Optimization for Data Science - Project 10 - Sparse Logistic Regression

Derek Sweet
derekallen.sweet@studenti.unipd.it

Johanna Weiss
johanna.weiss@studenti.unipd.it

Emanuele Zangrando
emanuele.zangrando@studenti.unipd.it

Sercan Albut
sercan.albut@studenti.unipd.it

1. Introduction

Sparsity constraints are leveraged to solve many problems such as regression feature selection, compression sensing techniques and portfolio optimization. This paper addresses and recreates three algorithms used to solve the NP hard problem given the following objective function and constraints:

$$(P) \begin{cases} \min_x f(x) \\ x \in X \\ ||x||_0 \leq s \end{cases}$$

The three algorithms discussed in this paper are: Sparse Neighborhood Search (SNS), sparse approximation via Penalty Decomposition method (PD), and Greedy Sparse-Simplex method (GSS). A penalization approach becomes more difficult when considering all features, therefore the strict enforcement of the sparsity constraint is necessary to be able to handle the data. This method is useful in real life problems which tend to be non-convex and high dimensional. It is important to focus on efficiency with newly created constraints even though it is more challenging. This paper will address the methodology of the three papers [3, 4, 1], compare their theoretical differences and prove those differences through an empirical implementation and analysis using logistic regression on three different data sets.

2. Method

Paper 1, "A Unifying Framework for Sparsity Constrained Optimization", addresses the problem by suitably splitting the search to find a better set of active variables and for a better point in that particular subset. The main idea is to introduce a new variable y that will take care of the "combinatorial side" of problem (P). [3].

In Paper 2, "Sparse Approximation via Penalty Decomposition Methods", the PD algorithm addresses the problem by switching from an hard constraint formulation into a penalizing one. The main idea is to introduce a new variable y and a penalty function by adding the quadratic term $\frac{\rho}{2}||x - y||$ to the loss function. Instead of minimizing the loss, one minimizes this new function using a Block coordinate method . In that way, the optimization on the y block can be done exactly if there are no other constraint except the sparsity one. The idea is to gradually augment the penalty term and use old points as restarting points to get a sparse solution. [4].

In paper 3, "Sparsity Constrained Nonlinear Optimization: Optimality Conditions and Algorithms", the GSS algorithm addresses the problem by using a line search approach. At each iteration, the GSS algorithms looks if all the "available" coordinates are saturated: if they are not, it will look for the best value in all the coordinate lines, otherwise it will "free" one of them and look in another one. [1].

The next section will cover a brief overview of each of the papers with the main idea behind their algorithms.

2.1. Paper 1 Method

The constrain set of the problem is defined as χ:

$$\chi := \{x \in X \,|\, ||x||_0 \leq s\}$$

In logistic regression, the set X is the whole $\mathbb{R}^n$. Problem (P) can be difficult for several reasons:

- f in general can be non-convex;
- "balls" of "norm" zero are closed, non-convex and non-compact subsets of $\mathbb{R}^n$. That means they do not share the usual properties of a ball, so in general the constrain set is non-convex.

The problem is both, a problem of continuous optimization and a combinatorial problem, since the best subset of active variables needs to be found. The main idea of the SNS algorithm is to split these two problems using two separate variables, y y takes care of indicating the active variables and x indicates the value of the variable.

Remark. The L^0 ball of radius s is made of the union of all subspaces of $\mathbb{R}^n$ spanned by at most s axis, so:

$$B_{||\cdot||_0}(0,s] = \bigcup_{k\leq s\,, i_1,..,i_k\in\{1,..,n\}} span(e_{i_1},..,e_{i_k})$$

Each one of these spanned spaces can be represented as the solution of a diagonal linear system, in which the complementary set of coordinates should be equal to zero. We can represent the diagonal of this linear system with a vector y and write the system equivalently as $x \bigodot y = 0$, which is the interpretation of y introduced in the paper. It can be noticed that since $||x||_0 \leq s$, all subspaces contained in the ball are spanned by at most s vectors, therefore the linear system has a rank of at least $n - s$. This can be formulated as: $e^T y \geq n - s$. With this new variable, a point x is represented as (x, y). x is lying in the subspace represented by y, so y specifies which active variables is being used.

With this observation, the problem is reformulated in this way:

$$(P') \begin{cases} \min_{x,y} f(x) \\ x \in X \\ e^T y = \sum_i y_i \geq n - s \\ x \bigodot y = 0 \\ y \in \{0,1\}^n \end{cases}$$

For simplicity, it is defined:

$$\mathcal{Y} = \{y \in \{0,1\}^n \,|\, e^T y \geq n - s\}$$

In a local search around x the set of active variables is not changed, instead of looking for solutions locally around x as usual, the notion of a neighbourhood is introduced:

Definition. *(Neighborhoods in the "product" space $\mathcal{Y} \times \chi(y)$)*
Let $(y, x) \in \mathcal{Y} \times \chi(y)$ and let $\rho \geq 0$. A neighborhood is defined as follows :

$$N_\rho(x, y) := \{(\hat{x}, \hat{y}) \,|\, \hat{y} \in \mathcal{Y}, d_H(y, \hat{y}) \leq \rho, \hat{x} = x \bigodot \mathbb{1}_{\{i:y_i=\hat{y}_i\}}\}$$

where $\mathbb{1}_{\{i:y_i=\hat{y}_i\}}$ is an indicator vector.

With a neighborhood of radius ρ around (x, y) at most ρ entries of the variable y can be changed, which affects the active variables of x. This means, the neighborhood is changing active variables and projecting x orthogonally in the new subspace represented by $\hat{y}$ (that now is convex for a fixed y). With this new definition of locality, a point is defined as local minimizer (P') for the problem as follows:

Definition. *(local minimizer)*
$(x^*, y^*) \in \chi(y^*) \times \mathcal{Y}$ is said to be a local minimizer for problem (P') if there exists an $\epsilon > 0$ such that for all $(\hat{x}, \hat{y}) \in N(x^*, y^*)$ it holds:

$$f(x^*) \leq f(x) \;\; \forall x \in B(\hat{x}, \epsilon) \cap \chi(\hat{y})$$

This definition highlights the usefulness of introducing the dummy variable y and the new neighborhoods in the product space. The definition expresses locality as changing active variables, and as every projected point $\hat{x}$ in $\chi(\hat{y})$. Figure 1 visualizes optimality for a point in this new framework. The yellow point x is optimal if in the area enclosed by the magenta circles, there is no point with a lower objective value. For bigger ρ and bigger ϵ the optimality becomes more restrictive.
After introducing this new local-optimality definition, a necessary condition for a point to be globally optimal is derived which is called $\mathcal{N}$-stationarity. One can use figure 1 for the interpretation. Formally:

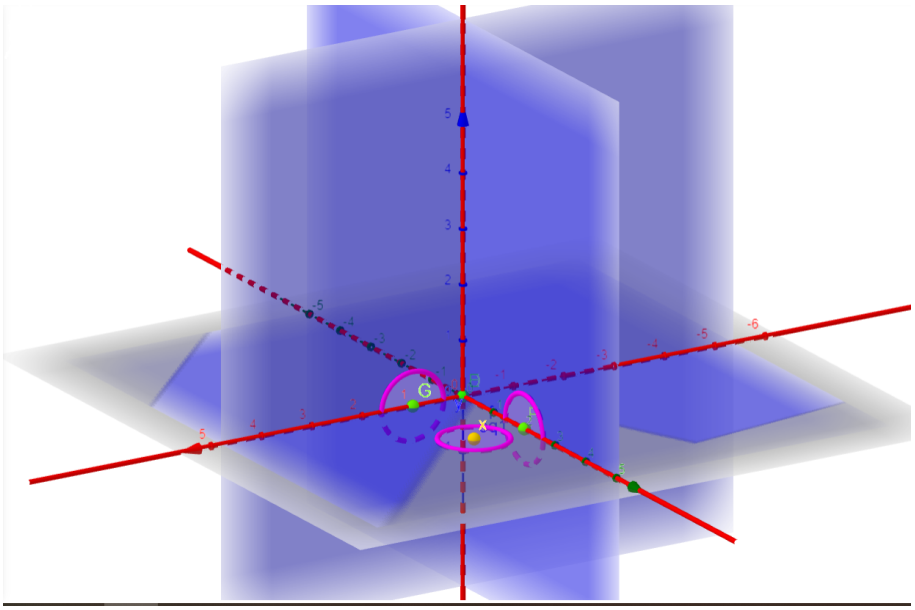

Figure 1. X=$\mathbb{R}^3$. In yellow: point x (as (x, y) with $y = (0, 0, 1)$). Green: $\hat{x}$ corresponding to points in $N_2(x, y)$. Magenta: balls of radius ϵ around every point $\hat{x}$ intersected with the corresponding $\hat{y}$.

Definition. *($\mathcal{N}$-stationarity)*
$x^* \in \chi$ is said to be $\mathcal{N}$-stationary for problem (P') if there exists $y^* \in \mathcal{Y}$ such that:

1. *(x^*, y^*) is feasible;*
2. *x^* is stationary for the continuous problem*
$$\begin{cases} \min_x f(x) \\ x \in \chi(y^*) \end{cases}$$
3. *$\forall(\hat{x}, \hat{y}) \in N_\rho(x^*, y^*)$ it holds $f(x^*) \leq f(\hat{x})$ and if $f(x^*) = f(\hat{x})$ then $\hat{x}$ is stationary for the problem*
$$\begin{cases} \min_x f(x) \\ x \in \chi(\hat{y}) \end{cases}$$

It can be proven that this condition is really necessary:

Remark. (Necessarity of $\mathcal{N}$-stationarity)
x^* optimal point for problem (P) $\implies$ x^* $\mathcal{N}$-stationary.

$\mathcal{N}$-stationarity is using the general definition of stationary points for sub-problems. The stationary condition depends on the structure of the constrain. There are two possible cases:

1. the constrain $\chi(y)$ has no special structure;
2. the constrain $\chi(y)$ has a special structure such that Karush-Kuhn-Tucker theory can be used.

For the first case (1) the following definition of stationarity is used:

Definition. *(General stationarity)*
Given $(x^*, y^*) \in \chi(y) \times \mathcal{Y}$, point x^* is stationary for the problem
$$\begin{cases} \min_x f(x) \\ x \in \chi(y^*) \end{cases}$$
if and only if the following happens:
$$x^* = \Pi_{\chi(y^*)}\left[x^* - \nabla f(x^*)\right]$$

In the case X$= \mathbb{R}^n$, the fixed point equation of this last definition can be interpreted geometrically, by observing that the direction of the gradient $\nabla f(x^*)$ is orthogonal to $\chi(y^*)$.
The main goal of the SNS algorithm is to search for $\mathcal{N}-$stationary points for problem (P), to find suitable candidates for global optimality. In a simplified way, this search is done following three main steps: given a starting point (x^k, y^k) and a radius ρ, one has to

1. locally explore around x^k in the $\chi(y^k)$ subspace by using an Armijo line search approach;
2. extract from $\mathcal{N}_\rho(x^k, y^k)$ only the points that are not significantly worse than x^k in term of objective value;
3. perform a local continuous search around every x and stop as soon as a point satisfying a sufficient decrease condition is found.

Increasing the parameter ρ is extending the set of active variables configurations, which refines the search and increases the complexity of the search.

2.2. Paper 2 Method

Paper two takes a different approach in trying to solve this problem. Instead of using hard constraints, a penalty formulation of problem (P) is introduced using a new variable y. the penalty formulation is, for $\rho > 0$:

$$(P') \begin{cases} \min_{x,y} q_\rho(x, y) = f(x) + \frac{\rho}{2}||x - y||_2^2 \\ x \in X \\ y \in \mathcal{Y} = \{y \in X \,|\, ||y||_0 \leq s\} \end{cases}$$

Since logistic regression problems are not constrained in x, $X = \mathbb{R}^n$ in this problem.

The main idea of penalty decomposition methods is to solve the unconstrained problem instead of the constrained one and then to gradually augment the penalty. The final points are used as restarting points for the next iteration . Just to be consistent with paper one, the L-BFGS method is optimizing q_ρ at every step. For a fixed x, the constrain optimization of q_ρ can be done exactly. The algorithm assumes that a feasible starting point x^{feas} is known.
The PD algorithm follows 3 main steps: Let (x^k, y^k) be the starting points and let ρ_k be the starting penalty term. Then one has to:

1. Use BCD on q_{ρ_k} (blocks are the one made of x and y) until a stopping condition is reached, call these new points (x^{k+1}, y^{k+1});
2. Augment the penalizing coefficient, call it ρ_{k+1};
3. If the minimization of $q_{\rho_{k+1}}$ for y^{k+1} leads to an objective value worse than the known feasible point and of the minimization of q_{ρ_0} for fixed y_0, then return to $y_{k+1} = x^{feas}$. In the other case, keep y^k and restart the procedure.

2.3. Paper 3

The same problem as in section 2.1 is considered. Compared to section 2.1, the notation is slightly different, which is explained below:

- $C_s = \{x \in \mathbb{R}^n \,|\, ||x||_0 \leq s\}$ the L^0 "ball";
- $M_i(x) = i^{th}$ largest absolute component of x;
- $I_1(x) = \{i = 1, .., n \,|\, x_i \neq 0\}$, $I_0(x) = \{i = 1, .., n \,|\, x_i = 0\}$

An additional assumption is that f is lower bounded in all $\mathbb{R}^n$.
There are several conditions that are necessary for an optimal point of the problem (P). In this paper, three necessary conditions are derived, one more restrictive than the other. The simplest one is Basic feasibility:

Definition. *(Basic feasibility)*
$x^* \in C_s$ is said to be BF for problem (P) if :

1. *if* $||x^*||_0 < s \Rightarrow \nabla f(x^*) = 0$

2. *if* $||x^*||_0 = s \Rightarrow \nabla_i f(x^*) = 0,\ \forall i \in I_1(x)$

Figures 2 and 3 visually explain the definition with examples in which $\mathbb{R}^n = \mathbb{R}^3$ and $s = 2$.
Basic stationarity turned out to be too weak as a necessary condition, so a stronger one is introduced.

Definition. *(L-stationarity/equivalent characterization)*
$x^* \in C_s$ is said to be L-stationary for (P) if :

$$x^* \in P_{C_s}(x^* - \frac{1}{L}\nabla f(x^*))$$

This condition is equivalent to the following:

$$|\nabla_i f(x^*)| \begin{cases} \leq LM_s(x^*), \textit{if } i \in I_0(x^*) \\ = 0, \textit{if } i \in I_1(x^*) \end{cases}$$

Remark. It is written $x^* \in P_{C_s}(x^* - \nabla f(x^*))$ because the orthogonal projection is not unique in general if the set in which we are projecting is not convex.
L-stationarity geometrically means that if we follow the gradient for a while from x^* and we reproject the new point on the constrain, x^* is again one of the projections.
Moreover, if f is L_1 stationary, then it is also L_2 stationary for all $L_2 \geq L_1$.

They proved the following:

Proposition. *(L-stat. and BF)*
L-stationarity for some $L > 0$ implies basic feasibility.

Assuming LCG condtion with Lipschitz constant $L(f)$ they proved also the following:

Proposition. *(L-stationarity is necessary for optimality)*
Let x^* be optimal for problem (P). Then:

1. *x^* is L-stationary for $L \geq L(f)$;*
2. *$P_{C_s}(x^* - \frac{1}{L}\nabla f(x^*))$ is a single point.*

A problem of L-stationarity is that a bound on the Lipschitz constant has to be known which can be hard to obtain. Therefore, another necessary condition stronger than L-stationarity is derived.

Definition. *(CW-minima)*
Let x^* be feasible for (P). Then x^* is said to be a CW minima if one of the following conditions is true:

1. *$||x||_0 < s$ and for all $i = 1, .., n$ one has:*

$$f(x^*) = \min_{t \in \mathbb{R}} f(x^* + te_i)$$

2. *$||x||_0 < s$ and $\forall i \in I_1(x^*), j = 1, .., n$ one has:*

$$f(x^*) \leq \min_{t \in \mathbb{R}} f(x^* - x_i^* e_i + te_j)$$

Remark. For the geometric interpretation of this last definition, figures 2 and 3 can be used.
Figure 2 represents the first case, in which $||x||_0 < s$. In this case at least another non-null coordinate to x can be added for searching. The CW minimum condition tells us that no matter in which coordinate direction around x we search, the green point is always the minimum along all the search lines.
Figure 3 represents the second case, in which $||x||_s = s$. In this case no more non-null coordinate can be added for searching. If searching locally around the point is not successful, one coordinate has to be deleted in order to add another for searching. The function $f(x - x_i e_i + te_j)$ indicates that the i^{th} coordinate of x is deleted and the j^{th} coordinate is added to look for better points in the direction of the line e_j. So, condition (2) tells us that even if we try all possible combinations of i and j for searching, the point in which we are is always better (in terms of objective value).

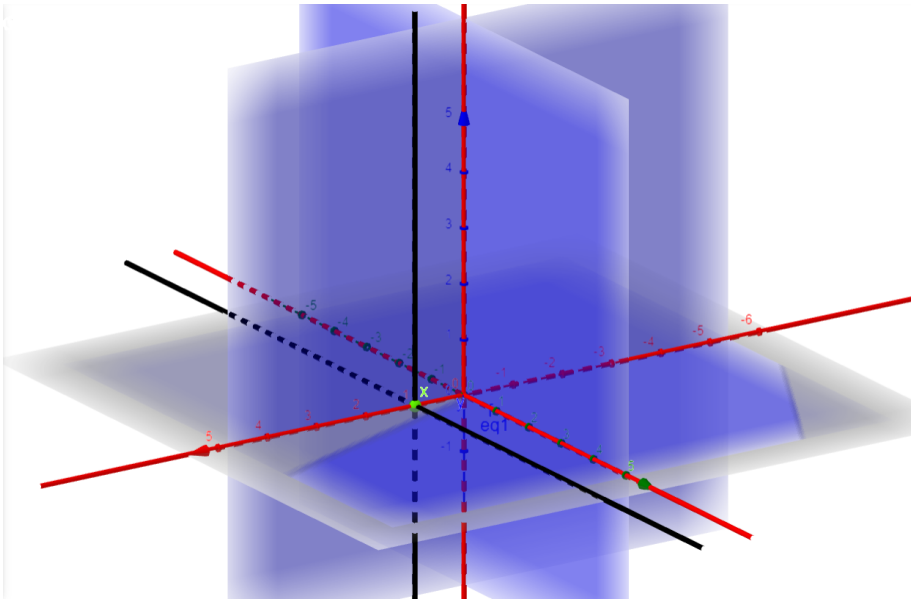

Figure 2. In green: x,optimal point for problem (P) with $||x||_0 = 1 < 2$. Blue: constrain set C_2. Since $||x||_0 = 1 < 2$ another variable can be added, so we can move along the black lines and the red line in which x lie. Since x is it optimal, the derivative in these directions is zero, but since they span $\mathbb{R}^3$ it means that the whole gradient is zero.

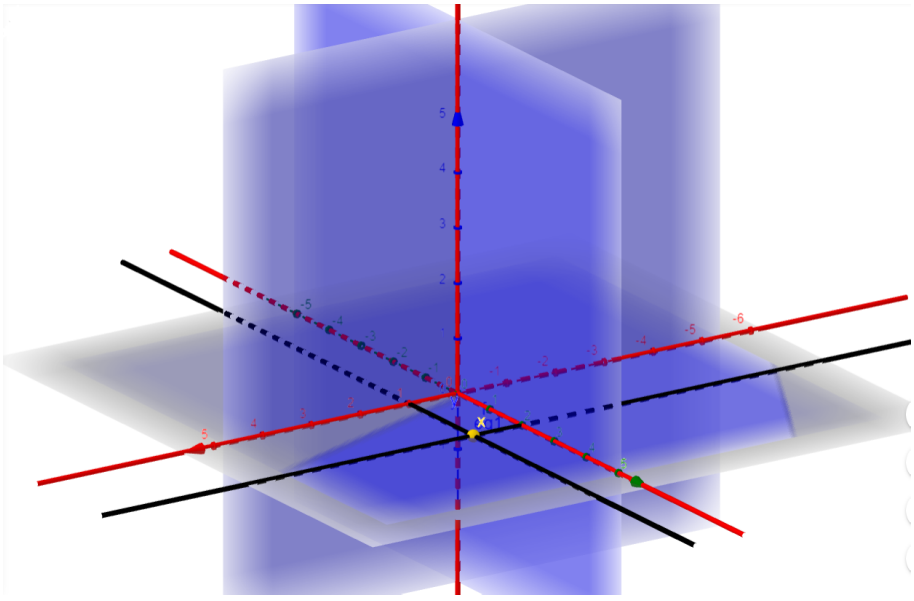

Figure 3. In yellow: x,optimal point for problem (P) with $||x||_0 = 2$. Blue: constrain set C_2. Since $||x||_0 = 2$ we cannot add another variable, so we can move along the black lines. Since x is it optimal, the derivative in these directions is zero, and they are exactly the directions e_i for $i \in I_1(x)$. This shows that $\nabla_i f(x) = 0$ for all $i \in I_1(x)$

.

The GSS algorithm looks for this last kind of stationary points, it search for coordinatewise minima by using a greedy approach. There are two main steps:

1. Scenario 1 $||x^k||_0 < s$: minimize f along each line $x^k + te_i$, if a better objective value is found in all the searches then update the point x^k with the best of these one. If no line minimization provided a better point, then stop;
2. Scenario 2 $||x^k||_0 = s$: for every $i \in I_1(x^k)$ and $j = 1, .., n$, try to delete the i^{th} coordinate and search along the j^{th}. If at least one better point is found then update x^k to the best one of all. If no better objective value is found, then stop.

3. Data Sets

In order to test the results of the algorithms from each paper, three data sets were used: (1) Heart (Statlog), (2) Breast Cancer Wisconsin (diagnostic), and (3) Spambase.

(1) The Heart Statlog data (will be referred to as 'heart' data) has 13 features of mixed-integer size containing demographic and health diagnostics of individuals. Each individual is identified to if there is absence or presence of heart disease. There are 270 observations in total. [2]

(2) The diagnostic Breast Cancer Wisconsin data (will be referred to as 'cancer' data) has 30 features of mixed-integer size containing descriptors of cell nuclei collected from images. Each nuclei is identified as being malignant or benign. There are 569 observations in total. [6]

(3) The QSAR biodegradation data set (will be referred to as 'QSAR' data) is a multivariate data set with 41 features of real and integer values, containing attributes of 1055 chemicals such as the number of oxygen or nitrogen atoms. Based on their characteristics, each chemical is classified into ready and not ready biodegradable.[5]

4. Experiments

The three algorithms described have been compared in terms of their performance with respect to different values of constraint parameter (s), neighboring measurement parameter (ρ) and the impact of different starting points with different sets of data. The convex subproblems of PD, GSS, and local optimization steps have been solved by L-BFGS algorithm, while for the stopping condition of BCD we employed the suggestion of Paper 2 [4]. Algorithms SNS, PD, and GSS were implemented using the parameters suggested in Paper 1 [3].

4.1. Expected Results

As a point of reference, the results of Paper 1 (2.1) are shortly discussed. For ρ = 2,3,4 the loss of SNS improves as this parameter increases. While an increase in the ρ parameter improves performance, the running time subsequently increases. Except for SNS(1), all SNS implementations are performing better than GSS. The reason is that GSS is a greedy algorithm and it also requires more iteration to converge. An increase in the neighborhood factor (ρ) of SNS is also considered beneficial on performance when the sparsity constraint becomes less strict. However, decreasing the restriction on sparsity constraints is more time-consuming since the combinations to expore increase. The PD methods do not have a prominent convergence time or loss. It converges to a feasible point but it lacks global optimization properties which means it is missing a globally optimal solution.

4.2. Experiment 1: Choice of ρ and s

The results of the first experiment were successful in generating sparse solutions across all three algorithms. Figure 4, shows a comparison of the algorithms with s=3 and s=4 on the QSAR, heart and cancer data. For SNS, two versions are compared, one with $\rho = 1$ and one with $\rho = 2$. Figure 4 shows the running time and quality of the solution of the five algorithms. Overall, the behaviour of the algorithms show similar behaviour on the three data sets.

SNS(1) stands out the most because of it's poor quality solution. The low quality of the result can be explained with the fact that the radius 1 does not allow to swap active coordinates, so that possibly important neighbours with better results are not explored. Increasing the radius to 2 improves the quality of the solution significantly, however, it also slows down the algorithms. This is due to the larger range of points in the neighbourhood that are explored.

The time for convergence of GSS and PD are similar in almost every one of our tests and also the final objective value is similar in the two algorithms.

The PD loss plot does not follow a continuous decline due to the fact that the point is not hard constrained during the whole process, but after a while it's converging to a constrained point (and that's why the loss is increasing a bit).

When comparing our results to Paper 1 there is one key difference with the behaviour of SNS in particular. As expected, SNS(1) is not attaining a good objective value, however contrary to the results in Paper 1, it is slower than GS and PD. As mentioned before, SNS(2) is clearly doing better than SNS(1) in terms of loss, but in our experiments it also did not attain a value similar to GSS or PD. The running time of it was also much bigger than the other ones, but this can be due to the stopping condition we employed for the PD. The behaviour of SNS(1) in the qsar dataset is much better, in our experiment it is still the last one in terms of objective value but the runtime now is competitive with PD.

When investigating the cause for the longer than expected run time of SNS, the key inefficiency identified was a symptom of

generating a large set of feasible W_k and the breaking condition

$$||x^{(j)} - \Pi_{\chi(y')}\Big[x^{(j)} - \nabla f(x^{(j)})\Big]|| > \mu_k + ||x^{(k)} - \Pi_{\chi(y^{(k)})}\Big[x^{(k)} - \nabla f(x^{(k)})\Big]||$$

not being satisfied in most of the W_k vectors resulting in an exceptionally long search.
In Figure 5 we can see the final loss vs the runtime of each one of the algorithm. As in the tests of 2.1 there not much overall difference in the runtimes of PD and GSS, and also the attained losses are similar. However, for the qsar (s=3) and for the qsar (s=4) the difference in the final loss is bigger. SNS(1) it not performing good in terms of objective value as expected, it is not even the fastest but it is competitive with the others in term of time.
SNS(2) instead was the slowest in our experiments, and its best objective value is not comparable with the others.
To summarize, the main difference we noticed with respect to the tests conducted in 2.1 is the behaviour of SNS(2), it took longer than expected but it stopped before reaching a competitive objective value.

4.3. Experiment 2: Starting points

The second experiment is testing the effect of using different starting vectors for SNS with $\rho = 1$, PD and GSS. Instead of using the zero vector as a starting point for the algorithms, as in the first experiment, the weights are sampled from a standard normal distribution and masked in such a way that they satisfy the constraints at the beginning. For each of the algorithms 9 starting vectors are generated. The resulting running times are reported in Table 1.

The results are consistent with the results from the first experiment for all starting points.PD and GSS have very consistent performance across all starting vectors and are both achieving a similar loss. The CPU run time of PD was greater than GSS and had a bit more variance ranging from 3.1 to 3.7. SNS, however had the largest variance in performance indicating it is much more sensitive to starting points. In the best case, SNS ran in 14.6 seconds but in the worse case, 21.7 seconds. The distribution plots in Figure 6 show that many of the results of SNS were consistently concentrated with two larger outliers. There is a high correlation between longer run time for SNS and a smaller loss. The best loss of 146.9 was associated to the longest run time of 21.7. Unfortunately, despite the improved loss performance of this starting point, SNS was still not better than PD and GSS in our tests.

SNS(1) Loss	SNS(1) Time	PD Loss	PD Time	GSS Loss	GSS Time
162.0811353	16.97429002	115.2951896	3.600581187	115.2664734	0.757495751
162.0815522	16.93981574	115.2951896	3.366991967	115.2664734	0.678498566
155.4878976	15.92001942	115.2951896	3.122872836	115.2664734	0.624822036
162.1220387	14.63003325	115.2951896	3.202642779	115.2664734	0.709196891
157.6064976	15.48795317	115.2951896	3.747683227	115.2664734	0.62362322
162.0675165	16.14920106	115.2951896	3.338881858	115.2664734	0.731399527
162.0811406	16.35405256	115.2951896	3.577261126	115.2664734	0.617396916
146.9280671	21.70400285	115.2951896	3.432035847	115.2572286	0.685006811
165.2021227	18.36031337	115.2951896	3.321582691	115.2664734	0.73311389

Table 1. Table to showcase performance differences of each model using different starting points (each row)

5. Conclusion

Overall, all three algorithms discussed in this paper address and can produce sparse solutions for tasks such as feature selection by enforcing sparsity with a L_0 norm constraint. While all of them have different approaches, the key takeaway is that all this algorithms work, but they all do that in a really different way. In our experiments, we showcase the ability of all three methods (SNS, PD, and GSS) to generate sparse solutions in the Logistic Regression scenario. With a higher ρ and s, SNS is capable of producing the best objective value, but run time of this method will significantly increase as these parameters increase and may vary based on the starting vector.

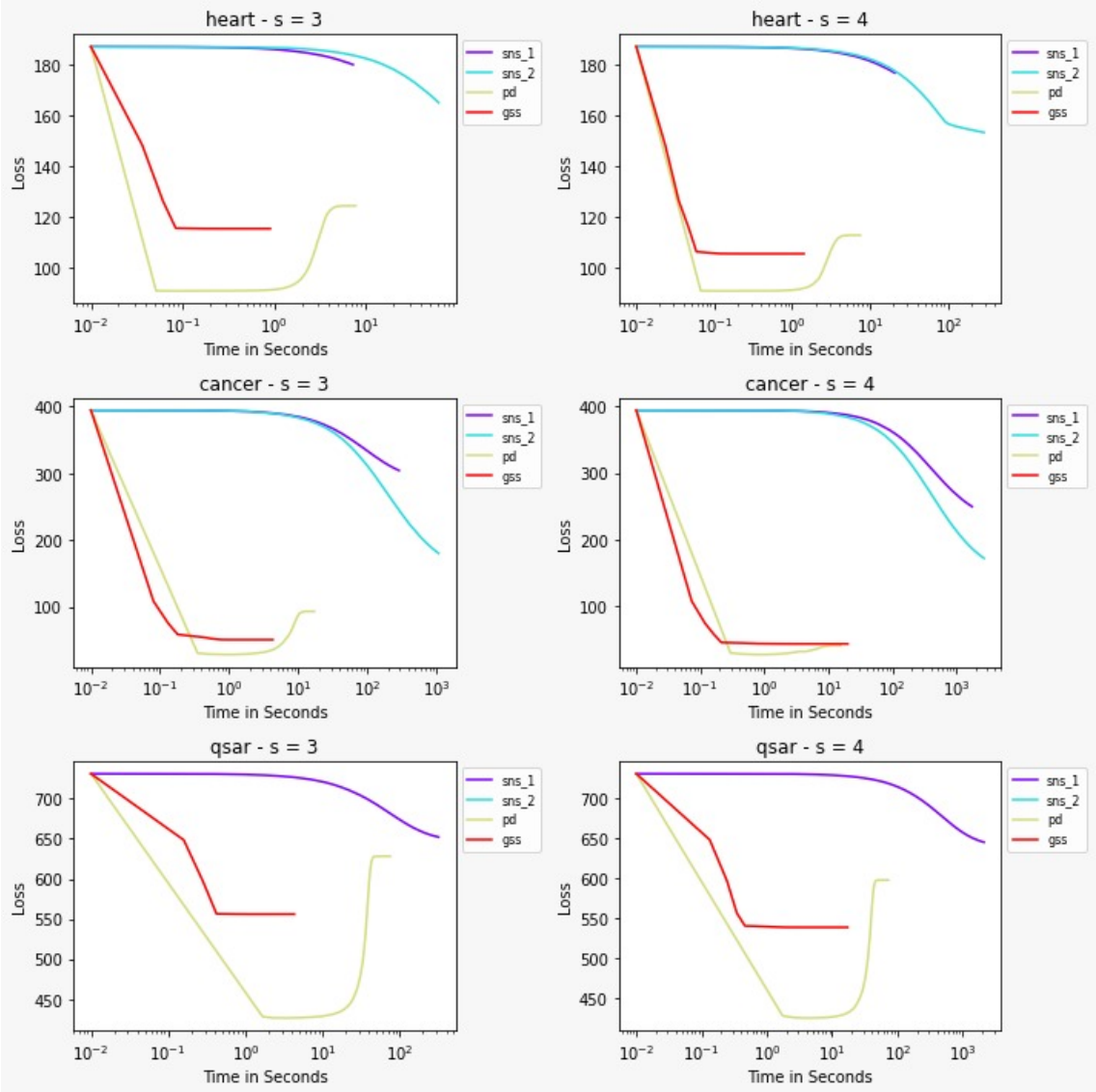

Figure 4. Quality and running time of algorithms

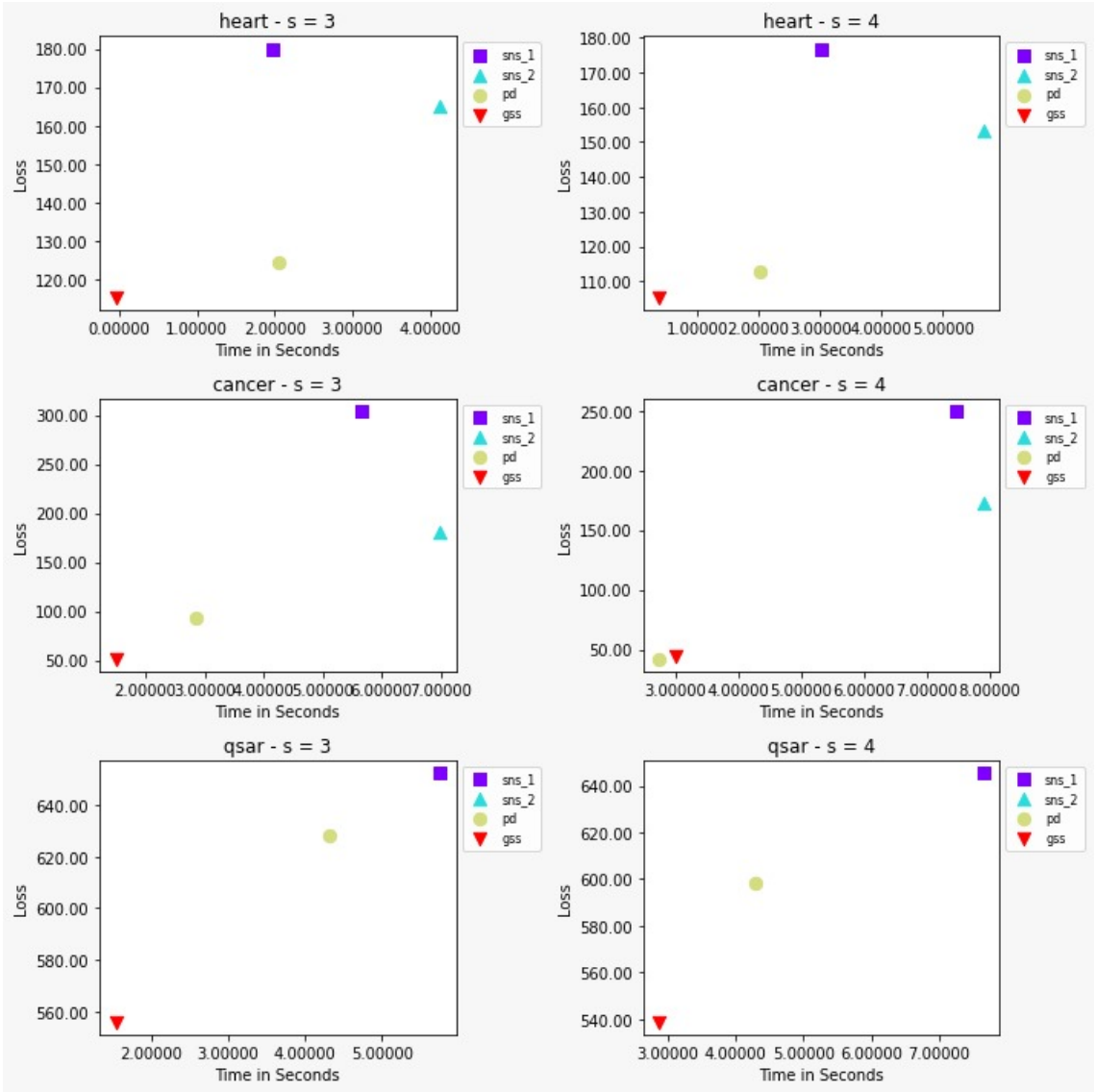

Figure 5. Quality and running time of algorithms

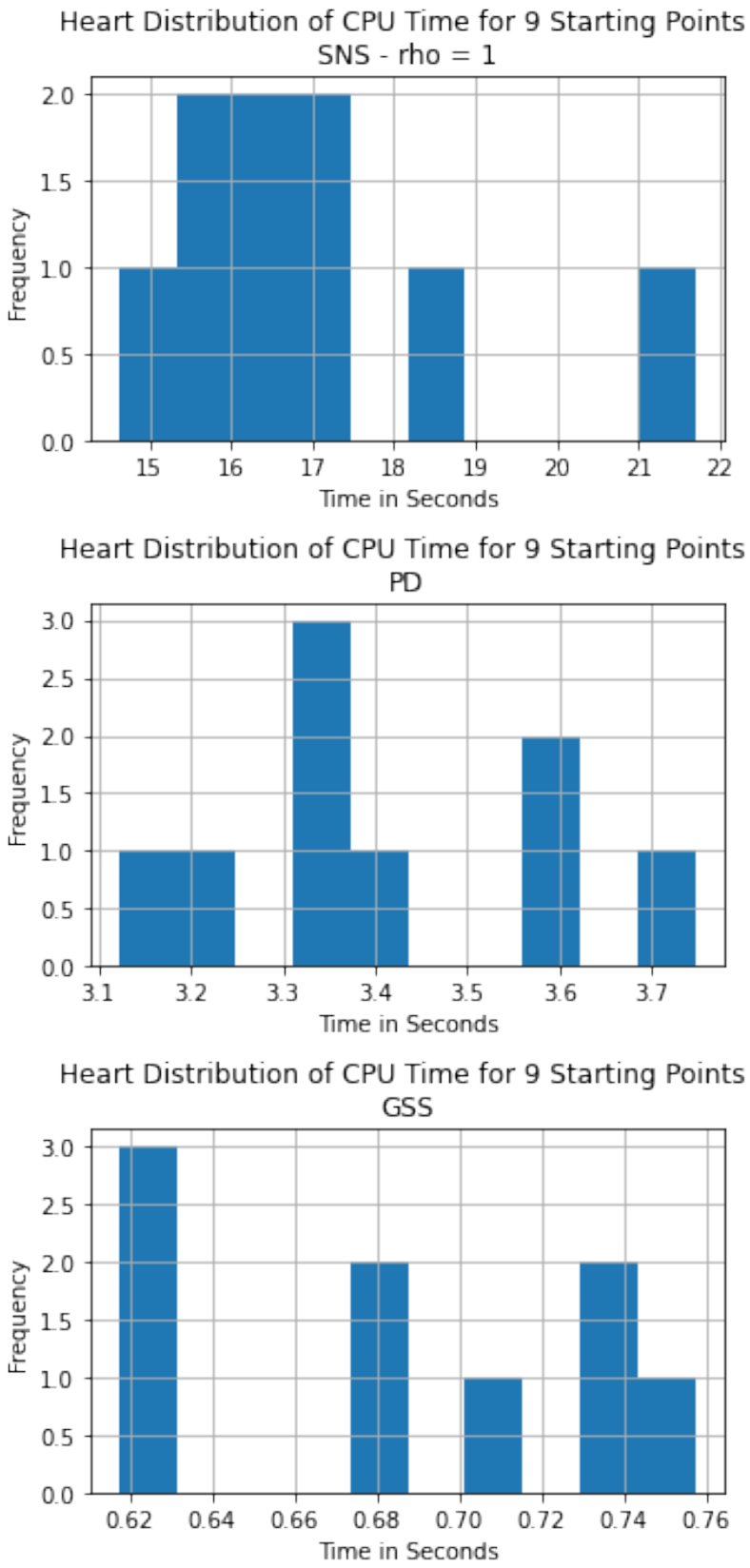

Figure 6. Distribution of CPU time at varying starting points

References

[1] Amir Beck and Yonina C. Eldar. Sparsity constrained nonlinear optimization: Optimality conditions and algorithms, 2012.

[2] A Frank and A Asuncion.

[3] M. Lapucci, T. Levato, F. Rinaldi, and M. Sciandrone. A unifying framework for sparsity constrained optimization, 2021.

[4] Zhaosong Lu and Yong Zhang. Sparse approximation via penalty decomposition methods, 2012.

[5] K. Mansouri, D. Ballabio T. Ringsted, and R. Todeschini. Quantitative structure - activity relationship models for ready biodegradability of chemicals, 2013.

[6] William H Wolberg, Nick L Street, and Olvi Mangasarian.