128	0.1673
`Carcinom`	5151	1.8218	44	1.8543	155	1.8770	550	1.8661
`14cancer`	5151	7.8481	68	7.8725	61	7.8992	309	7.8763

Algorithm 1 consists of two crucial phases, the initialization and the main iteration, which we now describe in more detail.

Initialization. For computing an initial inner approximation $\mathcal{I}_0$ of the upper image $\mathcal{P}$, one can use the weighted-sum scalarization with weights $w^0 = (\frac{1}{q+1}, \dots, \frac{1}{q+1})^\mathsf{T} \in \mathbb{R}^{q+1}$. Problem (P$_w$) then returns a solution x^0, whose image point $F(x^0) \in \partial\mathcal{P}$ lies on the boundary of the upper image. We set

$$\mathcal{I}_0 = \{F(x^0)\} + \mathbb{R}^{q+1} .$$

Figure 3 (left) illustrates this process.

Benson Iteration. At the beginning of each iteration, $\mathcal{I}_j$ is given by a V-representation and has to be converted into an H-representation. This is done by facet enumeration, see (Bremner, Fukuda, and Marzetta 1998). Then, we select one hyperplane $\{y \in \mathbb{R}^{q+1} \mid w^\mathsf{T} y = b\}$ of the current approximation $\mathcal{I}_j$, represented by its normal vector w, and solve the scalarized problem (P$_w$). The distance, the hyperplane is moved in w-direction, is just $b - w^\mathsf{T} F(x^w)$. But since we compute an ε-approximation w.r.t. the direction c, we have to take the angle between w and c into account. Therefore, the distance, the hyperplane is moved in c-direction, is given by

$$d_c(x^w) = \frac{c^\mathsf{T} w}{\|c\|_2 \, \|w\|_2} \left(b - w^\mathsf{T} F(x^w)\right) . \tag{3}$$

If $d_c(x^w) > \varepsilon$, we add the optimal solution x^w of (P$_w$) to our inner approximation and obtain

$$\mathcal{I}_{j+1} = \operatorname{conv}\left(\mathcal{I}_j \cup F(x^w)\right) + \mathbb{R}^{q+1}_+ .$$

If otherwise $d_c(x^w) \leq \varepsilon$, we continue with checking the next hyperplane of the inner approximation.

Remark 3.2 (Outer Approximation). By saving the moved hyperplanes of each iteration, it would also be possible to generate an outer approximation of the upper image $\mathcal{P}$ without further costs—cf. (Löhne, Rudloff, and Ulus 2014, Remark 4.3(2)). But since we aim for feasible solutions, this step is omitted here. $\diamond$

4 Experiments

In our implementation of Algorithm 1, we used Gurobi (Gurobi Optimization 2016) for solving the scalarized problems (P$_w$). Facet enumeration is done by *bensolve tools* (Ciripoi, Löhne, and Weißing 2018). We fixed the direction parameter $c = (1, \dots, 1)^\mathsf{T}$.

The Elastic Net. We apply Benson's algorithm to the problem of linear regression, where we consider the Elastic Net regularization (Zou and Hastie 2005). The corresponding optimization problem reads as

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} \|Ax - b\|_2^2 + \beta \|x\|_1 + \frac{\alpha}{2} \|x\|_2^2 , \tag{EN}$$

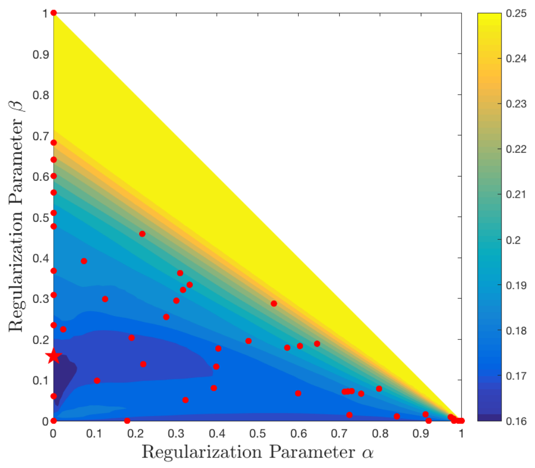
where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and the regularization parameters are $\alpha, \beta \in \mathbb{R}_+$. Thus, we have $q = 2$.

Remark 4.1. Setting $\beta = 0$ in Problem (EN) yields *Ridge regression*, while $\alpha = 0$ results in *Lasso regression*. Both scenarios have $q = 1$ regularization parameter and are later used to study the complexity of Benson's algorithm. $\diamond$

Data Sets. A common application for the Elastic net is *microarray classification and gene selection* (Zou and Hastie 2005, Section 6). In typical microarray data sets, there are thousands of genes but only a few samples. The Elastic Net turned out to be well-suited for such datasets since it combines the advantages of *variable selection* (L^1-norm) and *grouped selection* (L^2-norm), especially for the $m \gg n$ case.

For our experiments, we use the following data sets, which are well-known from the literature:

- `ALLAML` with $m =$ 7,129 features and $n = 72$ instances,
- `arcene` with $m =$ 10,000 and $n = 200$,
- `GLI-85` with $m =$ 22,283 and $n = 85$,
- `GLIOMA` with $m =$ 4,434 and $n = 50$,
- `Prostate-GE` with $m =$ 5,966 and $n = 102$,
- `SMK-CAN-187` with $m =$ 19,993 and $n = 187$,
- `Carcinom` with $m =$ 9,182 and $n = 174$,
- `14cancer` with $m =$ 16,063 and $n = 198$.



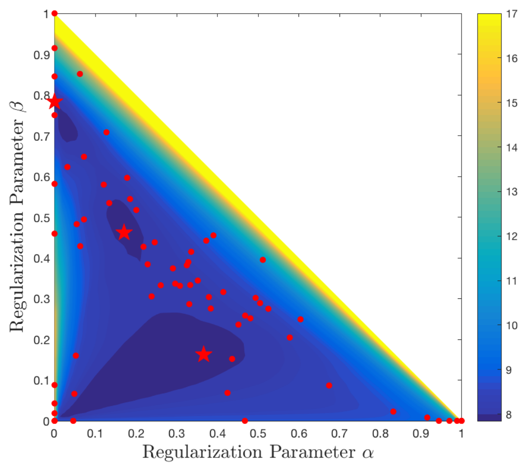


Figure 4: Shown are the test MSE values for the `SMK-CAN-187` (left) and `14cancer` (right) data sets. The results of the fine-mesh Grid Search are depicted by colors. The iterates of Benson's algorithm are illustrated by red points, and red stars indicate the best resp. the three best parameter combinations. All of them are located in the dark blue areas of low MSE.

Data Preparation. For all data sets, 70% of the data have been used for training, and 30% have been hold out for testing. We localize and standardize the training data, such that the response is centered and the predictors are standardized:

$$\sum_{i=1}^{m_{\text{train}}} b_i^{\text{train}} = 0\,, \quad \sum_{i=1}^{m_{\text{train}}} A_{i,j}^{\text{train}} = 0\,, \quad \sum_{i=1}^{m_{\text{train}}} \left(A_{i,j}^{\text{train}}\right)^2 = 1\,,$$

for $j = 1, \ldots, n$. We then re-use the training parameters (mean and standard derivation) to scale the test data sets, since they should be treated as new, unseen data.

Experimental Setup. Our aim is finding regularization parameters α, β for Problem (EN), such that its optimal solution $x^{\alpha,\beta}$ yields a low mean squared error (MSE) on the test data set. For this purpose, we compare Benson's algorithm with Grid Search, Random Search, and the Solution Gamut method of (Blechschmidt, Giesen, and Laue 2015). We proceed as follows: First, we re-scale Problem (EN) such that $\alpha, \beta \in [0,1]$ (compare Section 2),

$$\min_{x\in\mathbb{R}^n} \frac{1-\alpha-\beta}{2}\,\|Ax-b\|_2^2 + \beta\,\|x\|_1 + \frac{\alpha}{2}\,\|x\|_2^2\,. \quad (\text{EN}')$$

Second, we compute a fine-mesh solution with Grid Search by solving (EN′) for all $\alpha, \beta \in \{0, 0.01, 0.02, \ldots, 1\}$. The minimum test MSE for the fine-mesh solution is used as a baseline. We then run Benson's algorithm with approximation errors $\varepsilon = 0.1$ (for the first six data sets) and $\varepsilon = 1$ (for the last two data sets), resp., depending on the scale of the objective function values. After reporting the resulting minimum MSE, we run Random Search (averaged over 10 runs for each data set) and the Solution Gamut method until they reach at least the same MSE as Benson's algorithm. We report the minimum MSE for both methods that has been obtained until this iteration.

Remark 4.2. We only use the fine-mesh Grid Search for generating a baseline. Coarse grids are omitted in the comparison, since, in general, Random Search is superior to Grid Search (cf. (Bergstra and Bengio 2012)). ◇

Remark 4.3. Notice that Random Search and the Solution Gamut method get a big advantage by knowing the test MSE to be reached. Especially Random Search would not have a useful stopping criterion otherwise, in contrast to the Solution Gamut method and Benson's algorithm. ◇

Results. Table 2 reports the results of our experiments. As can be seen, the minimum MSE computed by Benson's algorithm reaches the fine-mesh Grid Search MSE up to 0.3–2.6% (on average 1.3%) by only having solved approx. 1/100 of the number of optimization problems. On most examples, Random Search also performs well, but overall loses to Benson's algorithm. The biggest limitation of Random Search is the missing stopping criterion.

It also turns out that Benson's algorithm is the clear winner over the Solution Gamut method. On the average of the problem instances that we considered, Benson's algorithm outperforms the Solution Gamut method by an order of magnitude in the number of optimization problems to be solved. Both approaches rely on a similar idea by exploiting the structure of the underlying optimization problem to generate good predictions. But due to the vector optimization setup, Benson's algorithm does not need a grid for the regularization parameter space which ends up as a great benefit.

Figure 6 is an example, how Benson's algorithm approximates the upper image of the corresponding vector optimization problem. Among all vertices of the approximation, the one with the best test MSE is chosen—see Figure 4 (right).

Case Study `SMK-CAN-187`. Figure 4 (left) shows the test MSE represented by colors for the `SMK-CAN-187` data set for all feasible combinations of $\alpha, \beta \in [0,1]$ on the fine-mesh grid computed by Grid Search. The red points indicate the chosen parameter combinations of Benson's algorithm, while the red star gives the best test MSE among them. This example nicely shows that Benson's algorithm tends to choose more parameter combinations (α, β) in areas, where the value of the MSE changes, and less combi-

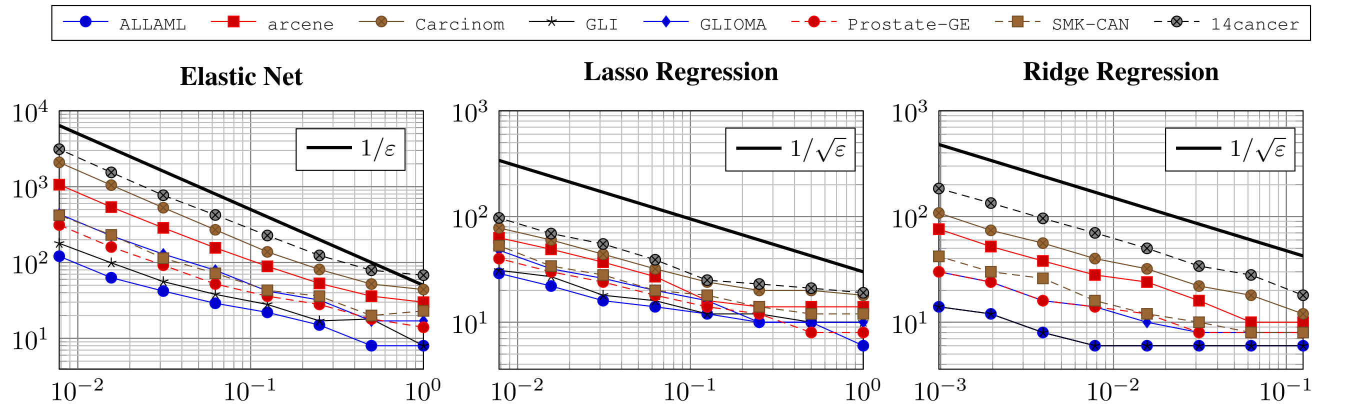


Figure 5: Log-log complexity plots of Alg. 1 for the Elastic Net (left), Lasso regression (middle) and Ridge regression (right).

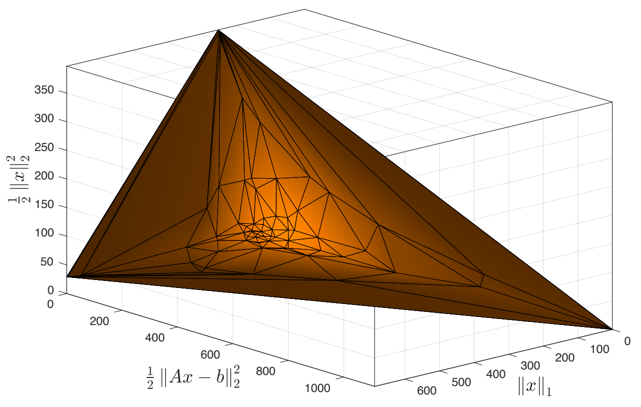


Figure 6: Approximation of the upper image $\mathcal{P}$ for the 14cancer data set using Algorithm 1 with $\varepsilon = 1$.

nations in "flat" areas, to obtain an overall good approximation. I.e., it only generates one weight combination for the big yellow area, where the predictor is zero, because the regularization terms are too dominant in Problem (EN′). Grid Search almost uses one third of its iterations to explore this area. This case is typical for most of the considered data sets.

Case Study 14cancer. Figure 4 (right) shows the test MSE for the 14cancer data set. Here, we have a good example that finding the best predictor among all solutions of the regularized optimization problem is itself a *non-convex* problem. There are three non-connected, dark blue areas, where certain parameter combinations (α, β) yield predictors with good test MSE. Benson's algorithm computes an ε-approximate solution for this data set in 68 iterations (red points) and thereby is able to find a representative predictor for each dark blue area (red stars). All of these three predictors yield a test MSE below 7.9500, while having different sparsity properties. From bottom to top, the three best computed predictors have 1814, 445, and 79 non-zeros, resp. Therefore, the user is able to decide between three similarly good candidates of very different sparsity.

While this is the only example, where Random Search needed slightly less iterations than Benson's algorithm, it was only able to find a good predictor from one of the three interesting areas.

Complexity. In (Blechschmidt, Giesen, and Laue 2015), a lower bound of $\Omega(\varepsilon^{-q/2})$ for the number of optimization problems that need to be solved for an ε-approximate solution gamut has been proven. For the Elastic Net (EN), we have $q = 2$ and thus a lower bound of $\Omega(1/\varepsilon)$ of necessary optimization problems. In Figure 5 (left), one can see that Benson's algorithm experimentally matches this lower bound.

Additionally, we carried out experiments for the Lasso and Ridge regression with $q = 1$, implying a lower bound of $\Omega(1/\sqrt{\varepsilon})$. Benson's algorithm also matches this bound—see Figure 5 (middle and right).

5 Conclusion

We have shown that Benson's vector optimization algorithm can be used for regularization parameter tracking with prescribed approximation guarantees. The fairly simple algorithm that works out of the box and does not rely on problem specific functions combines the advantages of previously known regularization parameter tracking methods: (i) it has a well-defined stopping criterion for a prescribed approximation guarantee, and (ii) it adapts to the problem structure and works grid-free, which entails fewer optimization problems to be solved.

We demonstrated these advantages for the Elastic Net problem on several real world data sets. It turned out that Benson's algorithm experimentally matches the theoretically known asymptotic lower bound on the number of optimization problems that need to be solved for a prescribed approximation guarantee. It needs to solve about an order of magnitude fewer optimization problems than the Solution Gamut method, the only other known algorithm with well-defined stopping criterion. On average, it needs to solve three times fewer optimization problems than the simple Random Search.

Acknowledgments

Joachim Giesen and Sören Laue acknowledge funding by Deutsche Forschungsgemeinschaft (DFG) under the grant GI 711/3-2. Sören Laue has been funded by Deutsche Forschungsgemeinschaft (DFG) under grant LA 2971/1-1. Christopher Schneider acknowledges financial support by Carl-Zeiss-Stiftung.

References

Allgower, E., and Georg, K. 1993. Continuation and path following. *Acta Numerica* 2:1–64.

Bach, F. R.; Thibaux, R.; and Jordan, M. I. 2004. Computing regularization paths for learning multiple kernels. In *Advances in Neural Information Processing Systems (NIPS)*.

Benson, H. P. 1998. An Outer Approximation Algorithm for Generating All Efficient Extreme Points in the Outcome Set of a Multiple Objective Linear Programming Problem. *Journal of Global Optimization* 13(1):1–24.

Bergstra, J., and Bengio, Y. 2012. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research* 13(1):281–305.

Blechschmidt, K.; Giesen, J.; and Laue, S. 2015. Tracking approximate solutions of parameterized optimization problems over multi-dimensional (hyper-)parameter domains. In *International Conference on Machine Learning (ICML)*, 438–447.

Bremner, D.; Fukuda, K.; and Marzetta, A. 1998. Primal-Dual Methods for Vertex and Facet Enumeration. *Discrete & Computational Geometry* 20(3):333–357.

Bücker, H.; Löhne, A.; ing, B. W.; and Zumbusch, G. 2018. On Parallelizing Benson's Algorithm: Limits and Opportunities. In *International Conference on Computational Science and Its Applications*, 653–668.

Ciripoi, D.; Löhne, A.; and Weißing, B. 2018. Calculus of convex polyhedra and polyhedral convex functions by utilizing a multiple objective linear programming solver. *Optimization*.

Efron, B.; Hastie, T.; Johnstone, I.; and Tibshirani, R. 2004. Least angle regression. *The Annals of Statistics* 32(2):407–499.

Ehrgott, M.; Löhne, A.; and Shao, L. 2012. A dual variant of Benson's "outer approximation algorithm" for multiple objective linear programming. *Journal of Global Optimization* 52(4):757–778.

Friedman, J.; Hastie, T.; Höfling, H.; and Tibshirani, R. 2007. Pathwise Coordinate Optimization. *The Annals of Applied Statistics* 1(2):302–332.

Gärtner, B.; Jaggi, M.; and Maria, C. 2012. An Exponential Lower Bound on the Complexity of Regularization Paths. *Journal of Computational Geometry (JoCG)* 3(1):168–195.

Giesen, J.; Müller, J. K.; Laue, S.; and Swiercy, S. 2012. Approximating Concavely Parameterized Optimization Problems. In *Advances in Neural Information Processing Systems (NIPS)*, 2114–2122.

Giesen, J.; Jaggi, M.; and Laue, S. 2012a. Approximating parameterized convex optimization problems. *ACM Transactions on Algorithms* 9(1):10.

Giesen, J.; Jaggi, M.; and Laue, S. 2012b. Regularization Paths with Guarantees for Convex Semidefinite Optimization. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 432–439.

Gurobi Optimization, I. 2016. Gurobi optimizer reference manual.

Hamel, A. H.; Löhne, A.; and Rudloff, B. 2014. Benson type algorithms for linear vector optimization and applications. *Journal of Global Optimization* 59(4):811–836.

Hastie, T.; Rosset, S.; Tibshirani, R.; and Zhu, J. 2004. The Entire Regularization Path for the Support Vector Machine. In *Advances in Neural Information Processing Systems (NIPS)*.

Jahn, J. 2011. *Vector Optimization: Theory, Applications, and Extensions*. Springer, second edition.

Löhne, A.; Rudloff, B.; and Ulus, F. 2014. Primal and dual approximation algorithms for convex vector optimization problems. *Journal of Global Optimization* 60(4):713–736.

Mairal, J., and Yu, B. 2012. Complexity analysis of the lasso regularization path. In *International Conference on Machine Learning (ICML)*.

Rosset, S., and Zhu, J. 2007. Piecewise linear regularized solution paths. *The Annals of Statistics* 35(3):1012–1030.

Rosset, S. 2004. Following curved regularized optimization solution paths. In *Advances in Neural Information Processing Systems (NIPS)*.

Tibshirani, R., and Taylor, J. 2011. The solution path of the generalized lasso. *The Annals of Statistics* 39(3):1335–1371.

Wang, G.; Yeung, D.-Y.; and Lochovsky, F. H. 2006. Two-dimensional solution path for support vector regression. In *International Conference on Machine Learning (ICML)*, 993–1000.

Zhu, J.; Rosset, S.; Hastie, T.; and Tibshirani, R. 2003. 1-norm Support Vector Machines. In *Advances in Neural Information Processing Systems (NIPS)*.

Zou, H., and Hastie, T. 2005. Regularization and Variable Selection via the Elastic Net. *Journal of the Royal Statistical Society, Series B* 67(2):301–320.