An extension of Stable Regression to Support Vector Machines and Logistic Regression

Hamza Tazi Bouardi
Operations Research Center
Massachusetts Institute of Technology
htazi@mit.edu

Pierre-Henri Ramirez
Operations Research Center
Massachusetts Institute of Technology
phenri@mit.edu

Abstract

In classical machine learning, it has become common practice to randomly split our data into train, validation and test sets. While it is arguably a good practice to have a random split to obtain the test set in order to replicate real life and limit the bias in our model assessment, it is absolutely not when we split the second part into train and validation. Moreover, such a model fitted and validated with randomly split data might be extremely variable, which might hurt both its interpretability and its performances. This paper extends the robust optimization fitting techniques already developed for Linear Regression to other classical machine learning models - such as Support Vector Machines and Logistic Regression - and conducts experiments on real-life datasets comparing classical approaches to this novel approach in terms of prediction errors and coefficients variability. We have found that performance-wise the Stable versions were at least as good as their Classic counterparts, while the coefficient variability (quantified by the average standard deviation) was reduced drastically, sometimes up to 200%, therefore proving the robustification of the models studied.

1 Introduction

Today in Machine Learning, the current paradigm when performing a regression is widely accepted and normalized. Namely, partitioning (randomly or not) data into training, validation and test set. The first two are used to respectively train models and tune their hyperparameters through assessing the models' performance on the validation set. Another well-known way of proceeding is considering the training and validation set as a single training set and perform cross-validation on it (**?**). Finally, the test set is used to compute the final model's out-of-sample performance, conceptually seen as a way to approximate the model's ability to generalize on new data.
However this approach might present some undesired characteristics for real life applications. One of them is the variability of the model itself, which might change significantly with the training and validation sets. To tackle this challenge, [2] proposed a novel optimization based approach, in which the objective used in the regression to minimize bias over the training set is no longer the classical average of all n "individual losses" (n being the size of the training set $(x_i, y_i)_{i\in[n]}$), $\frac{1}{n}\sum_{i=1}^{n}\ell(x_i, y_i)$, but the worst average among a sub-sample of k observations in the training dataset such that the ratio $\frac{k}{n}$ corresponds to the train/validation split ratio we want. This is formulated as follows:

$$(\alpha) \qquad \min_{w} \max_{z_i\in\{0,1\},\, \sum_{i=1}^{n} z_i = k} \sum_{i=1}^{n} z_i \ell(f_w(x_i), y_i)$$

As it is, this problem is not tractable, however [2] proposed a linear optimization approach solving (α) when ℓ is the ℓ_1 norm and $f_w(x) = w^\top x$. It consists in taking the dual of (α) and remarking that

MIT 9.520/6.860: Statistical Learning Theory and Applications (Fall 2019).

the optimization over the set of feasible z can be done in its convex hull. We can indeed rewrite (α) as :

$$(\beta) \qquad \min_{w} \max_{z_i \in [0,1],\, \sum_{i=1}^{n} z_i = k} \sum_{i=1}^{n} z_i \left| w^\top x_i - y_i \right|$$

Taking the dual of $\beta givesus \ :(\gamma) \quad \max \sum_{i=1}^{n} u_i + k\theta s.t. \quad u_i + \theta \geq w^\top x_i - y_i \quad \forall i = 1 \cdots n \qquad u_i + \theta \geq -(w^\top x_i - y_i) \quad \forall i = 1 \cdots n$

And we are now facing a simple linear optimization problem which can be solved extremely efficiently. Inspired by this approach, we will extend it to Stable Support Vector Machines (SVM) and Stable Logistic Regression.

2 A Stable Soft-Margin SVM formulation

2.1 Support Vector Machines

Support Vector Machines (SVMs) are a useful technique for data classification. It is a supervised learning model, that can be formulated very simply as follows :

We are given n observations $(\mathbf{x}_i, y_i)$

- $\mathbf{x}_i \in \mathbb{R}^p$
- $y_i \in \{-1, 1\}$

The goal is to learn to classify y using x through the relation $y = sign(\langle w, x \rangle + b)$ by choosing correctly w. The model fitting is usually done with the following optimization problem :

$$\begin{aligned} \min \quad & \frac{1}{2}\mathbf{w}^\top \mathbf{w} = \frac{1}{2}||\mathbf{w}||^2 \qquad \text{(objective function)} \\ s.t. \quad & y_i(\mathbf{x}_i^\top \mathbf{w} + b) \geq 1, \ \text{ or } \ 1 - y_i(\mathbf{x}_i^\top \mathbf{w} + b) \leq 0, \quad (i = 1, \cdots, n) \end{aligned}$$

This Quadratic Programming problem can be solved using Lagrange multipliers method and dualizing the constraints to minimize the following:

$$L_p(\mathbf{w}, b, \alpha) = \frac{1}{2}||\mathbf{w}||^2 + \sum_{i=1}^{n} \alpha_i (1 - y_i(\mathbf{x}_i^\top \mathbf{w} + b))$$

with respect to $\mathbf{w}$, b and the Lagrange coefficients $\alpha_i \geq 0$ $(i = 1, \cdots, \alpha_n)$. We let (KKT conditions):

$$\frac{\partial}{\partial W} L_p(\mathbf{w}, b) = 0, \quad \frac{\partial}{\partial b} L_p(\mathbf{w}, b) = 0$$

These lead, respectively, to

$$\mathbf{w} = \sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j, \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

Substituting these two equations back into the expression of $L(\mathbf{w}, b)$, we get the *dual problem* (with respect to α_i) of the above *primal problem*:

$$\begin{aligned} \max \quad & L_d(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \\ s.t. \quad & \alpha_i \geq 0, \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \end{aligned}$$

The dual problem is related to the primal problem by:

$$L_d(\alpha) = \inf_{(\mathbf{w}, b)} L_p(\mathbf{w}, b, \alpha)$$

i.e., L_d is the greatest lower bound (infimum) of L_p for all $\mathbf{w}$ and b.

Solving this dual problem (an easier problem than the primal one), we get the Lagrangean multipliers α_i, as well as the optimal separating hyperplane $\mathbf{w}$. Those points $\mathbf{x}_i$ on either of the two subspaces H_+ (for which +1 is predicted) and H_- (for which -1 is predicted) for which the equality $y_i(\mathbf{w}^\top \mathbf{x}_i + b) = 1$ holds are called *support vectors* and they correspond to positive Lagrange multipliers $\alpha_i > 0$. The training (and model fitting) depends only on the support vectors, while all other samples away from the planes H_+ and H_- are not important. For a support vector $\mathbf{x}_i$ (in H_- or H_+), the constraining condition is

$$y_i \left(\mathbf{x}_i^\top \mathbf{w} + b\right) = 1 \quad (i \in \mathcal{SV})$$

Where $\mathcal{SV}$ is a set of all indices of support vectors $\mathbf{x}_i$ (corresponding to $\alpha_i > 0$). Substituting

$$\mathbf{w} = \sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j = \sum_{j \in \mathcal{SV}} \alpha_j y_j \mathbf{x}_j$$

we get

$$y_i(\sum_{j \in \mathcal{SV}} \alpha_j y_j \mathbf{x}_i^\top \mathbf{x}_j + b) = 1$$

Notice that the sum only contains terms corresponding to those support vectors $\mathbf{x}_j$ with $\alpha_j > 0$, i.e.

$$y_i \sum_{j \in \mathcal{SV}} \alpha_j y_j \mathbf{x}_i^\top \mathbf{x}_j = 1 - y_i b$$

For the optimal weight vector $\mathbf{w}$ and optimal b, we have:

$$\begin{aligned} ||\mathbf{w}||^2 &= \mathbf{w}^\top \mathbf{w} = \sum_{i \in \mathcal{SV}} \alpha_i y_i \mathbf{x}_i^\top \sum_{j \in \mathcal{SV}} \alpha_j y_j \mathbf{x}_j = \sum_{i \in \mathcal{SV}} \alpha_i y_i \sum_{j \in \mathcal{SV}} \alpha_j y_j \mathbf{x}_i^\top \mathbf{x}_j \\ &= \sum_{i \in \mathcal{SV}} \alpha_i (1 - y_i b) = \sum_{i \in \mathcal{SV}} \alpha_i - b \sum_{i \in \mathcal{SV}} \alpha_i y_i \\ &= \sum_{i \in \mathcal{SV}} \alpha_i \end{aligned}$$

The last equality is due to $\sum_{i=1}^{n} \alpha_i y_i = 0$ shown above. Recall that the distance between the two margin planes H_+ and H_- is $2/||\mathbf{w}||$, and the margin, the distance between H_+ (or H_-) and the optimal decision plane H_0, is

$$\frac{1}{||\mathbf{w}||} = \left(\sum_{i \in \mathcal{SV}} \alpha_i \right)^{-1/2}$$

2.2 Soft-Margin SVMs

When the two classes are not linearly separable (e.g., due to noise), the constraints for the optimal hyperplane can be relaxed by including an extra term:

$$y_i(\mathbf{x}_i^\top \mathbf{w} + b) \geq 1 - \xi_i, \quad (i = 1, \cdots, n)$$

For minimum error, $\xi_i \geq 0$ should be minimized as well as $||\mathbf{w}||$, and the objective function becomes:

$$\begin{aligned} \min \quad & \mathbf{w}^\top \mathbf{w} + \lambda \sum_{i=1}^{n} \xi_i^k = ||w||_2^2 + \lambda ||\xi||_k^k \\ s.t. \quad & y_i(\mathbf{x}_i^\top \mathbf{w} + b) \geq 1 - \xi_i, \quad \text{and} \quad \xi_i \geq 0; \quad (i = 1, \cdots, n) \end{aligned}$$

Here λ is a regularization parameter that controls the trade-off between maximizing the margin and minimizing the training error. Small λ tends to emphasize the margin while ignoring the outliers in the training data, while large λ may tend to overfit the training data. We will be considering the 2-Norm soft margin SVMs where $k = 2$.

$$\begin{aligned} \min \quad & \mathbf{w}^\top \mathbf{w} + \lambda \sum_{i=1}^{n} \xi_i^2 \\ s.t. \quad & y_i(\mathbf{x}_i^\top \mathbf{w} + b) \geq 1 - \xi_i, \quad (i = 1, \cdots, n) \end{aligned} \tag{1}$$

Note that the condition $\xi_i \geq 0$ is dropped, since if $\xi_i < 0$, we can set it to zero and the objective function is further reduced. Alternatively, if we let $k = 1$, the problem can be formulated as

$$\begin{aligned} \min \quad & \mathbf{w}^\top \mathbf{w} + \lambda \sum_{i=1}^{n} \xi_i \\ s.t. \quad & y_i(\mathbf{x}_i^\top \mathbf{w} + b) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0; \quad (i = 1, \cdots, n) \end{aligned} \tag{2}$$

Which corresponds to the 1-norm soft margin problem. The algorithm based on 1-norm setup, when compared to 2-norm algorithm, is less sensitive to outliers in training data. When the data is noisy, 1-norm method is generally used to ignore the outliers.

2.3 Stable 1-Norm Soft-Margin SVM

First of all, notice that another approach to 1-norm Soft Margin SVM is the following :

$$\min \quad \mathbf{w}^\top \mathbf{w} + \lambda \sum_{i=1}^{n} |1 - y_i(x_i^\top w + b)|_+ \tag{3}$$

Where $y \mapsto |y|_+ = \max(y, 0)$. The stable formulation of (3) is, for $k < n$:

$$\begin{aligned} \min \quad & \mathbf{w}^\top \mathbf{w} + \lambda \max_z \sum_{i=1}^{n} z_i |1 - y_i(x_i^\top w + b)|_+ \\ s.t. \quad & \sum_{i=1}^{n} z_i = k \quad \text{and} \quad z_i \in \{0, 1\}; \quad (i = 1, \cdots, n) \end{aligned} \tag{4}$$

Let us approach (4) sequentially. We can see that to solve it, we need to perform two optimizations, a minimization over a maximization problem. Let us first consider the inner maximization problem for a given w :

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} z_i |1 - y_i(x_i^\top w + b)|_+ \\ s.t. \quad & \sum_{i=1}^{n} z_i = k \quad \text{and} \quad z_i \in \{0, 1\}; \quad (i = 1, \cdots, n) \end{aligned} \tag{5}$$

It is clear that the maximization can be done over the convex hull of the feasible set, i.e.

$$CH\left(\left\{ z \in \mathbb{R}^n \middle| \sum_{i=1}^{n} z_i = k, z_i \in \{0, 1\}, i = 1, \cdots, n \right\}\right) = \left\{ z \in \mathbb{R}^n \middle| \sum_{i=1}^{n} z_i = k, 0 \leq z_i \leq 1, i = 1, \cdots, n \right\}$$

Now that the inner problem is a linear optimization problem, let's compute the dual. We define

$$\mathcal{L}(z, \theta, u_i) \in \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^n \mapsto \sum_{i=1}^{n} z_i |1 - y_i(x_i^\top w + b)|_+ + \theta(\sum_{i=1}^{n} z_i - k) + \sum_{i=1}^{n} u_i(1 - z_i)$$

(5) is equivalent to $\max_{z \geq 0} \min_{\theta \in \mathbb{R}, u_i \geq 0} \mathcal{L}(z, \theta, u_i)$. The dual is :

$$\min_{\theta \in \mathbb{R}, u_i \geq 0} \max_{z \geq 0} \mathcal{L}(z, \theta, u_i) \tag{6}$$

We easily show that (6) can be re-written as a Linear Optimization Problem :

$$\begin{aligned} & \min_{\theta \in \mathbb{R}, u_i \geq 0} k\theta + \sum_{i=1}^{n} u_i \\ \text{s.t.} \quad & u_i + \theta \geq 0 \quad \text{and} \quad u_i + \theta \geq (1 - y_i(x_i^\top w + b)) \quad \forall i = \{1, ..., n\} \end{aligned} \tag{7}$$

Finally, if we introduce the dualization inside (4) we obtain the **1-norm Soft Margin SVM as a Quadratic Optimization Problem**, with only linear constraints, and a quadratic objective function, which will be tractable at a large scale:

$$\begin{aligned} \min_{u,\theta,w} \quad & \mathbf{w}^\top \mathbf{w} + \lambda(k\theta + \sum_{i=1}^{n} u_i) \\ s.t. \quad & u_i + \theta \geq 0 \quad \text{and} \quad u_i + \theta \geq (1 - y_i(x_i^\top w + b)) \\ & u_i \geq 0 \quad (i = 1 \cdots n) \end{aligned} \tag{8}$$

The precedent reasoning can be applied to the 2-norm Soft Margin SVM, although we can see that we will no longer have a Quadratic Optimization Problem. Indeed, we will have a quadratic constraint, making this problem less tractable although we know that nowadays quadratic constraints are fairly well handled by commercial solvers such as Mosek, CPLEX or Gurobi.

2.4 Multi-Class Stable 1-Norm Soft-Margin SVM

A natural extension is the multi-class case (with K distinct classes) is the following :

$$\min_{w_j} \quad \sum_{j=1}^{K} \mathbf{w_j}^\top \mathbf{w_j} + \lambda \sum_{i=1}^{n} \sum_{j=1}^{K} |1 - y_{ij}(x_i^\top w_j + b)|_+ \tag{9}$$

Where we defined $\forall j \in \{1, ..., K\}$, $y_{ij} = 1$ if $y_i = k$, $y_{ij} = -1$ otherwise, or equivalently, $y_{ij} = 2 \cdot \delta_{\{y_i = j\}} - 1$. Its stable version can be written as follows :

$$\begin{aligned} \min_{u,\theta,w} \quad & \sum_{j=1}^{K} \mathbf{w_j}^\top \mathbf{w_j} + \lambda(k\theta + \sum_{i=1}^{n} u_i) \\ s.t. \quad & u_i + \theta \geq \sum_{j=1}^{K} \eta_{ij} \quad \text{and} \quad \eta_{ij} \geq (1 - y_{ij}(x_i^\top w_j + b)), \eta_{ij} \geq 0 \\ & u_i \geq 0 \qquad \forall i \in \{1, \cdots, n\}, \forall j \in \{1, \cdots, K\} \end{aligned} \tag{10}$$

3 A Stable Logistic Regression Formulation

Another well known scalar classification technique is the logistic regression. If we reuse the notations we used in section 2, the regression is usually performed by solving the following problem :

$$\min_w \quad \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp -y_i x_i^\top w) + \lambda \mathbf{w}^\top \mathbf{w} \tag{11}$$

The natural raw Stable formulation is the following :

$$\begin{aligned} \min_w \quad & \frac{1}{k} \max_z \sum_{i=1}^{n} z_i \log(1 + \exp -y_i x_i^\top w) + \lambda \mathbf{w}^\top \mathbf{w} \\ s.t. \quad & \sum_{i=1}^{n} z_i = k, \quad z_i \in \{0, 1\} \quad (i = 1 \cdots n) \end{aligned} \tag{12}$$

By applying the same dualization approach as for SVMs, we obtain the following formulation :

$$\begin{aligned} \min_w \quad & \frac{1}{k}(k\theta + \sum_{i=1}^{n} u_i) + \lambda \mathbf{w}^\top \mathbf{w} \\ s.t. \quad & u_i + \theta \geq \log(1 + \exp -y_i x_i^\top w) \\ & u_i \geq 0 \quad (i = 1 \cdots n) \end{aligned} \tag{13}$$

However, we can see that the obtained formulation leads to a quite intricate set of constraints. After some research, we have found [1] which performs a sparse classification (logistic regression) from which we now inspire ourselves to find a tractable algorithm to solve (12). Indeed we can rewrite (12) as a min-max problem:

$$\begin{aligned} \min_w \quad & f_k(w) + \lambda \mathbf{w}^\top \mathbf{w} \\ s.t. \quad & w \in \mathbb{R}^p \end{aligned} \tag{14}$$

Where:

$$f_k(w) = \frac{1}{k} \max_{z \in Z_k} \sum_{i=1}^{n} z_i \log(1 + \exp(-y_i x_i^\top w))$$

$$Z_k = \{z \in \{0,1\}^n s.t. \sum_{i=1}^{n} z_i = n\}$$

Here, f_k is a convex function of w as a maximum of a finite set of convex functions. Therefore, we are going to solve (14) through a cutting planes algorithm (**see Algorithm 1**). This algorithm solves the optimization problem (14) by approximating a convex function by the maximum of its tangents taken at specific points.

Data: Choose w_0 to initialize, λ the regularization parameter, ϵ the convergence parameter, set $t_0 = 0$, $J = 0$
Result: w_{opt}
initialization;
while $f_k(w_J) + \lambda w_J^\top w_J - t_J > \epsilon$ **do**
  solve (π_J) :
    $\min_w t$
    s.t. $t \geq 0$
    $t \geq f_k(w_j) + + \lambda w_j^\top w_j + (\nabla_w f_k(w_j) + 2\lambda w_j)^\top (w - w_j) \quad \forall j = 1 \cdots J$
  set t_{J+1}, w_{J+1} the solutions of the above problem
end

Algorithm 1: Cutting planes algorithm computing the Stable Logistic Regression

4 Empirical Results

For all of the following results, the data has been randomly split into 90/10 train/test sets, and then the train test has been separated with ratio $\alpha = k/n$ varying. For Stable implementations, the new training set consisted of the k worst errors, while for classic implementations, the training/validation sets were selected randomly but had the same ratio. We have considered two binary classification datasets from the UCI Machine Learning Repository [3] (Adult and WPBC) and one multi-class (Glass, with 6 classes).

4.1 Stable Support Vector Machines

Both Classic 1-norm soft-margin SVM (C-SVM) and Stable SVM (S-SVM) were implemented in Julia and the optimization problems were solved using Gurobi. We report the Test Accuracy (Acc.), the True Positive Rate (TPR) and False Positive Rate (FPR) of both methods on three datasets: Adult (binary), WPBC (binary) and Glass (multi-class).

Dataset	**Acc. C-SVM**	**Acc. S-SVM**	**TPR C-SVM**	**TPR S-SVM**	**FPR C-SVM**	**FPR S-SVM**
Adult	0.85	**0.86**	0.51	**0.52**	0.05	**0.04**
WPBC	0.85	0.85	0.4	0.4	0	0
Glass	0.68	0.68	N/A	N/A	N/A	N/A

Table 1: Classification test scores on UCI data sets, using all training data ($k = n$)

As we can see, there is no clear winner between both methods when we consider all the training data for both. We will now study how do Stable SVM's scores change with k. This hyperparameter allows to set how many worst errors the Stable SVM has to take into account in the loss function. In other words, it allows to adjust how strongly we penalize the worst errors.

Dataset	k	**Accuracy**	**TPR**	**FPR**
Adult	$\lfloor 0.2n \rfloor$	0.846	0.49	0.04
	$\lfloor 0.4n \rfloor$	0.860	0.522	0.05
	$\lfloor 0.6n \rfloor$	0.860	0.522	0.05
WPBC	$\lfloor 0.2n \rfloor$	0.85	0.4	0
	$\lfloor 0.4n \rfloor$	0.85	0.4	0
	$\lfloor 0.6n \rfloor$	0.85	0.4	0
Glass	$\lfloor 0.2n \rfloor$	0.545	N/A	N/A
	$\lfloor 0.4n \rfloor$	0.64	N/A	N/A
	$\lfloor 0.6n \rfloor$	0.68	N/A	N/A

Table 2: Stable SVM test scores' sensitivity to k

On the other hand, when considering only a portion of the training data, Stable SVMs are indeed extremely stable, as their performance is very quickly extremely close to the best performance they achieve with all the data (which can be seen in Table 2). This is mostly due to the fact that, when considering the worst errors in the objective function, we're actually fitting the SVM only on the support vectors. Namely, we always choose the support vectors first, which is absolutely not the case when randomly splitting the data, and this therefore adds a lot of stability to the performances.

Doing that also adds a lot of stability to the coefficients of the hyperplanes, therefore robustifying the model. The standard deviations have been computed using the hyperplanes' coefficients for values of k in $\{\lfloor 0.2n \rfloor, \lfloor 0.3n \rfloor, \lfloor 0.4n \rfloor, \lfloor 0.5n \rfloor, \lfloor 0.6n \rfloor, \lfloor 0.7n \rfloor, \lfloor 0.8n \rfloor, \lfloor 0.9n \rfloor, n\}$, and we defined the average coefficients' standard deviation (presented in Table 3) for binary classification as follows:

$$\overline{\sigma}^{bin}_{Dataset} = \sqrt{\frac{1}{p}\sum_{j=1}^{p}\sigma^2(w_j)} = \sqrt{\frac{1}{p}\sum_{j=1}^{p}\left(\sum_{k=1}^{K}(w_j^{(k)})^2 - \left(\frac{1}{K}\sum_{k=1}^{K}w_j^{(k)}\right)^2\right)}$$

And for multi-class (M classes) classification, given that now $w \in \mathbb{R}^{p\times M}$ with M hyperplanes:

$$\overline{\sigma}^{multi}_{Dataset} = \frac{1}{M}\sum_{m=1}^{M}\sqrt{\frac{1}{p}\sum_{j=1}^{p}\sigma^2(w_{jm})} = \frac{1}{M}\sum_{m=1}^{M}\sqrt{\frac{1}{p}\sum_{j=1}^{p}\left(\sum_{k=1}^{K}(w_{jm}^{(k)})^2 - \left(\frac{1}{K}\sum_{k=1}^{K}w_{jm}^{(k)}\right)^2\right)}$$

Dataset	**C-SVM Coefficient std**	**S-SVM Coefficient std**
Adult	1.554	**1.396**
WPBC	0.212	**0.179**
Glass	4.773	**2.157**

Table 3: Hyerplanes' coefficients standard deviations for different values of k

As we can see in Table 3, the Stable SVM formulation allows indeed a stabilization of the coefficients of the hyperplane(s) by reducing (drastically for the multi-classification case) the total average standard deviation, which justifies our willingness to study this method in the first place. The reduction in standard deviation ranged from 10% to 55%.

4.2 Stable Logistic Regression

In order to compare this new method to classical ones, we performed ℓ_1, ℓ_2 regularized and Elastic Net Logistic Regression. We first used 10-fold cross validation to find the ℓ_1 and ℓ_2 regularization hyperparameters. We then fitted the model on all the training data and computed the accuracy, the true positive rate (TPR) and false positive rate (FPR). For the Stable Regression, we solely fitted one hyperparameter, the regularization constant λ through 10-fold cross validation.

Dataset	**Acc. CLR**	**Acc. SLR**	**TPR CLR**	**TPR SLR**	**FPR CLR**	**FPR SLR**
Adult	0.85	0.87	0.85	0.73	0.20	0.10
WPBC	0.85	0.83	0.80	0.80	**0.17**	0.20

Table 4: Classification test scores on UCI data sets, $k = \left\lfloor \frac{n}{2} \right\rfloor$

- The out-of-sample accuracy scores of the Stable and Classical Logistic regression are quite similar. There is no "clear winner"
- In the Adult dataset case, the Stable Logistic Regression's FPR is half the Classical Logistic Regression's

We will now study how do Stable Logistic Regression's scores change with k. This hyperparameter allows to set how many worst errors the Stable Logistic Regression has to take into account in the loss function. In other words, it allows to adjust how strongly we penalize the worst errors. We will do this exclusively on the Adult data set.

k	**Accuracy**	**TPR**	**FPR**
$k = \lfloor 0.5n \rfloor$	0.87	0.73	0.09
$k = \lfloor 0.6n \rfloor$	0.83	0.70	0.13
$k = \lfloor 0.7n \rfloor$	0.80	0.82	0.20

Table 5: Stable SVM test scores' sensitivity to k on the Adult data set, $\lambda = 0.1$

- As k increases, *i.e.* we become less conservative, the accuracy decreases. We can then interpret that there is noise in the data, otherwise we would not be seeing this phenomenon
- The decrease in accuracy is explained by a significantly worse performance in terms of FPR for $k = \lfloor 0.7n \rfloor$. Again, the noise hypothesis can explain that. The noise makes it more difficult to finely classify negative outcomes.

As a conclusion, we can interpret the effect of k as follows. Diminishing k can help to decrease the effect of noise of "easily classifiable" data, allowing to achieve a finer classification.

Concerning the stability of the classifier returned by Stable Logistic Regression, we conducted a study of the standard deviation of the hyperplanes' coefficients w perfectly similar to the one done for the Stable SVM. In this case however, we will only study the stability for the binary classification. Again, from Table 6 we see that the Stable Logistic Regression is loyal to its name. For instance, we achieve a 200% reduction in the standard deviation of the classifiers for the Adult dataset.

Dataset	**C-Log Reg Coefficient std**	**S-Log Reg Coefficient std**
Adult	0.155	**0.059**
WPBC	0.212	**0.156**

Table 6: Classifiers' coefficients standard deviations for different values of k

This method thus leads to more stability and robustness while conserving or even improving the performance scores. Notice also that the results are even more conclusive for Stable Logistic Regression than for Stable Soft-Margin SVMs: this is due to the fact that Soft-Margin SVMs are already by construction somewhat robust (since we allow a slack ξ in order to be able to classify more points). But as we have shown, our formulation allows to robustify it even more.

5 Conclusion

As a conclusion, we have developed the theoretical framework of an extension of the Stable Regression developed in [1] to Stable Support Vector Machines and Stable Logistic Regression for Classification tasks. We have also implemented tractable algorithms to solve the resulting optimization problems, either using generic Linear Programming solvers or using a custom-made Cutting Planes algorithm. Finally, we have given substantial evidence of both how Stable versions keep the same level of performance as Classic implementations, while reducing the variability of the optimal coefficients (which depends mostly on the train/validation/test data splitting) drastically, therefore robustifying the models at hand. The next steps would be to evaluate the same metrics on many more datasets from the UCI ML Repository as a first basis, and then to other real-life datasets in order to establish the properties that we claim.

References

[1] Bertsimas, D., Pauphilet, J. and Van Parys, B. (2018) *Sparse Classification: a scalable discrete optimization perspective*, https://arxiv.org/pdf/1710.01352.pdf

[2] Bertsimas, D. and Dunn, J. (2019) *Machine Learning Under a Modern Optimization Lens*, Dynamic Ideas LLC, 2019, 333-366.

[3] Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.