Univerzitet u Novom Sadu
Prirodno-matematički fakultet
Departman za matematiku i informatiku

Luka Rutešić

The gradient sampling algorithm for solving binary classification problems

Master's thesis

2021, Novi Sad

Contents

I would like to express deepest gratitude towards my mentor Nataša Krejić for the valuable guidance and aid in this project. Her insightful feedback and help brought this work to a much higher level. I am grateful to professor Nataša Krklec Jerinkić for introducing me to the wonders of optimization and numerics, and sharing her knowledge. Also, many thanks to professor Dušan Jakovetić and professor Miloš Savić for helping in this marvelous journey. In addition, I would like to thank my mother Smiljana and father Jovica for being loving and caring parents, and Dunja for being my oasis.

Chapter 1

Introduction

Predicting the future started fascinating me from the early age. For centuries, people have been trying to find ways to deal with the forthcoming, and foresee the future. The motivation behind it is usually materialistic, however its charm might even be in the power that complements the ability to do so. What is more, that curiosity led to the marvelous results in the field of applied mathematics. Mathematical modelling emerged as a consequence of formalizing the nature, in an attempt to mathematize the world around us.

The task to formally and precisely describe natural phenomena is non-trivial. Firstly, there are many factors, whose existence is not as evident, and defining them would be extremely complicated, as we are not fully aware of their properties. There is also choosing the approach for solving a problem. Mathematics offers a variety of tools, so finding a path is a matter of preference. Same task can be modelled in plethora of ways, so the question arises: Which model is the best? The correct answer is usually unobtainable, however in the set of different descriptions of a problem, it is possible to distinguish the one (or multiple) that describes the problem the most efficiently.

With the advancement of technology and computers, we acquired the machinery to simulate the reality more precisely. Consequently, a breakthrough of new methods of modelling occured (one of which is machine learning) . The advent of internet networked the world, and created a demand for more robust and complex methods which are designed for nonlinear and multidimensional problems. Mathematical apparatus has received the application in a larger scale than ever before. Moreover, the models with mathematical background have become omnipresent in our everyday lives.

This tesis will focus on an optimization algorithm and implementing it in a machine learning model. The goal is to present the reasoning behind this method's functionality and formally prove the techniques used. Alongside the theoretical part, an industrial application is included. Namely the motivation of the thesis, besides the full implementation of the model, is to apply the algorithm to a real world issue. The industrial problem that will be analysed is introduced in the last chapter. Suffice it to say, the reader will witness the beautiful 'life cycle' of a real world mathematical problem, and the amount of theory that is necessary to and solve it.

Chapter 2

Machine learning

Machine learning (ML) studies algorithms which simulate human learning and experience. Using probability, statistics and numerical analysis it is possible to "teach" a computer, with a quantified certainty, how to predict, cluster or identify properties of an entity. These methods have found their application in various forms, in fields such as agriculture, banking, medical diagnosis, insurance, marketing etc, and they are still evolving to this day. Based on the nature of a problem, there are different variants of ML which can be used to tackle the problem, the most frequent being: supervised, unsupervised and reinforced learning. Supervised learning (which we will focus on) is used to extract the rules from the input data to determine the output information. The main goal of these algorithms is to predict a characteristic (target, label) of a sample, with the help of its attributes (features).

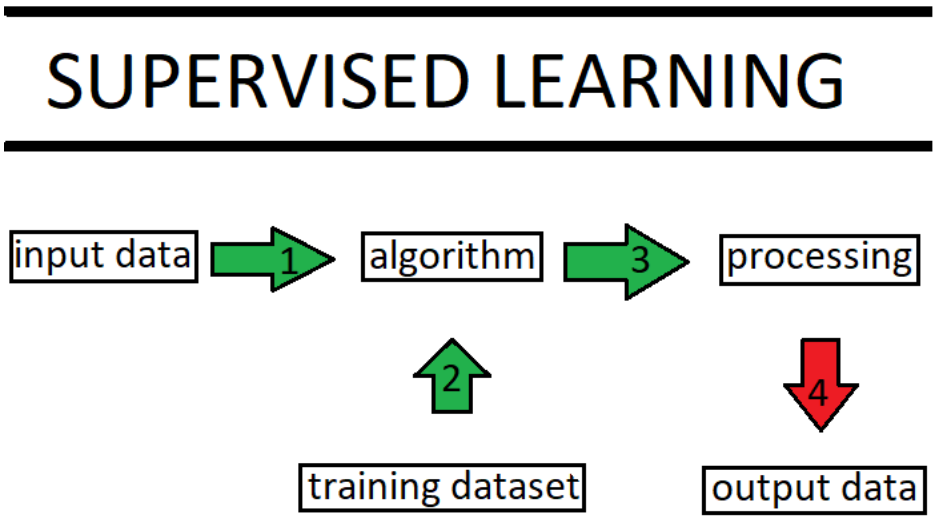

Figure 2.1: Scheme of supervised learning

Preprocessing the data is the first step in every method. Without modifying the data correctly, most of the algorithms would not be able to

train properly. The datasets contain rows of samples, where each column represents an attribute (feature), and every sample can have a target value, which we want to predict.

These datasets exist in various forms, hence the most common steps in the preprocessing are, dealing with the missing data, and encoding the nonnumerical features into real numbers. The rows with the missing data can either be deleted, or if there are too many of such rows, one can approximate the missing values. In order to encode the string features, it is a common practice to use dummy variables. Each feature, that has k different categories, is turned into k columns, where every column represents one category of the feature. The sample can have the value 1 or 0, depending on whether it belongs to the category. Similar technique can be applied to labels. If the target set is non ordered and non numerical, the targets can be encoded with either dummy variables, or a single column with 0 and 1 values, in the case of binary classification. Note that samples don't need to have labels, however in that case, you may want to apply clustering algorithms to find the clusters and make labels artificially.

The second step of supervised learning is splitting the data into two parts. The purpose of the first set (training set) is to fit the model, while the latter (test set) is used to validate the efficiency of the system. Scaling the data is also recommended after the split, however it is not always necessary for the algorithm to function properly. After training the model, the test data is processed, and with the usage of known features we receive the label predictions as the output values. We can then compare them to the real test labels, and approximately calculate the efficiency of the model.

The sample sets can have a numerical or categorical label, hence there are two types of algorithms, for regression and classification. Various methods have been developed with the same goals, that is to maximize the number of correct predictions and minimize the error. In order to achieve highest predicting performance, an objective (loss, cost) function is defined, which measures the distance of predicted from the true label values. Training the algorithm implies minimizing the loss, after which the optimal parameters are derived in order to improve the model. Nonetheless, there are various optimization methods, all of which are extremely reliant on the properties of the objective function.

As mentioned, datasets with categorical labels require classification methods, due to the fact that they use cost functions designed for binary labels. Specifically, binary classification is applied if there are only two label categories. For the algorithms to evaluate the predictions correctly, numerous loss functions are employed, one of which is hinge loss. Namely, the machine learning method we will discuss is the support vector machine (SVM). Its cost function, hinge loss, is going to be analysed and we will prove model's functionality .

Chapter 3

Hinge loss

3.1 SVM for binary classification

Hinge loss is a cost function used for training binary classifiers. The most noteworthy example is its implementation in the support vector machines (SVM). Namely, the goal of the (linear) SVM algorithm is to detect an optimal hyperplane in the n-dimensional space (where n is the number of features) which separates the two classes of points. Note that the hyperplane is a $n-1$ dimensional subspace for the n-dimensional space. In $\mathbb{R}^3$ it is a plane, whereas in $\mathbb{R}^2$ it is a straight line. Figure 3.1 shows separating hyperplanes, for a 2 dimensional dataset.

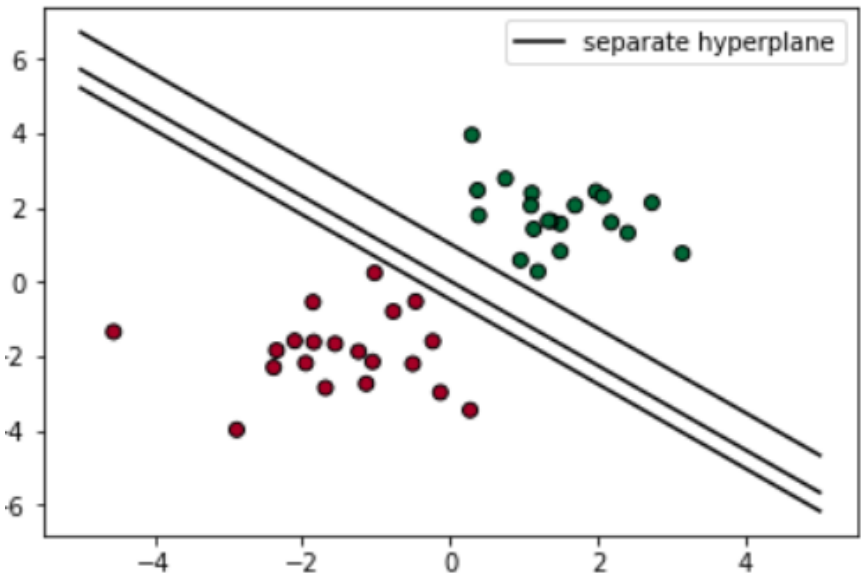

Figure 3.1: Separating hyperplane in $\mathbb{R}^2$, Source: [TDS]

As we can see, there are more than a single separating line, however the algorithm will search for the one that has the widest margin. We define the margin of a hyperplane as the minimal distance from the hyperplane to the dataset points. The points laying on it are called support vectors. Resulting maximized hyperplane can be seen in Figure 3.2.

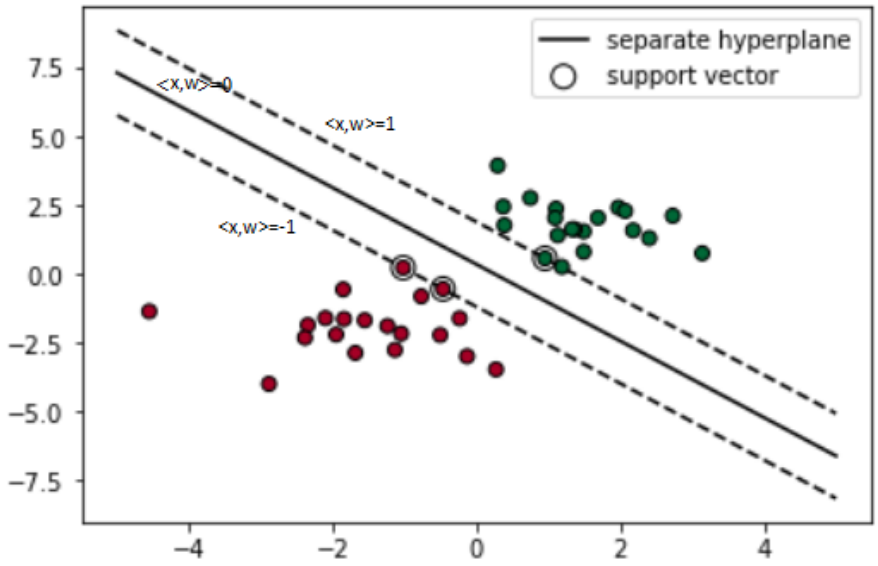

Figure 3.2: Support vectors and optimized margin in $\mathbb{R}^2$, Source: [TDS]

The width of the margin can be calculated as: $\frac{2}{\|x\|}$, where x is the coefficient vector of the hyperplane. To maximize it, we need to minimize $\|x\|$. What is more, based on two classes, we have the following constraints:

$$\langle x, \omega_i \rangle \geq 1, \text{ if } t_i = +1, i = 1, ..., K \tag{3.1}$$

$$\langle x, \omega_i \rangle \leq -1, \text{ if } t_i = -1, i = 1, ..., K \tag{3.2}$$

where K is the number of points in the plane (samples), $\omega_i, i = 1, ..., K$ are the sample attribute vectors, and $t_i \in \{-1, 1\}$ is the target value of the sample ω_i. Here, we denote the scalar product of vectors as $\langle \cdot, \cdot \rangle$. These constraints can be simplified in the form of one inequality:

$$t_i * \langle x, \omega_i \rangle \geq 1, i = 1, ..., K \tag{3.3}$$

We get the following constrained quadratic optimization problem:

$$\min_x \frac{1}{2}\|x\|^2, \text{ subject to } t_i * \langle x, \omega_i \rangle \geq 1, i = 1, ..., K \tag{3.4}$$

which has a unique solution. Since we are minimizing a convex function with an affine constraint function, this problem is convex, for which we know that

local solutions are global. This case is called a 'hard margin', and it is applicable only in the linearly separable dataset.

When the dataset is not linearly separable, which in reality is most often the case, SVM applies what is called a 'soft margin'. That is, when the data is being misclassified, the penalty function establishes slight tolerance for mistakes. In other words, this problem of misclassification is solved by 'softening' the constraint by employing the hinge loss with an L_2 regularization factor. We will define the L_2 regularized binary hinge loss as $f : \mathbb{R}^n \to \mathbb{R}$,

$$f(x) := \frac{\lambda}{2}\|x\|^2 + \frac{1}{K}\sum_{i=1}^{K} l(t_i, x, \omega_i), \tag{3.5}$$

where $K \in \mathbb{N}$ is the number of samples, $\lambda > 0$ a regularization parameter, $\omega_i \in \mathbb{R}^n, i = 1, ..., K$ are the feature vectors of each sample, and $t \in \mathbb{R}^K$ is the vector of corresponding real labels, with $t_i \in \{-1, 1\}$. Independent variable $x \in \mathbb{R}^n$ represents the coefficient vector of the hyperplane, which we will need to find to achieve the optimal setting. Function $l(t_i, x, \omega_i)$ is the basic hinge loss defined as:

$$l(t_i, x, \omega_i) = \max(0, 1 - t_i * \langle x, \omega_i \rangle), t_i \in \{-1, 1\}. \tag{3.6}$$

More common in literature, the hinge loss (3.6) can be written as:

$$l(t, y) := \max(0, 1 - t * y), t \in \{-1, 1\}, y \in \mathbb{R}.$$

where $y = \langle x, \omega_i \rangle$ in equation (3.6).

Figure 3.3 shows the graph of the function for $t = 1$. It can be seen that the error penalization in the objective function increases the closer the y is to 0, and the further y is from t, on the opposite side of 0, the lesser the loss is. This intuitively satisfies the constraint (3.3) mentioned previously.

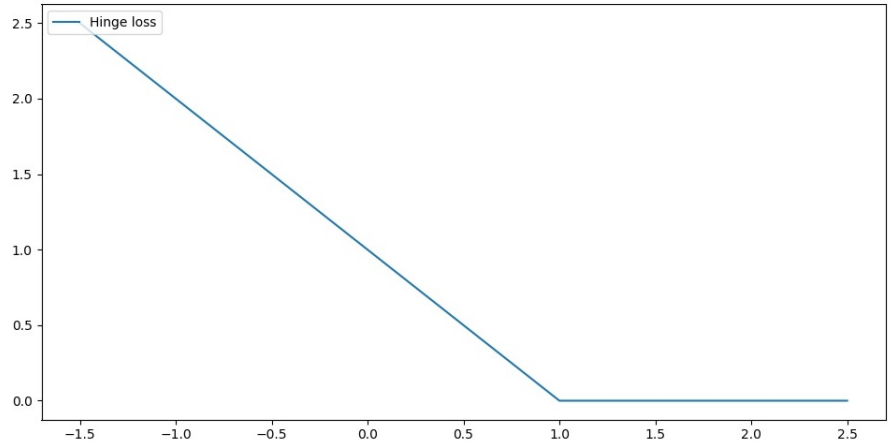

Figure 3.3: Hinge loss for t=1

We can notice that the hinge loss with the L_2 regularization (3.5) is constructed out of two parts. The first one $\frac{\lambda}{2}\|x\|^2$ is the regularizator, which has the task to decrease the importance of misclassification. The second part, however, is meant to measure the mean distance of all predictions from real label values. Suffice it to say, as λ gets larger, the latter part has less of an impact on the loss.

Additionaly, in the case when $|\langle x, \omega_i \rangle| \geq 1$ the cost function does not increase, whereas if $|\langle x, \omega_i \rangle| \leq 1$ the loss starts to build up.

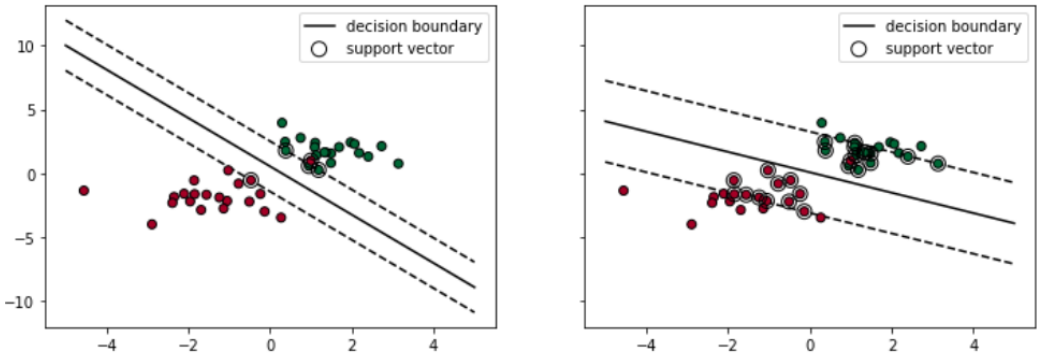

Figure 3.4: Hard margin (left) and Soft margin (right), Source: [TDS]

In order to find the appropriate coefficients for classifying the samples the following optimization problem is solved:

$$\min_x f(x) = \min_x (\frac{\lambda}{2}\|x\|^2 + \frac{1}{K}\sum_{i=1}^{K} l(t_i, x, \omega_i)). \tag{3.7}$$

The hinge loss function is not differentiable on its entire domain. However, we can calculate the subderivative. Firstly, we will need to define the index sets as:

$$\mathbb{E} = \{i \in \{1, ..., K\}, 1 - t_i\langle x, \omega_i\rangle > 0\} \tag{3.8}$$

$$\mathbb{M} = \{i \in \{1, ..., K\}, 1 - t_i\langle x, \omega_i\rangle = 0\} \tag{3.9}$$

$$\mathbb{W} = \{i \in \{1, ..., K\}, 1 - t_i\langle x, \omega_i\rangle < 0\} \tag{3.10}$$

Sets $\mathbb{E}, \mathbb{M}, \mathbb{W}$ represent the indexes of samples that are *within*, *on* and *outside* of the margin respectively. Consequently, the subderivative is defined as:

$$\partial f(x) = \lambda x - \frac{1}{K}\sum_{i=1}^{K} \beta_i t_i \omega_i = \lambda x - \frac{1}{K}\sum_{i\in\mathbb{E}} t_i\omega_i - \frac{1}{K}\sum_{i\in\mathbb{M}} \beta_i t_i \omega_i. \tag{3.11}$$

$$\beta_i = \begin{cases} 1 & i \in \mathbb{E} \\ [0, 1] & i \in \mathbb{M} \\ 0 & i \in \mathbb{W} \end{cases} \tag{3.12}$$

The hinge loss can also be used as a cost function for outlier detection algorithms.

3.2 SVM for anomaly detection

The datasets are often filled with imbalanced samples that stand out from the rest, and should be extracted. The anomaly (outlier) detection is a convenient process which is used to discover and eliminate such data. In industry, detecting anomalies within the operating systems can be beneficial, as we can predict the malfunctioning of the equipment and mitigate the potential losses. These outliers can be detected in many ways, however, different methods may not discover the same anomalies within the set. One of such methods is the 'one class SVM', which is an unsupervised machine learning method. As the name suggests, the notion is similar to the binary

classification SVM, where we are trying to find the optimal hyperplane to classify the data. However, one class SVM doesn't use labels for training, hence it is considered to be an unsupervised algorithm. The algorithm separates the points from the origin by finding the separating hyperplane with the largest margin based on the features. In Figure 3.5, we can see that the separating line defines the anomaly boundary.

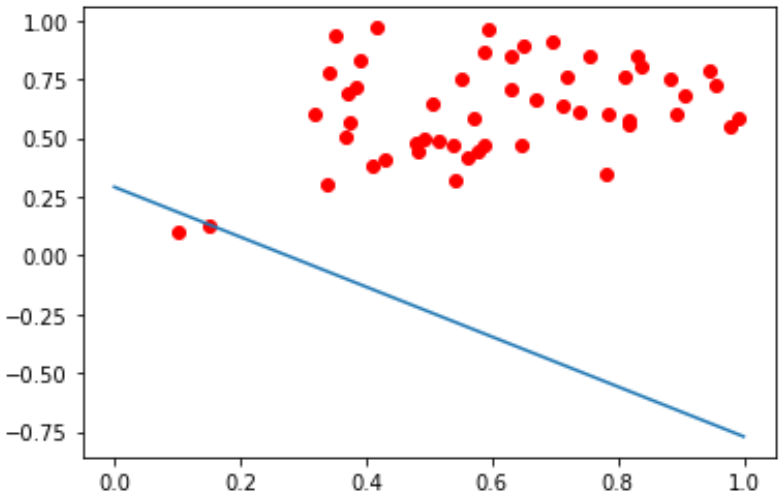

Figure 3.5: One class SVM hyperplane for a 2-dimensional set

To find the optimal hyperplane we solve the following optimization problem:

$$\min_{x,r} \frac{1}{2}\|x\|^2 - r, \text{ subject to } |\langle x, \omega_i \rangle| \geq |r|, i = 1, ..., K \tag{3.13}$$

The samples ω_i for which $|\langle x, \omega_i \rangle| \leq |r|$ are considered to be anomalies. Adding the regularization parameter and the hinge loss penalization, we can rewrite the problem as:

$$\min_{x,r} f(x, r) = \min_{x,r} \Big(\frac{\lambda \|x\|^2}{2} - \lambda r + \frac{1}{K} \sum_{i=1}^{K} \max\{0, r - \langle x, \omega_i \rangle\} \Big), \tag{3.14}$$

We can also find the subgradient of $f(x, r)$, as:

$$\begin{aligned}
\partial_x f(x,r) &= \lambda x - \frac{1}{K}\sum_{i=1}^{K}\beta_i\omega_i = \lambda x - \frac{1}{K}\sum_{i\in\mathbb{E}'}\omega_i - \frac{1}{K}\sum_{i\in\mathbb{M}'}\beta_i\omega_i \\
\partial_r f(x,r) &= -\lambda + \frac{1}{K}\sum_{i\in\mathbb{E}'}1 + \frac{1}{K}\sum_{i\in\mathbb{M}'}\beta_i \\
\partial f(x,r) &= [\partial_x f(x,r)\ \partial_r f(x,r)]
\end{aligned} \tag{3.15}$$

where,

$$\mathbb{E}' = \{i \in \{1, ..., K\}, r - \langle x, \omega_i\rangle > 0\} \tag{3.16}$$

$$\mathbb{M}' = \{i \in \{1, ..., K\}, r - \langle x, \omega_i\rangle = 0\} \tag{3.17}$$

$$\mathbb{W}' = \{i \in \{1, ..., K\}, r - \langle x, \omega_i\rangle < 0\} \tag{3.18}$$

and β_i is as in (3.12)

The optimization problems (3.7) and (8.2), require minimizing a nonsmooth function. The traditional minimization methods have trouble in dealing with large and sparse datasets, thus the stochastic method is proposed. To solve this, we are going to use the gradient sampling for nonsmooth functions, which was first introduced by Burke, Lewis and Overton [BLO], and later revisited and upgraded by Kiwiel [KIW]. However, we will need to show its functionality and prove the convergence first. In order to do so, basic theory of nonsmooth analysis needs to be introduced.

Chapter 4

The nonsmooth analysis

4.1 Generalized gradient

The terms and ideas presented in this chapter are mostly from [CLA], and the notation is directly taken from it. The rest is adapted to be in the spirit of Clarke's work.

Suppose $(X, \|\cdot\|)$ is a Banach space, where X is a nonempty set, containing elements x (vectors).

Definition 1. Let $Y \subseteq X$, and $f : Y \to \mathbb{R}$. We say that f is rank K Lipschitz continuous if, for a nonnegative scalar K holds:

$$|f(y_1) - f(y_2)| \leq K\|y_1 - y_2\|, \text{ for all } y_1, y_2 \in Y. \tag{4.1}$$

This condition, intuitively speaking, means that the function is not 'too steep'. Notice that the function doesn't need to be differentiable in order to be Lipschitz continuous, however all differentiable functions are Lipschitz continuous. For example $f = |x|$ is Lipschitz continuous, but not differentiable at $x = 0$.

We say that f is locally Lipschitz (near x), if for some $\epsilon > 0$, f satisfies the condition 4.1 on the ball $B(x, \epsilon) := \{x', \|x - x'\| < \epsilon\}$. This leads us to the term *generalized directional derivative.*

Definition 2. Let f be locally Lipschitz at x, and let v be a vector in X different from x. We define the generalized directional derivative of f at x, denoted as $f^o(x; v)$, in the following way:

$$f^o(x; v) = \limsup_{y \to x, t \to 0} \frac{f(y + tv) - f(y)}{t} \tag{4.2}$$

where $y \in X$, and $t > 0$.

This definition does not imply the existence of any limit, because it involves only the behaviour of f near x. It also differs from traditional def-

inition of directional derivative, as the base point y is not fixed.

Theorem 1. [CLA] Let f be a rank K locally Lipschitz continous function at x.

i) The function $v \to f^o(x;v)$ is finite, positively homogenous, and subadditive on X, satisfying:

$$|f^o(x;v)| \leq K\|v\|$$

ii) $f^o(x;v)$ is upper semicontinuous as a function of (x,v) and, is rank K Lipschitz on X in terms of v alone.

iii) $f^o(x;-v) = (-f)^o(x;v)$

Proof. From Lipschitz inequality, we know that:

$$|\frac{f(y+tv)-f(y)}{t}| \leq K\frac{\|y+tv-y\|}{t} = K\|v\|$$

As $|f^o(x;v)|$ is the lowest upper bound, it is bounded by $K\|v\|$. This proves the part (i).
Positive homogeneity comes from:

$$f^o(x;\lambda v) = \limsup_{y\to x, t\to 0} \frac{f(y+\lambda tv)-f(y)}{t} = \lambda \limsup_{y\to x, p\to 0} \frac{f(y+pv)-f(y)}{p}.$$

We get the subadditivity from the following:

$$\begin{aligned} f^o(x;v+w) &= \limsup_{y\to x,t\to 0} \frac{f(y+tv+tw)-f(y)+f(y+tw)-f(y+tw)}{t} \\ &\leq \limsup_{y\to x,t\to 0} \frac{f(y+tv+tw)-f(y+tw)}{t} + \limsup_{y\to x,t\to 0} \frac{f(y+tw)-f(y)}{t} \\ &= f^o(x;v) + f^o(x;w) \end{aligned} \tag{4.3}$$

which proves i).

For ii), we will need sequences $\{x_i\}$ and $\{v_i\}$ which converge to x and v respectively. Thus, for each i, there exists $y_i \in X$, $t_i > 0$, such that if:

$$\|y_i - x_i\| < \frac{1}{i} + t_i, \text{ then}$$

$$\begin{aligned} f^o(x;v) - \frac{1}{i} \leq f^o(x_i;v_i) &= \frac{f(y_i + t_i v_i) - f(y_i)}{t_i} \\ &= \frac{f(y_i + t_i v) - f(y_i)}{t_i} + \frac{f(y_i + t_i v_i) - f(y_i + t_i v)}{t_i} \\ &\leq \frac{f(y_i + t_i v) - f(y_i)}{t_i} + K\|v_i - v\|. \end{aligned} \tag{4.4}$$

which, after applying the upper limit (as $i \to \infty$), takes the form:

$$\limsup_{i\to\infty} f^o(x_i;v_i) \leq f^o(x;v).$$

Hence the upper semicontinuity has been established.

Finally, for $v, w \in X$ we have (from Lipschitz inequality):

$$f(y + tv) - f(y) \leq f(y + tw) - f(y) + K\|v - w\|,$$

for y near x, and t approaching 0. After dividing by t and applying upper limits ($\limsup_{y\to x, t\to 0} f$) we get:

$$f^o(x;v) \leq f^o(x;w) + K\|v - w\|.$$

This implies Lipschitz continuity for the second argument, and proves ii). The following holds:

$$\begin{aligned} f^o(x;-v) &= \limsup_{x'\to x, t\to 0} \frac{f(x' - tv) - f(x')}{t} \\ &= \limsup_{u\to x, t\to 0} \frac{(-f)(u + tv) - (-f)(u)}{t}, \text{where } u = x' - tv. \\ &= (-f)^o(x;v) \end{aligned} \tag{4.5}$$

■

The following theorems, which we will not prove, will give us the tools to define the generalized gradient. However, first we need to define the space of linear functionals.

Definition 3. Let $(X, \|\cdot\|_X)$, and $(Y, \|\cdot\|_Y)$ be normed spaces over the same field F. Function $f : X \to Y$ is linear, if for each $\alpha, \beta \in F$, and $x, y \in X$

$$f(\alpha x + \beta y) = \alpha f(x) + \beta f(y). \tag{4.6}$$

The set of all continuous linear functions from X to Y is denoted as $\mathbb{L}(X, Y)$.

Definition 4. Let X be a vector space over F. Linear function $f : X \to F$ is called a linear functional on X.

The set $\mathbb{L}(X, Y)$ is a vector space over F, if the addition of the vectors $f, g \in \mathbb{L}(X, Y)$, and multiplication by scalar $\alpha \in F$ is defined as:

$$(\alpha f + \beta g)(x) = \alpha f(x) + \beta g(x), \text{ for every } x \in X. \tag{4.7}$$

If $Y = \mathbb{R}$, then the space $\mathbb{L}(X, R)$ is Banach, and we call it the dual space of X, denoted as X'.

Definition 5. Linear function $f : X \to Y$ is bounded if there exists $M > 0$, such that

$$\|f(x)\|_Y \leq \|x\|_X, \text{ for every } x \in X. \tag{4.8}$$

Theorem 2. [HP] Linear function $f : X \to Y$ is continuous iff it is bounded.

This lets us formulate the Hahn-Banach theorem.

Theorem 3 (Hahn-Banach,[HP])**.** Let X be a vector space over $\mathbb{R}$, and p a subadditive and positively homogenous functional on X. Let A be a nonempty subspace of X and $f : A \to \mathbb{R}$ a linear functional for which holds

$$f(x) \leq p(x), \text{ for every } x \in A.$$

then, there exists a linear functional $F \in X'$ satisfying:

$$F_{|A} = f, F(x) \leq p(x), \text{ for every } x \in X. \tag{4.9}$$

where $F_{|A}$ is the restriction of F over A,

The Hahn-Banach Theorem claims that any positively homogenous and subadditive functional on X has at least one linear functional on X which it majorizes. Therefore, with the results of Theorem 1, a linear functional $\xi : X \to R$ exists, such that for every vector $v \in X$, $f^o(x, v) \geq \xi(v)$ holds. Note that ξ is bounded, thus belonging to the dual space X', of continuous linear functionals on X. We will denote the elements of X' as $\xi(v) = (\xi, v)$. This notation was used in [CLA], and it is similar to the one of the scalar product's. As the similarity isn't only of the visual kind, it is left this way.

Definition 6. We can define the generalized gradient of f at x as a subset of X^*:

$$\partial f(x) = \{\xi \in X^* : f^o(x; v) \geq (\xi, v) \text{ for all } v \text{ in } X\} \tag{4.10}$$

The norm of the dual space is defined in the following way:

$$\|\xi\|_* = \sup\{(\xi, v) : v \in X, \|v\| \leq 1\} \tag{4.11}$$

Next theorem states some basic properties of the set.

Theorem 4. [CLA] Let us assume f is a rank K Lipschitz continuous function (near x). Then:

i) $\partial f(x)$ is a nonempty, convex set and $\|\xi\|_* \leq K$ for every $\xi \in \partial f(x)$

ii) For every point $v \in X$,

$$f^o(x; v) = \max\{(\xi, v) : \xi \in \partial f(x)\}. \tag{4.12}$$

Proof. The nonemptiness follows from the Hahn-Banach theorem, whereas the convexity can easily be shown. If we take $\xi, \theta \in \partial f(x)$ then:

$$f^o(x; v) \geq \max\{(\xi, v), (\theta, v)\} \geq (\lambda\xi + (1 - \lambda)\theta, v), \text{ for all } v \in X, \lambda \in [0, 1]$$

Thus $\lambda\xi + (1 - \lambda)\theta \in \partial f(x)$. The upper bound we get from Theorem 1.

To prove ii), we suppose that for some $v \in X$, $f^o(x; v)$ exceeds the maximum, (4.12). The version of the Hahn-Banach theorem states that, there is a linear functional ξ, which is majorized by $f^o(x; \cdot)$, and it equals f at the point v. From that proposition, ξ belongs to $\partial f(x)$, and we get a contradiction: $f^o(x; v) > f^o(x; v)$, hence confirming the theorem. ■

The following lemma shows the equivalency of the relations of dual sets and the respective support functions.

Lemma 1. [CLA] Suppose that sets $\Sigma, \Omega \subset X^*$ are closed and convex. Then

$$\Sigma \subset \Omega \text{ iff } \sup\{(\xi, x) : \xi \in \Sigma\} \leq \sup\{(\eta, x) : \eta \in \Omega\}. \tag{4.13}$$

for all $x \in X$. Function $\sigma(x) = \sup\{(\xi, x) : \xi \in \Sigma\}$ is also called a support function

Proof. The first direction is trivial. The second direction is derived from the proposition that in a locally convex space, each closed convex set is an intersection of all the half-spaces which contain him.

■

Useful property of the generalized gradient $\partial f(x)$ is that it reduces to the standard derivative if f is C^1, and to the subdifferential of convex analysis, in the case when f is convex. The set $\partial f(x)$ can be written in different forms, one of which we will show. First let us recall the standard definition of a directional derivative.

Definition 7. Let $F : (X, \|\cdot\|_X) \to (Y, \|\cdot\|_Y)$, where X and Y are Banach spaces. Directional derivative of F at x in the direction v is

$$F'(x; v) := \lim_{t \to 0} \frac{F(x + tv) - F(x)}{t} = (DF(x), v). \tag{4.14}$$

when the limit exists.

We say that F has a Gâteaux derivative at x , denoted as $DF(x)$, if for every $v \in X$, $F'(x; v)$ exists. Additionaly, Gâteaux derivative is an element of the continuous linear functionals, the space $\mathbb{L}(X, Y)$. Connection between the functional Df and function f can be seen in the following proposition:

Theorem 5. [CLA] Let f be locally Lipschitz at x and admit a Gâteaux derivative $Df(x)$. Then $Df(x) \in \partial f(x)$.

Proof. From the definition, $f'(x; v) = (Df(x), v)$. We know that $f'(x; v) \leq f^o(x; v)$, thus $f^o(x; v) \geq (Df(x), v)$. This, together with the definition of $\partial f(x)$, implies the required inclusion $Df(x) \in \partial f(x)$.

■

For the following statement, we will first need to recall the famous Boltzano Weierstrass theorem.

Theorem 6 (Boltzano Weierstrass)**.** Every bounded sequence in $\mathbb{R}^n$ has a convergent subsequence.

Further, an equivalent definition of $\partial f(x)$ is given.

Theorem 7. [CLA] Let f be Lipschitz continuous near x, and suppose D is an open dense subset of $\mathbb{R}^n$ where f is differentiable. Then

$$\partial f(x) = \text{conv}\{\lim \nabla f(x_i) : x_i \to x, x_i \in D\}, \tag{4.15}$$

where conv$(\cdot)$ denotes the convex hull of a set. The convex hull of the set is the unique minimal convex set containing it.

Proof. From Theorem 4 (i), ∂f is locally bounded near x, and from Theorem 5 we have $\nabla f(x_i) \in \partial f(x_i)$. Hence, from Bolzano-Weierstrass Theorem, there exists a convergent subsequence $\{\nabla f(x_i')\}$ of the sequence $\{\nabla f(x_i)\}$. The limit of such subseqence remains in $\partial f(x)$, which we need to show. It is known from the definition of $\partial f(x_i)$ that $f^o(x_i, v) \geq (\nabla f(x_i'), v)$, and from Theorem 1 (the semicontinuity of f^o) that $f^o(x, v) \geq (\lim \nabla f(x_i'), v)$. This implies the inclusion $\lim \nabla f(x_i') \in \partial f(x)$. Hence, it follows that the set:

$$\{\lim \nabla f(x_i) : x_i \to x, x_i \in D\} \tag{4.16}$$

is a subset of $\partial f(x)$. It is nonempty, bounded and closed, hence it is a compact set. As $\partial f(x)$ is convex, the right hand side of 4.15 is contained in the $\partial f(x)$. The second inclusion will follow from Lemma 1, after we show that the $f^o(x; v)$ doesn't exceed the support function of the right hand side (as $f^o(x; v)$ is the support function of $\partial f(x)$) . To achieve this we will need another lemma.

Lemma 2. [CLA] For any $v \in \mathbb{R}^n$, and $\epsilon > 0$,

$$f^o(x; v) - \epsilon \leq \limsup\{\nabla f(y) \cdot v : y \to x, y \in D\} =: \alpha. \tag{4.17}$$

Proof. From the definition of the upper limit, there exists $\delta > 0$, such that the condition $y \in B(x, \delta)$ implies $\nabla f(y) \cdot v \leq \alpha + \epsilon$. Since D is dense in $\mathbb{R}^n$, the measure of $D \cap B(x, \delta)$ differs from 0. Now, consider the line segments $L_y = \{y + tv : 0 < t < \frac{\delta}{2|v|}\}$. From Fubini's theorem, for almost every

$y \in B(x, \frac{\delta}{2})$, the function is differentiable. Thus

$$f(y+tv) - t(y) = \int_0^t \nabla f(y+sv) \cdot v ds$$

follows from the Fundamental theorem of calculus. Since we have that $|y + sv - x| < \delta$ for $0 < s < t$, $\nabla f(y+sv) \cdot v \leq \alpha + \epsilon$ holds, hence

$$f(y+tv) - f(y) \leq t(\alpha + \epsilon)$$

Consequently,

$$f^o(x; v) \leq \alpha + \epsilon$$

which completes the proof of Lemma 2, and Theorem 7.

■

Definition 8. We define ϵ-subdifferential of function f as

$$\partial_\epsilon f(x) := \text{conv}\partial f(B(x, \epsilon)). \tag{4.18}$$

The terms *stationarity* and *ϵ-stationarity* are introduced in the following way.

Definition 9. Let f be Lipschitz continuous near x. We say that x is stationary for f if $0 \in \partial f(x)$, and ϵ-stationary for f if $0 \in \partial_\epsilon f(x)$.

Now, as an introduction to Lebourg's theorem, we will demonstrate some basic calculus of $\partial f(x)$.

Theorem 8. [CLA] Let us assume that f is a Lipschitz continuous function near the point x.

i) If $s \in R$, then $\partial(sf)(x) = s\partial f(x)$.

ii) If x is a point where f reaches maximum or minimum, then $0 \in \partial f(x)$.

iii) If $f_i, i = 1, ..., n$ is a finite family of Lipschitz functions near x, then their sum $f = \sum f_i$ is also Lipschitz near x, and

$$\partial(\sum f_i)(x) \subset \sum \partial f_i(x).$$

Proof. We know that sf is Lipschitz near x. Additionaly, when s is non-negative $(sf)^o = sf^o$ holds, which would imply that $\partial(sf)(x) = s\partial f(x)$. For a negative scalar s, it is now sufficient to prove the case when $s = -1$. From 4.10, an element $\xi \in X^*$ belongs to $\partial(-f)(x)$ if and only if $(-f)^o(x; v) \geq (\xi, v)$, for all v. Theorem 1 (iii) allows us to write $f^o(x; -v) \geq (\xi, v)$. As ξ is a linear functional, from its homogeneity it follows that $(\xi, -v) = (-\xi, v)$, which implies that $f^o(x; -v) \geq (-\xi, v)$ and $-\xi \in \partial f(x)$. Consequently, $\xi \in -\partial f(x)$, thus proving i).

Since $\partial(-f) = -\partial f$, for (ii) it is enough to prove the instance when x is a local minimum. However, in that case, for any $v \in X$, the definition of a minimum implies that $f^o(x, v) \geq 0 = (0, v)$ (rate of change can only be positive near x). Hence, $0 \in \partial f(x)$.

To prove iii), first notice that the right-hand side of the inclusion $(\sum \partial f_i(x))$ represents a set of points ξ, obtained by the sum $\sum \xi_i$, where $\xi_i \in \partial f_i(x)$. For $n = 2$, we have to prove that $\partial(f_1 + f_2)(x) \subset \partial f_1(x) + \partial f_2(x)$, and the case $n = k$ will inductively follow. After applying Lemma 1, we need to prove the inequality: $(f_1 + f_2)^o(x; v) \leq f_1^o(x; v) + f_2^o(x; v)$. Furthermore, from the definition:

$$\begin{aligned}(f_1 + f_2)^o(x; v) &= \limsup_{y \to x, t \to 0} \frac{(f_1 + f_2)(y + tv) - (f_1 + f_2)(y)}{t} \\ &\leq \limsup_{y \to x, t \to 0} \frac{f_1(y + tv) - f_1(y)}{t} + \limsup_{y \to x, t \to 0} \frac{f_2(y + tv) - f_2(y)}{t} \\ &= f_1(x; v) + f_2(x; v)\end{aligned} \tag{4.19}$$

thus proving iii). ■

One very important property of the gradient set is given in the form of a mean value theorem. It will have a role in proving the functionality of the [KIW] optimization method. Foremost, a directional derivative can be denoted as:

$$f^o(x; v) = \limsup_{y \to x, t \to 0} \frac{f(y + tv) - f(y)}{t} = \max(\partial f(x), v). \tag{4.20}$$

Where, $(\partial f(x), v) =: \{(\xi, v), \xi \in \partial f(x), v \in X\}$

4.2 The Lebourg's mean-value theorem

Theorem 9 (**Lebourg**). [CLA] Let $x, y \in X$ be arbitrary points, and suppose that $f : X \to \mathbb{R}$ is a Lipschitz continuous function on an open set containing the line segment $[x, y]$. Then there exists a vector $u \in (x, y)$ such that

$$f(y) - f(x) \in (\partial f(u), y - x). \tag{4.21}$$

For the proof, the following lemma is introduced. Let us first denote the point $x + t(y - x)$ by x_t.

Lemma 3. [CLA] The function $g : [0, 1] \to \mathbb{R}$ defined by $g(t) = f(x_t)$ is Lipschitz on interval (0,1), and inclusion holds:

$$\partial g(t) \subset (\partial f(x_t), y - x). \tag{4.22}$$

Proof. The Lipschitz continuity of g is implied by the fact that f is Lipschitz. Since both sets from (4.22) are convex, knowing Lemma 1, we only need to prove

$$\max\{(\partial g(t), v)\} \leq \max\{(\partial f(x_t), v(y - x))\} \tag{4.23}$$

Notice that supremum became maximum, as functionals are linear, continuous and defined on the closed segment. Additionally, the left side of the ineqality equals $g^o(t; v)$. Hence

$$\begin{aligned} g^o(t; v) &= \limsup_{s \to t, \lambda \to 0} \frac{g(s + \lambda v) - g(s)}{\lambda} \\ &= \limsup_{s \to t, \lambda \to 0} \frac{f(x + (s + \lambda v)(y - x)) - f(x + s(y - x))}{\lambda} \\ (=) &\leq \limsup_{y' \to x_t, \lambda \to 0} \frac{f(y' + \lambda v(y - x)) - f(y')}{\lambda} \\ &= f^o(x_t; v(y - x)) = max(\partial f(x_t), v(y - x)). \end{aligned} \tag{4.24}$$

Proof of Lebourg's theorem. Observe the help function $\theta : [0, 1] \to \mathbb{R}$, defined by

$$\theta(t) = f(x_t) + t[f(x) - f(y)].$$

Notice that $\theta(0) = \theta(1) = f(x)$, which from continuity implies the existence

of a point $t \in (0,1)$, which minimizes or maximizes θ. By Theorem 8 (ii), we know that $0 \in \partial\theta(t)$.
From Theorem 8 (i), (iii) and Lemma 3, we have:

$$\begin{aligned} 0 \in \partial\theta(t) &= \partial(f(x_t) + t[f(x) - f(y)]) \\ &\subset \partial f(x_t) + t\partial[f(x) - f(y)] \\ &\subset (\partial f(x_t), y - x) + [f(x) - f(y)], \end{aligned} \tag{4.25}$$

which implies (if we take $u = x_t$)

$$\begin{aligned} 0 &= [f(x) - f(y)] + (\xi, y - x) \\ \Rightarrow & f(y) - f(x) \in (\partial f(u), y - x) \end{aligned} \tag{4.26}$$

where $\xi \in \partial f(u)$. This proves Lebourg's mean value theorem.

■

Some basic notion of convex analysis will be required, when proving the convergence of the algorithms. The Carathéodory's theorem follows.

4.3 Carathéodory's theorem

As one of the basic principles in convex analysis, Caratheodory's theorem has many applications. Intuitively, it states that any point in the convex hull of the set $C \subset \mathbb{R}^n$ (denoted as conv(C)), can be represented as the convex combination of at most $n + 1$ points in C.

Definition 10. Set C is convex if it cointains every line segment that connects each pair of points from the set:

$$x, y \subset C \Rightarrow \lambda x + (1 - \lambda)y \subset C \tag{4.27}$$

Definition 11. A point $x \in C$ is said to have a convex combination if there exists $k \in N$, $\omega_i > 0$ and $x_i \in C, i = 1, ..., k$, such that

$$x = \sum_{i=1}^{k} \omega_i x_i, \text{ and } \sum_{i=1}^{k} \omega_i = 1 \tag{4.28}$$

Definition 12. We define (equivalently) a convex hull of the set C , conv(C), as the set of all convex combinations of points in C.

Definition 13. Closure of the set C is defined as:

$$\mathrm{cl}(C) = \bigcap B(C, \epsilon) \tag{4.29}$$

where, $B(C, \epsilon) = \bigcup\{x : \|x - c\| \leq \epsilon, \text{for some } c \in C\}$.

Theorem 10 (**Carathéodory,[DU]**)**.** Let $x \in conv(C)$, where $C \subset \mathbb{R}^n$. Then there exists $n+1$ vectors (not necessarily distinct) $x_0, ..., x_n \in C$, such that $x \in conv(x_0, ..., x_n)$.

Proof. Knowing that x belongs to the convex hull of C, from the Definition 6, we can find $k \in N$, $\omega_i > 0$ and $x_i \in C, i = 0, ..., k$ such that

$$x = \sum_{i=0}^{k} \omega_i x_i, \text{ and } \sum_{i=0}^{k} \omega_i = 1.$$

If $k \leq n$ there is nothing more to prove. Suppose that $k > n$. Since the dimension of the space is n, we know that the vectors $x_1 - x_0, ..., x_k - x_0$ are linearly dependent. Thus, there exist scalars $\lambda_i, i = 1, ..., k$, which are not all zeros, and $\sum_{i=0}^{k} \lambda_i x_i = 0$ holds. Let us define $\lambda_0 = -\sum_{i=1}^{k} \lambda_i$. It follows that $\sum_{i=0}^{k} \lambda_i x_i = 0$, and for at least one index i, $\lambda_i > 0$. Now:

$$\begin{aligned} x &= \sum_{i=0}^{k} \omega_i x_i - \gamma * 0 \\ &= \sum_{i=0}^{k} \omega_i x_i - \gamma \sum_{i=0}^{k} \lambda_i x_i \\ &= \sum_{i=1}^{k} (\omega_i - \gamma \lambda_i) x_i \end{aligned} \tag{4.30}$$

If we choose $\gamma = \min\limits_{0 \leq i \leq k} \{\frac{\omega_i}{\lambda_i}, \lambda_i > 0\} = \frac{\omega_j}{\lambda_j}$, for some $j = 0, ..., k$, we have that

$$\omega_i + \gamma \lambda_i \geq 0, i = 0, ..., k \text{ with } \omega_j + \gamma \lambda_j = 0 \text{ and } \sum_{i=0}^{k} (\omega_i + \gamma \lambda_i) = 1.$$

So we can omit j-th element, and write x as a convex combination of k elements. This can be done until $k = n$. ■

4.4 Geometric representation

In this section, we will show some basic geometric concepts related to the gradient.

Definition 14. Let $C \subset X$. The distance function $d_C(\cdot) : X \to R$ is defined by

$$d_C(x) = \inf\{\|x - c\| : c \in C\}. \tag{4.31}$$

In the case when C is closed, we know that $x \in C$ iff $d_C(x) = 0$. The function d_C is globally Lipschitz, which the following theorem states.

Theorem 11. [CLA] The distance function d_C is globally Lipschitz on X (also known as a short map):

$$|d_C(x) - d_C(y)| \leq \|x - y\|, \tag{4.32}$$

Proof. By the definition of infimum, for $\epsilon > 0$, there exists $c \in C$, such that $d_C(y) \geq \|y - c\| - \epsilon$. Hence,

$$\begin{aligned} d_C(x) \leq \|x - c\| &\leq \|x - y\| + \|y - c\| \\ \|x - y\| + d_C(y) + \epsilon & \\ d_C(x) - d_C(y) &\leq \|x - y\| + \epsilon \end{aligned} \tag{4.33}$$

Similarly, we get $d_C(x) - d_C(y) \geq -(\|x - y\| + \epsilon)$, from which the desired statement follows.

■

Suppose that $x \in C$. We say that a vector $v \in X$ is tangent to C at x if $d_C^o(x; v) = 0$. More formally:

Definition 15. The set of all tangets to C at x is denoted as $T_C(x)$ and

$$T_C(x) = \{v \in \mathbb{R}^n : d_C^o(x; v) = 0, \} \tag{4.34}$$

where

$$d_C^o(x; v) = \limsup_{y \to x, t \to 0} \frac{d_C(y + tv) - d_C(y)}{t} \tag{4.35}$$

From Theorem 1, $T_C(x)$ is a closed and convex cone in X, and it always contains 0. We can define a normal cone, using the tangent set.

Definition 16. The set of vectors $\xi \in X^*$ is called a normal cone if its elements only form obtuse or right angles with the vectors from the tangent cone:

$$N_C(x) = \{\xi \in X^* : (\xi, v) \leq 0 \text{ for all } v \text{ in } T_C(x)\} \tag{4.36}$$

Figure 4.1 shows the normal and tangent cone for different type of points of a set. We can notice that none of the vectors from the normal cone form an acute angle with a tangent cone vector.

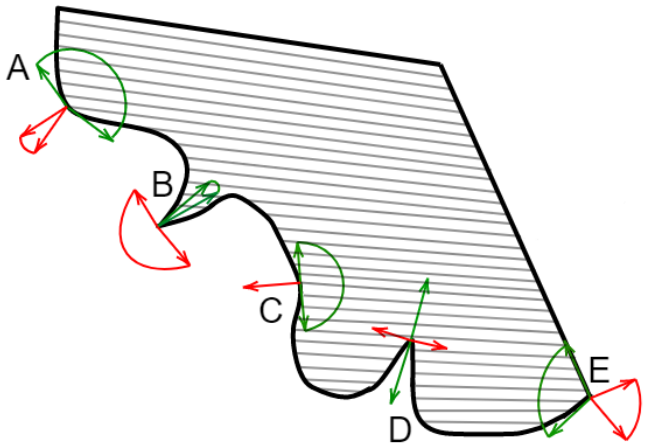

Figure 4.1: Five normal and tangent cones, red and green respectively

Suppose that $C \subset \mathbb{R}^n$ is a closed convex set. For a scalar product $\langle \cdot, \cdot \rangle$, and a vector $y \in C$, $\langle y, \cdot \rangle$ is a bounded linear functional on C, thus we can use the scalar product in the previous definition (instead of the general functional).

We define an ortogonal projection of a point z on the closed convex set C as:

$$proj(z|C) = \{y' : \|z - y'\| = \min_{y \in C} \|z - y\|\}. \tag{4.37}$$

This implies,

$$proj(0|C) = \{y' : \|y'\| = \min_{y \in C} \|y\|\} \tag{4.38}$$

Suppose that $\|y''\| = \min_{y \in C} \|y\|$. We know that $\|y''\| = d_C(0)$, which is the distance of vector 0 from the set C. Hence, we can write $y'' = proj(0|C)$. As y'' is the projection of 0 onto C, $y'' \in T_C$ holds, and we have $\langle y'', -y'' \rangle \leq 0$. This implies $-y'' \in N_C(y'')$, thus $\langle -y'', y - y'' \rangle \leq 0$, for every $y \in C$. We can conclude that:

$$\|y''\| = \min_{y \in C} \|y\| \Leftrightarrow \|y''\|^2 \leq \langle y, y'' \rangle, \text{ for every } y \in C \tag{4.39}$$

Based on these principles, the gradient sampling optimization algorithm was constructed by Burke, Lewis and Overton [BLO], and later enhanced by Kiwiel [KIW]. This algorithm minimizes nonsmooth functions, and can be used in mentioned optimization models. The next chapter will introduce the Kiwiel's method, some modifications, and prove the convergence of all versions.

Chapter 5

The Gradient Sampling method

The Burke, Lewis, Overton's algorithm minimizes a locally Lipschitz function $f : \mathbb{R}^n \to R$, which is continuously differentiable on an open dense subset $D \subset \mathbb{R}^n$, and has bounded level sets. At each iteration, it computes gradients of f at the current iterate and at $m \geq n+1$ points from the ϵ-radius ball. From these gradients, an optimal descent direction can be found, as the solution of a quadratic program. Armijo line search provides the next iterate candidate, by calculating the step size of the iteration. It does so by finding the maximum step in the chosen direction, such that the condition of the sufficient descent holds. In the case the new iterate doesn't belong to D, it is slightly perturbed, so that it preserves the Armijo condition.

There are two convergence results, based on whether the ϵ radius is constant or reduced dynamically. The first one states that with the probability 1, given the constant radius ϵ, the gradient sampling (GS) algorithm generates a sequence with a cluster point, which is ϵ-stationary. In the case when ϵ is reducing, they established that if the algorithm converges to a point, the limit of the sequence is a stationary point for f (with probability 1).

Kiwiel's results are however stronger. For a fixed ϵ, he shows that every cluster point generated by the GS algorithm is ϵ-stationary, almost always. Whereas, when ϵ is dynamically reduced, every cluster point of an arbitrary subsequence is stationary, without the assumption that the whole sequence converges. Evidently, the latter results are strictly better. In both cases, it will be showed that the algorithm terminates with probability 1.

Alongside these changes, Kiwiel made a revision of the algorithm, in which the perturbation of the Armijo candidate is limited by the step size, instead of the ϵ radius as before. This modification allows slightly weaker assumptions, however the results are strong. For a fixed ϵ, with probability 1, either the f-values go to $-\infty$, or every cluster point of a subsequence of iterates is ϵ-stationary. When ϵ is reduced, again, it is showed that either the iterates go to $-\infty$ or every cluster point they make is stationary.

There are three modifications of the GS algorithm in Kiwiel's work, with the intention to improve the practical performance. Since the search directions are normalized, as the algorithm converges, Armijo's line search can grow to infinity. Hence the different approaches are recommended, all of which will be covered.

5.1 The Algorithm

Firstly we will assume that the function $f : \mathbb{R}^n \to R$ is locally Lipschitz and continuously differentiable on an open dense subset of D of $\mathbb{R}^n$. We will use the second definition of the generalized gradient (also called Clarke's subdifferential),

$$\partial f(x) = \text{conv } \{\lim_{j\to\infty} \nabla f(y^j) \to x, y^j \in D\} \tag{5.1}$$

Similarily, a Clarke ϵ-subdifferential is defined as:

$$\partial_\epsilon f(x) := \text{ conv } \partial f(B(x,\epsilon)). \tag{5.2}$$

We can approximate this set with:

$$G_\epsilon(x) := \text{ cl conv } \nabla f(B(x,\epsilon) \cap D), \tag{5.3}$$

as it is clear that $G_\epsilon(x) \subset \partial_\epsilon f(x)$, and $\partial_{\epsilon_1} \subset G_{\epsilon_2}$ for $0 \leq \epsilon_1 < \epsilon_2$. Again, we say that x is stationary for f if $0 \in \partial f(x)$, and ϵ-stationary for f if $0 \in \partial_\epsilon f(x)$. The following is the Kiwiel's GS algorithm, which does not require compact level sets as the previous version.

GS algorithm [KIW]

Step 0 (the initialization). Initialize with $x^1 \in D$, optimality tolerances $\nu_{opt}, \epsilon_{opt} \geq 0$, line search parameters $\beta, \gamma \in (0,1)$, reduction factors $\mu, \theta \in (0,1]$, a sampling radius $\epsilon > 0$, a stationarity target $\nu_1 \geq 0$, and a sample size $m \geq n+1$. Set k:=1.

Step 1 (gradient sampling). Sample m points $\{x_i^k\}_{i=1}^m$ from $B(x^k, \epsilon_k)$. If $\{x_i^k\} \notin D$, stop. Otherwise, set the approximation of G_ϵ:

$$G_k := \text{conv}\{\nabla f(x^k), \nabla f(x_1^k), ..., \nabla f(x_m^k)\}. \tag{5.4}$$

Step 2 (direction search). Set $g^k := proj(0|G_k)$

Step 3 (stopping criterion). If $\|g^k\| \leq \nu_{opt}$ and $\epsilon_k \leq \epsilon_{opt}$, **stop**.

Step 4 (radius update). If $\|g^k\| \leq \nu_k$, set $\nu_{k+1} := \theta\nu_k$, $\epsilon_{k+1} := \mu\epsilon_k$, $t_k := 0$, $x^{k+1} := x^k$ and go to **Step 7**. Otherwise, set $\nu_{k+1} := \nu_k$, $\epsilon_{k+1} := \epsilon_k$,

and

$$d^k = -\frac{g^k}{\|g^k\|}. \tag{5.5}$$

Step 5 (Armijo line search). Find the step size with the Armijo search:

$$t_k := \max\{t : f(x^k + td^k) < f(x^k) - \beta t\|g^k\|, t \in \{1, \gamma, \gamma^2, ...\}\}. \tag{5.6}$$

Step 6 (differentiability check and update). If $x^k + td^k \in D$, set $x^{k+1} := x^k + t_k d^k$. Otherwise, let x_{k+1} be a point in D that satisfies:

$$f(x^{k+1}) < f(x^k) - \beta t_k \|g^k\|, \tag{5.7}$$

$$\|x^k + t_k d^k - x^{k+1}\| \leq \min\{t_k, \epsilon_k\} \tag{5.8}$$

Step 7 (start again). Increase k by 1, and go to **Step 1**.

Notice that the algorithm keeps the iterates inside the set D. What is more, from Step 2, $g^k = proj(0|G_k) \Leftrightarrow \|g^k\| = \min_{g \in G_k} \|g\|$, which implies $\|g^k\| = d_{G_k}(0)$. Thus, from equation (4.39), it is known that $\langle g, g^k \rangle \geq \|g^k\|^2$, for all $g \in G_k$. Since $\nabla f(x^k) \in G_k$ and (5.5), we have that:

$$\begin{aligned} \langle \nabla f(x^k), g^k \rangle &\geq \|g^k\|^2 \\ \langle \nabla f(x^k), -\|g^k\| d^k \rangle &\geq \|g^k\|^2 \\ -\langle \nabla f(x^k), d^k \rangle &\geq \|g^k\| \\ \langle \nabla f(x^k), d^k \rangle &\leq -\|g^k\|. \end{aligned} \tag{5.9}$$

Armijo line search is hence well defined, as we have chosen a valid descent direction. Therefore, $t' > 0$ exists, such that $f(x^k + td^k) < f(x^k) - \beta t\|g^k\|$, for all $t \in (0, t')$.

The main difference between Kiwiel's algorithm and Burke, Lewis and Overton's is the stronger requirement of Step 6 in the latter. In the case when $x^k + t_k d^k \notin D$, x^{k+1} can be found from a uniform distribution in $B(x^k + t_k d^k, \min\{t_k, \epsilon_k\}/i)$, for $i \in N$. By Armijo condition, and the continuity of f, this procedure terminates with probability 1. Contrasting, the original GS

algorithm requires to find $\hat{x}^k$ in $B(x^k, \epsilon_k)$, such that $x^{k+1} = \hat{x}^k + t_k d^k \in D$ and x^{k+1} satisfies (5.7). The desired iterate can be sampled from uniform distribution on $B(x^k, \epsilon_k/i)$ with the probability 1. Thus $\hat{x}^k := x^{k+1} - t_k d^k$ satisfies the original conditions, and explains the part of (5.8) with ϵ_k. The presence of t_k in (5.8) yields, with help from the reverse triangle inequality:

$$\left| \|x^k - x^{k+1}\| - t_k \|d^k\| \right| \leq \|x^k + t_k d^k - x^{k+1}\| \leq \min\{t_k, \epsilon_k\} \leq t_k \quad (5.10)$$

from where $\|x^{k+1} - x^k\| \leq 2t_k$ follows, knowing that $\|d^k\| = 1$. Thus, from 5.7

$$f(x^{k+1}) \leq f(x^k) - \beta \frac{1}{2} \|x^{k+1} - x^k\| \|g^k\|. \quad (5.11)$$

This inequality holds in both cases, when $x^{k+1} = x^k + t_k d^k$ and $x^{k+1} = x^k$. The algorithm stops when the values $\|g^k\|$ and $\|\epsilon_k\|$ are low enough, as only then are we guaranteed the closeness to Clarke's stationarity.

Chapter 6

Convergence analysis

The following two lemmas will have a major role in proving the convergence of the algorithm.

Lemma 4. [KIW] Let $\emptyset \neq C \subset \mathbb{R}^n$ be a compact convex set, and $\beta \in (0,1)$. If $0 \notin C$, there exists $\delta > 0$ such that $u, v \in C$ and $\|u\| \leq d_C(0) + \delta$ imply $\langle u, v\rangle > \beta\|u\|^2$.

Proof. Suppose the opposite, that is, we can pick two sequences $\{u^i\}, \{v^i\} \subset C$ satisfying $\|u^i\| \leq d_C(0) + 1/i$ and $\langle u^i, v^i\rangle \leq \beta\|u^i\|^2$. Since C is compact, limits stay in C, so $u^i \to \hat{u} \in C, v^i \to \hat{v} \in C$, and $\langle \hat{u}, \hat{v}\rangle \leq \beta\|\hat{u}\|^2$. The inequality $\|\hat{u}\| \leq d_C(0)$ and $0 \notin C$ imply that $\|\hat{u}\| = d_C(0)$, thus $\hat{u} = proj(0|C) \neq 0$. For such $\hat{u}$, from (4.39), $\langle \hat{u}, v\rangle \geq \|\hat{u}\|^2$ follows. Contradiction, as $\beta \in (0,1)$.

■

The second lemma will cover basic properties of the set of points close to a given point $\bar{x}$, which can be used to provide a δ-approximation of the least-norm element of $G_\epsilon(\bar{x})$. For $\epsilon, \delta > 0$ and $\bar{x}, x \in R$, we define a measure of proximity to $\epsilon-$stationarity as

$$\rho(\bar{x}) := d_{G_\epsilon(\bar{x})}(0), \tag{6.1}$$

which represents the distance of 0 from the Clarke's $\epsilon-$subdifferential. Let

$$D^m_\epsilon(x) := \prod_1^m (B(x,\epsilon) \cap D) \subset \prod_1^m \mathbb{R}^n, \tag{6.2}$$

$$\bar{G} = \bar{G}(\{y_i\}_1^m) = \text{conv}\{\nabla f(y^1), ..., \nabla f(y^m)\}, \tag{6.3}$$

$$V_\epsilon(\bar{x}, x, \delta) := \{(y^1, ..., y^m) \in D^m_\epsilon(x) : d_{\bar{G}}(0) \leq \rho_\epsilon(\bar{x}) + \delta\}. \tag{6.4}$$

Lemma 5. [KIW] Let $\epsilon > 0$ and $\bar{x} \in \mathbb{R}^n$. Then the following holds:

(i) For any $\delta > 0$, there is $\tau > 0$ and a nonempty open set $V \subset V_\epsilon(\bar{x}, x, \delta)$ for all $x \in B(\bar{x}, \tau)$, and $d_{\bar{G}}(0) \leq \rho_\epsilon(\bar{x}) + \delta$ for all $(y^1, ..., y^m) \in V$.

(ii) Assuming $0 \notin G_\epsilon(\bar{x})$, pick $\delta > 0$ as in Lemma 4 for $C := G_\epsilon(\bar{x})$, and τ, V as in statement (i). Suppose that at iteration k of GS algorithm, Armijo search is reached (Step 5) with $x^k \in B(\bar{x}, \min\{\tau, \epsilon/3\})$, $\epsilon_k = \epsilon$ and $(x_1^k, ..., x_m^k) \in V$. Then $t_k \geq \min\{1, \gamma\epsilon/3\}$

(iii) If $\lim_k \max\{\|x^k - \bar{x}\|, \|g^k\|, \epsilon_k\} = 0$ with $g^k \in \partial_{\epsilon_k} f(x^k)$ for all k, then $0 \in \partial f(\bar{x})$.

Proof. (i) Let $u \in \text{conv}\nabla f(B(\bar{x}, \epsilon) \cap D)$ be such that $\|u\| < \rho_\epsilon(\bar{x}) + \delta$. From Carathéodory's theorem (10), there exists $(\bar{x}_1, ..., \bar{x}_m) \in D_\epsilon^m(\bar{x})$ and $\lambda \in R_+^m$ with $\lambda^T e = 1$, such that $u = \sum_{i=1}^m \lambda_i \nabla f(\bar{x_i})$. Since f is continuously differentiable on the open set D, there exists $\bar{\epsilon} \in (0, \epsilon)$ such that the set $V := \prod_{i=1}^m \text{int}B(\bar{x}_i, \bar{\epsilon})$ lies in $D_{\epsilon-\bar{\epsilon}}^m$ and $\|\sum_{i=1}^m \lambda_i \nabla f(\bar{y}_i)\| < \rho_\epsilon(\bar{x}) + \delta$ for all $(y_1, ..., y_m) \in V$. Thus, for all $x \in B(\bar{x}, \tau)$, where $\tau := \bar{\epsilon}$, the inclusion $B(\bar{x}, \epsilon - \bar{\epsilon}) \subset B(\bar{x}, \epsilon)$ yields that $D_{\epsilon-\bar{\epsilon}}^m \subset D_\epsilon^m$, from which $V \subset V_\epsilon(\bar{x}, x, \delta)$ holds.

(ii) Define $\bar{G}_k(\{x_i^k\}_1^m) := \text{conv}\{\nabla f(x_i^k)\}_1^m$. Since $(x_1^k, ..., x_m^k) \in V \subset V_\epsilon(\bar{x}, \bar{x}, \delta)$, from (i) we get $d_{G_k}(0) \leq \rho_\epsilon(\bar{x}) + \delta$ and $\bar{G}_k \subset G_\epsilon(\bar{x})$ from 5.3). As $x^k \in B(\bar{x}, \epsilon/3) \cap D$, $\nabla f(x^k) \in G_\epsilon(\bar{x})$ holds. Hence, by the construction of $g^k = \text{proj}(0|G_k)$, it follows that $g \in G_\epsilon(\bar{x})$ (as $G_k \subset G_\epsilon$) and clearly $\|g^k\| = d_{G_k}(0) \leq \rho(\bar{x}) + \delta$ ($(x_1^k, ..., x_m^k) \in V$). From Lemma 4,

$$(v, g^k) > \beta\|g^k\|^2, \text{ for all } v \in G_\epsilon(\bar{x}). \tag{6.5}$$

Suppose the opposite, that $t_k < \min\{1, \gamma\epsilon/3\}$. Then as t_k is the maximum $\gamma^l, l \geq 0$ for which Armijo condition holds, for $t = \gamma^{-1}t_k$ we have

$$-\beta\gamma^{-1}t_k\|g^k\| \leq f(x^k + \gamma^{-1}t_k d^k) - f(x^k). \tag{6.6}$$

Additionaly, the Lebourg's theorem 9 states that there exists $x_o^k \in [x^k, x^k + \gamma^{-1}t_k d^k]$, and $v^k \in \partial f(x_o^k)$ such that

$$f(x^k + \gamma^{-1}t_k d^k) - f(x^k) = (v^k, \gamma^{-1}t_k d^k) = \gamma^{-1}t_k(v^k, d^k) \tag{6.7}$$

From (6.6), and $d^k = \frac{-g^k}{\|g^k\|}$it follows that,

$$-\beta\gamma^{-1}t_k\|g^k\| \leq \gamma^{-1}t_k(v^k, d^k)$$

$$-\beta\|g^k\| \leq (v^k, d^k) \tag{6.8}$$

$$\beta\|g^k\|^2 \geq (v, g^k).$$

Thus from (6.5), $v^k \notin G_\epsilon(\bar{x})$. However, $\gamma^{-1}t_k\|d^k\| \leq \epsilon/3$, and $\|\bar{x}-x^k\| \leq \epsilon/3$ imply that $x_o^k \in B(\bar{x}, 2\epsilon/3)$, hence $v^k \in G_\epsilon(\bar{x})$, which is a contradiction.

(iii)Since we know that $g^k \in \partial_{\epsilon_k} f(x^k)$, latter follows from the closedness (and compactness) of ∂f, as the limits stay in the compact set.

■

As mentioned, there are two types of convergence results. First we will focus on the Kiwiel's algorithm, where the ϵ_k and ν_k are decreasing.

Theorem 12. [KIW] Let $\{x^k\}$ be a sequence generated by the GS algoritm, with $\nu_1 > \nu_{opt} = \epsilon_{opt} = 0$ and $\mu, \theta < 1$. Then, the algorithm does not stop almost always, and either $\lim_k f(x^k) = -\infty$, or $\nu_k \to 0$, $\epsilon_k \to 0$ and every cluster point of $\{x^k\}$ is stationary.

Proof. Finding the nondifferentiable point in Step 1 has the zero probability of occuring, hence let us suppose that the algorithm didn't terminate. Moreover, if $f \to -\infty$, there is nothing to prove, so we assume that $\inf_k f(x^k) > -\infty$. From 5.7 we have

$$\begin{aligned}
\beta t_1\|g^1\| &< f(x^1) - f(x^2) \\
\beta t_2\|g^2\| &< f(x^2) - f(x^3) \\
&\cdot \\
&\cdot \\
&\cdot \\
\beta t_k\|g^k\| &< f(x^k) - f(x^{k+1})
\end{aligned} \tag{6.9}$$

Summing everything gives $\beta\sum_1^k t_i\|g^i\| < f(x^1) - f(x^{k+1})$, thus if $k \to \infty$

$$\sum_1^\infty t_i\|g^i\| < \frac{1}{\beta}(f(x^1) - \lim_k f(x^k)) < \frac{1}{\beta}(f(x^1) - (-\infty)) < \infty \tag{6.10}$$

From (5.11), similarly

$$\sum_{1}^{\infty} \|x^{k+1} - x^k\| \|g^i\| < \infty \tag{6.11}$$

Suppose that there exist $k_1, \bar{\nu}, \bar{\epsilon} > 0$, such that $\nu_k = \bar{\nu}$ and $\epsilon_k = \bar{\epsilon}$, for $k \geq k_1$. That would mean that the algorithm would never terminate. Knowing that $\|g^k\| \geq \bar{\nu}$ =const, from (6.10) and (6.11) we have $t_k \to 0$, and $\|x^{k+1} - x^k\| \to 0$. Consequently the limit $\bar{x}$ exists, such that $x^k \to \bar{x}$. Since the stationarity target, and sampling radius are not reducing, we have two cases.

First, assume that $0 \notin G_{\bar{\epsilon}(\bar{x})}$. If we choose δ, τ and V as in Lemma 5 (ii), we can pick $k_2 \geq k_1$, such that $x^k \in B(\bar{x}, \bar{\epsilon}/3)$ and $t_k \leq \min\{1, \gamma\bar{\epsilon}/3\}$. Thus $(x_1^k, ..., x_m^k) \notin V$ for all $k \geq k_2$, since t_k is not larger than $\min\{1, \gamma\bar{\epsilon}/3\}$. Suffice it to say, this event has the probability 0 of happening, as we choose $(x_1^k, ..., x_m^k)$ independently and uniformly from $D_{\bar{\epsilon}}^m(\bar{x})$, which contains the open set $V \neq \emptyset$ which has nonzero measure.

Now, suppose that $0 \in G_{\bar{\epsilon}}(\bar{x})$, which means $\rho_\epsilon(\bar{x}) = 0$. For $\delta := \bar{\nu}/2$ and τ, V from Lemma 5, we can choose $k_3 \geq k_1$ such that $x^k \in B(\bar{x}, \tau)$. However from $\bar{\nu} \leq \|g^k\| \leq d_{\bar{G}_k}(0)$, we have that $(x_1^k, ..., x_m^k) \notin V$ for all $k \geq k_3$. The probability of this event occuring is also 0. Hence the assumption that a lower bound different from 0 exists was incorrect.

Finally, let us observe the case when $\epsilon_k \to 0$, $\nu_k \to 0$, and $\{x^k\}$ has a cluster point $\bar{x}$. If that cluster point is also a limit, then by Lemma 5, $0 \in \partial f(\bar{x})$, hence the point is stationary. Suppose that $x^k \not\to \bar{x}$. We need to prove that $\lim_k \max\{\|x^k - \bar{x}\|, \|g^k\|\} = 0$ (ϵ_k is approaching zero, thus smaller than the rest of max function). Considering the opposite, there exists $\bar{\nu}, \bar{k} > 0$, and a convergent subsequence of $\{x^k\}$, whose set of indexes we will denote as $K := \{k, k \geq \bar{k} : \|x^k - \bar{x}\| \leq \bar{\nu}\}$. For such $\bar{\nu}$, we know that $\|g^k\| > \bar{\nu}$, for all $k \in K$. Hence, 6.11 gives us: $\sum_{k \in K} \|x^{k+1} - x^k\| < \infty$. Since $x^k \not\to \bar{x}$, there is ϵ such that for each $k \in K$ there exists $\bar{\bar{k}} > k$ satisfying $\|x^{\bar{\bar{k}}} - x^k\| > \epsilon$ and additionally $\|x^i - \bar{x}\| \leq \bar{\nu}$ for $k \leq i < \bar{\bar{k}}$. Thus, we have

$$\begin{aligned} \epsilon < \|x^{\bar{\bar{k}}} - x^k\| &= \|x^{\bar{\bar{k}}} + x^{\bar{\bar{k}}-1} - x^{\bar{\bar{k}}-1} ... + x^{k+1} - x^{k+1} - x^k\| \\ &\leq \sum_{i=k}^{\bar{\bar{k}}-1} \|x^{i+1} - x^i\| < \epsilon \end{aligned} \tag{6.12}$$

which follows for a large k from $\sum_{k \in K} \|x^{k+1} - x^k\| < \infty$, hence a contradiction. ■

This proves the convergence of Kiwiel's GS algorithm, for reducing factors. The following theorem focuses on the original GS, also with non-constant ϵ_k, ν_k.

Theorem 13. [KIW] Let $\{x^k\}$ be a sequence generated by the original (BLO) GS algorithm, with $\nu_1 > \nu_{opt} = \epsilon_{opt}$ and $\mu, \theta < 1$. Suppose that the level set $\{x : f(x) < f(x^1)\}$ is bounded. Then the algorithm does not stop with probability 1, $\nu_k \to 0, \epsilon_k \to 0$, there is a subsequence $K \subset \{1, 2, ...\}$ such that $\lim_{k\in K} g^k = 0$ and every cluster point of $\{x^k\}_{k\in K}$ is stationary for f.

Proof. Again, as in the previous theorem, there are two main cases for ϵ_k and ν_k. The first one, where the factors approach 0, is easier. If we take $K := \{k : \nu_{k+1} \leq \nu_k\}$, it is implied that $\lim_{k\in K} g^k = 0$, and the rest follows from (iii).

We need to reconsider the second case when there exists k_1 such that $\nu_k = \bar{\nu} > 0$ and $\epsilon_k = \bar{\epsilon} > 0$, for $k > k_1$. Since $\|g^k\| \geq \bar{\nu}$, from 6.10, $\lim_k t_k = 0$. Sequence $\{f(x^k)\}$ is decreasing and the set $\{x : f(x) < f(x^1)\}$ is compact, hence there exists a subsequence $J \subset \{1, 2, ...\}$ and a limit $\bar{x}$ such that $\lim_{k\in J} x_k = \bar{x}$. Therefore, $\lim_{k\in J} t_k = 0$ as well, so we argue as in the last theorem, when $0 \in G^m_\epsilon$ and when it is not in the set. We deduce that k_4 and V exist, and similarly that $(x^k_1, ..., x^k_m) \notin V \subset D^m_\epsilon$ for all $k > k_4, k \in J$, which has the probability of occuring 0.

■

For a fixed radius, we have another two theorems. The first one is for Kiwiel's [KIW] algorithm, whereas the latter is for the original [BLO].

Theorem 14. [KIW] Let $\{x^k\}$ be a sequence generated by GS algorithm (Kiwiel's) with $\nu_1 > \nu_{opt} = 0, \epsilon_1 = \epsilon_{opt} = \epsilon > 0$, and $\mu = 1$. With probability 1, the algorithm stops at some iteration k with $0 \in G_\epsilon(x^k)$, or $\lim_k f(x^k) = -\infty$, or a subsequence exists $K \subset \{1, 2, ...\}$ such that $\lim_{k\in K} g^k = 0$ and every cluster point $\bar{x}$ of $\{x^k\}_{k\in K}$ satisfies $0 \in \partial_\epsilon f(\bar{x})$ (ϵ-stationary).

Proof. If the algorithm terminated at iteration k, it happened at Step 3 with probability 1, thus $0 = g^k \in G_\epsilon(x^k)$. We may assume that it doesn't stop and that $\inf_k f(x^k) > -\infty$. From the proof of Theorem 9, almost always $\inf_k \|g^k\| = 0$, hence the infimum stays in the $\partial_\epsilon f(\cdot)$, as the Clarke's ϵ-subdifferential is a closed set. ■

Theorem 15. [KIW] Let $\{x^k\}$ be a sequence generated by the original GS algorithm (BLO), with $\nu_1 = \nu_{opt} = 0$, $\epsilon_1 = \epsilon_{opt} = \epsilon > 0$ and $\mu = 1$. Suppose the set $\{x : f(x) \leq f(x^1)\}$ is bounded. Then almost always, either the algorithm terminates at some iteration k with $0 \in G_\epsilon(x^k)$, or $\lim_k g^k = 0$ and every cluster point $\bar{x}$ of $\{x^k\}$ satisfies $0 \in \partial_\epsilon f(\bar{x})$.

Proof. As before, assume that the termination did not happen, as there would be nothing to prove. Suppose the opposite, that there exists $J \subset \{1, 2, ...\}$, and $\bar{\nu} > 0$, such that $\inf_{k\in J} \|g^k\| > \bar{\nu}$. Now, as $f(x^k)$ is decreasing

and the level set $\{x : f(x) \leq f(x^1)\}$ is compact, there exists a convergent subsequence $\lim_{k\in J} x^k = \bar{x}$ (without the loss of generality). From 6.10, $\lim_{k\in J} t^k = 0$, hence as in Theorem 9, we argue whether or not 0 is in $G_\epsilon(\bar{x})$. Again we deduce that there exists k_5 and an open set V such that $(x_1^k, ..., x_m^k) \notin V \subset D_\epsilon^m(x^k)$, which occurs with probability 0, since the vector product is sampled independently and uniformly from $D_\epsilon^m(x^k)$. This implies that almost always the theorem is indeed correct. ■

Theorem 12 is stronger than 13, however it was achieved by a slight alteration. The original algorithm required compact level sets and resampling wasn't controlled in Step 6 with t_k as in Kiwiel's method. Nonetheless, the result of this renewed algorithm is that all cluster points are stationary, which was an open question in [BLO]. The following section will focus on more slight changes to the algorithm that will have an impact in the practical sense.

Chapter 7

Modifications

In practice, both algorithms are implemented similarly. However confirming the differentiability in each iteration can be problematic. In order to remove that step, algorithm can be modified even more. There will be three different methods introduced in this section, for all of which will the convergence be proved,

7.1 Nonnormalized search directions

The problem with GS algorithm is that sometimes, as the search directions $d_k := -g^k/\|g^k\|$ are normalized, Armijo's line search can grow to infinity. This happens, for example, when $x^{k+1} = x^k + t_k d^k$ for almost all k and $t_k = \|x^{k+1} - x^k\| \to 0$. To resolve this issue, we consider the direction $d_k = -g^k$, as in the gradient descent method in the smooth case.

Formally, the relations 5.5-5.8 from the GS algorithm are replaced by

$$d^k := -g^k \tag{7.1}$$

$$t_k := \max\{t : f(x^k + td^k) < f(x^k) - \beta t\|g^k\|^2, t \in \{1, \gamma, \gamma^2, ...\}\}. \tag{7.2}$$

$$f(x^{k+1}) < f(x^k) - \beta t_k\|g^k\|^2, \tag{7.3}$$

$$\|x^k + t_k d^k - x^{k+1}\| \leq \min\{t_k, \epsilon_k\}\|d^k\| \tag{7.4}$$

By doing so, the equation (5.11) still holds, since $\|x^{k+1} - x^k\| \leq 2t_k\|d^k\| = 2t_k\|g^k\|$. Lemma 5 (ii) is replaced by:

Lemma 6 (Lemma 5 (ii)')**.** [KIW] Let $\epsilon > 0$ and $\bar{x} \in \mathbb{R}^n$. Assuming $0 \notin G_\epsilon(\bar{x})$, choose $\delta > 0$ as in Lemma 4 for $C := G_\epsilon\bar{x}$, and τ, V as in Lemma 5 (i). Suppose that $x^k \in B(\bar{x}, \min\{\tau, \epsilon/3\})$, $\epsilon_k = \epsilon$, and $(x_1^k, ..., x_m^k) \in V$. Then $t_k \geq \min\{1, \gamma\epsilon/3\bar{k}\}$, where $\bar{k}$ is the Lipschitz constant of f on $B(\bar{x}, 2\epsilon)$.

Proof. Following the proof of Lemma 5 (ii), we can get 6.8 by assuming that $t_k \leq \min\{1, \gamma\epsilon/3\bar{k}\}$ instead. We get the same contradiction as before, since $\gamma^{-1}t_k\|d^k\| < \gamma^{-1}t_k\bar{k} < \epsilon/3$ yields $v^k \in G_\epsilon(\bar{x})$, whereas from Lebourg's theorem we get $v^k \notin G_\epsilon(\bar{x})$ for $v_k \in \partial f(\bar{x})$. ■

Further, proofs are modified similarly. For example, equation 6.10, using 7.3 becomes

$$\sum_{k=1}^{\infty} t_k \|g^k\|^2 < \infty, \tag{7.5}$$

and the theorems 12-15 follow. Asymptotically, this new direction may be better, however it can be worse inititally when $\|g^k\|$ is large. Scaling d^k might help, so that the first trial point $x^k + d^k$ is close enough to x^k. If we use the sampling radius ϵ_k as the measure of "closeness", we get the second variant of the algorithm.

7.2 Searching within the trust region

The idea is to restrict the Armijo line search to the trust region $B(x^k, \epsilon_k)$. Relations 5.5-5.8 are replaced by

$$d^k := -\epsilon_k g^k / \|g^k\|, \tag{7.6}$$

$$t_k := \max\{t : f(x^k + td^k) < f(x^k) - \beta t \epsilon_k \|g^k\|, t \in \{1, \gamma, \gamma^2, ...\}\}. \tag{7.7}$$

$$f(x^{k+1}) < f(x^k) - \beta t_k \epsilon_k \|g^k\|, \tag{7.8}$$

$$\|x^k + t_k d^k - x^{k+1}\| \leq \min\{t_k, \epsilon_k\} \|d^k\| \tag{7.9}$$

Again, (5.11) holds as $\|x^{k+1} - x^k\| \leq 2t_k \|d^k\| = 2t_k \epsilon_k$. The Lemma 5 (ii) becomes:

Lemma 7 (Lemma 5 (ii)"). [KIW] Let $\epsilon > 0$ and $\bar{x} \in \mathbb{R}^n$. Assuming $0 \notin G_\epsilon(\bar{x})$, choose $\delta > 0$ as in Lemma 4 for $C := G_\epsilon(\bar{x})$, and τ, V as in Lemma 5 (i). Suppose that $x^k \in B(\bar{x}, \min\{\tau, \epsilon/3\})$, $\epsilon_k = \epsilon$, and $(x_1^k, ..., x_m^k) \in V$. Then $t_k \geq \gamma/3$.

Proof. As in the previous modification, we change the assumption to $t_k < \gamma/3$, and use $d_k := -\epsilon_k g^k / \|g^k\|$ to get $(v^k, g^k) \leq \beta \|g^k\|$, and $\gamma^{-1} t_k \|d\| = \gamma^{-1} t_k \epsilon_k = \gamma^{-1} t_k \epsilon \leq \epsilon/3$ to get $v^k \in G_\epsilon(\bar{x})$ ($\bar{x}$ is from Lebourg's), hence a contradiction.

■

The equation 6.10 now becomes

$$\sum_{k=1}^{\infty} t_k \epsilon_k \|g^k\| < \infty \tag{7.10}$$

which is used to prove analogously the rest of the convergence theorems.

7.3 Limiting the line search

First if we generalize the series of equations 5.5-5.8, we get

$$d^k := -\alpha_k g^k, \alpha_k > 0, \tag{7.11}$$

$$t_k := \max\{t : f(x^k + td^k) < f(x^k) - \beta t \|d^k\| \|g^k\|, t \in \{1, \gamma, \gamma^2, ...\}\}. \tag{7.12}$$

$$f(x^{k+1}) < f(x^k) - \beta t_k \|d^k\| \|g^k\|, \tag{7.13}$$

$$\|x^k + t_k d^k - x^{k+1}\| \leq \min\{t_k, \epsilon_k\} \|d^k\| \tag{7.14}$$

where $\alpha_k = 1/\|g^k\|$ in the original algorithm, $\alpha_k = 1$ in the first, and $\alpha_k = \epsilon_k/\|g^k\|$ in the second modification. The corresponding lower bounds of t_k produced by the Lemma 5(ii),(ii)' and (ii)" have the form of $t_k >\geq \min\{1, \gamma\epsilon/3\|d^k\|\}$. This procedure introduces a lower bound ($\min\{1, \gamma\epsilon/3\|d^k\|\}$) to t_k during the line search, and it accepts the null step size ($t_k = 0$) in the case the bound is reached. After that, the Step 1 resamples the gradient bundle G_k, so that the search direction improves eventually (if x^k is not stationary already). With this implementation, the control of differentiability becomes unnecessary. The improved Armijo line search and updated Lemma 5 (ii) follow:

Step 5'(Limited search)

(i) Choose an initial step size $t \geq \min\{1, \gamma\epsilon_k/3\|d^k\|\}$.

(ii) If $f(x^k + t_k d^k) < f(x^k) - \beta t \|d^k\| \|g^k\|$, return $t_k := t$.

(iii) If $t \leq \min\{1, \gamma\epsilon_k/3\|d^k\|\}$, return $t_k := 0$.

(iv) Set $t := \gamma t$ and go to (ii).

Lemma 8 (Lemma 5(ii)"')**.** [KIW] Let $\epsilon > 0$ and $\bar{x} \in \mathbb{R}^n$. Assuming $0 \notin G_\epsilon(\bar{x})$, choose $\delta > 0$ as in Lemma 4 for $C := G_\epsilon \bar{x}$, and τ, V as in Lemma 5 (i). Suppose that $x^k \in B(\bar{x}, \min\{\tau, \epsilon/3\})$, $\epsilon_k = \epsilon$, $(x_1^k, ..., x_m^k) \in V$ and $d^k := -\alpha g^k$ with $\alpha_k > 0$. Then the Step 5' finds a step size $t_k \geq \min\{1, \gamma\epsilon/3\|d^k\|\}$.

Proof. Similarly as before, we use 6.5 to get to the contradiction. Suppose $t \in (0, \epsilon/3\|d^k\|)$. From Lebourg's mean value theorem, there is $v \in \partial f(x), x \in [x^k + t_k d^k, x^k]$ such that $f(x^k + t_k d^k) - f(x^k) = t\langle v, d^k \rangle$. It follows that $(v, d^k) < -\beta\|d^k\|\|g^k\|$, hence $v \notin G_\epsilon(x)$. From $t\|d^k\| < \epsilon/3$ and $\|x^k - \bar{x}\| < \epsilon/3$ we know that $x \in B(\bar{x}, 2\epsilon/3)$ and thus $v \in G_\epsilon(x)$, a contradiction. ■

From this lemma, the analogue convergence theorems follow. We can control the number of f-evaluations made by the Step 5' via the choice of the step size t. If we choose $t := \min\{1, \epsilon_k/3\|d^k\|\}$, only one evaluation occurs. However, if the initial step size t looks small, we can divide t by γ ($t =: t/\gamma$), until $f(x^k + td^k) \geq f(x^k) - \beta t\|d^k\|\|g^k\|$, then $t_k := \gamma t$. Here, the trade-off is lowering the number of f-evaluations for the cost of more gradient sampling at Step 1. More modifications can be done to the lower bound of t_k, after which the convergence is preserved.

The replacement of standard Armijo line-search makes confirmation of differentiability in x^k and inclusion of $\nabla f(x^k) \in G_k$ obsolete. A simplified version of the algorithm can be made in the following way: At Step 0, we select an arbitrary $x^1 \in \mathbb{R}^n$. At Step 1, we set $G_k := \text{conv}\{\nabla f(x_i^k)\}_{i=1}^m$. Additionaly, the Step 5 is replaced by Step 5'. Finally, at Step 6, set $x^{k+1} := x^k + t_k d^k$. From Step 5', the equation 7.13 holds if $t_k > 0$, and the inequality 5.11 is valid. The theorems for convergence hence follow. Unfortunately, it is not proven that the event $x^k \notin D$ has probability 0, however in practice it rarely happens.

In the last section, an implementation of the algorithm will be shown in the programming language Python, and its application in the SVM method will be demonstrated. The results will be presented for four different datasets, which will all be of different sizes.

Chapter 8

Numerical results

8.1 Binary classification

First, let us go back to the section 3.1, equations (3.7) and (3.11), where we defined the binary hinge loss with the regularization factor and its subdifferential. Namely, the problem we are solving in the SVM has the form:

$$\min_x f(x) = \min_x (\frac{\lambda}{2}\|x\|^2 + \frac{1}{K}\sum_{i=1}^{K} l(t_i, x, \omega_i)) \tag{8.1}$$

The columns of matrix ω are different attributes, whereas the samples are represented by rows. Hence $\omega_i \in \mathbb{R}^n, i = 1, ..., K$. The variable $x \in \mathbb{R}^n$ is the vector of weight coefficient we are searching for. We start by creating the function and its subdifferential with the help of Python's 'numpy' library A.1. The function *hinge_izvod* returns the subderivative of f at the point x, and tells whether the function is differentiable. However, when the function is non differentiable at x, that subderivative is not used in the algorithm, as the algorithm resamples the vector in that case. In figure A.2 we can see the implementation of the Kiwiel's GS algorithm.

Similarly, the modifications of the algorithm have also been implemented. There are four algorithms in total, the (regular) gradient sampling algorithm [5.1], the nonnormalized gradient sampling [7.1], the limiting line search [7.3] and the trust region gradient sampling algorithm [7.2]. The codes for the modifications are in the appendix (A.3,A.4 and A.5).

During the implementation, a problem at Step 2 arises. Notice that finding $g_k = proj(0|G_k)$ translates to finding a vector with a minimum norm from the convex hull of the sampled gradient vectors. Practically speaking, that means it is necessary to solve a quadratic constrained optimization problem: $\min_\lambda \|G\lambda\|^2, \lambda e = 1, \lambda_i > 0$, where $\lambda \in \mathbb{R}^m$ is the convex combination coefficient vector and $G \in \mathbb{R}^{n,m}$ is the matrix whose columns are sampled gradients. To solve this problem, a Python library module 'scipy.optimize' is used, together with the SLSQP (Sequential Least Squares Quadratic Programming) method. This is a Lagrangian optimization method described in [DK], by Dieter Kraft.

The preprocessing of the datasets has been accomplished using the Python library 'sklearn'. For every dataset, the label column has been transformed to only contain 1 and -1 values, and the features have been encoded and standardized accordingly. Additionaly, the missing data was

either deleted or approximated, depending on the set. Table 1 shows the dimensions of the four tested datasets.

	K	n	K-train	K-test
Mushroom	8124	112	6499	1625
Adult	32561	108	26048	6513
Heart	303	13	242	61
Bank	45211	48	36168	9043

Table 8.1: Tested datasets' properties

Each dataset has been split into training and test set. The gradient sampling algorithm is using the training set to find the coefficients, and it validates the predictions after each iteration with the test set.

8.1.1 Mushroom dataset

Here, we are trying to predict whether the mushroom is edible (1) or not (-1). We are given 8124 samples, with 22 non numerical attributes. One of the columns had too much missing values, hence it was removed from the dataset. Once the features had been encoded, the number of columns went up to 112, all of which only consisted of 0 and 1 values. After splitting the dataset into training and test set, the data has been standardized. The algorithm was trained with parameters:$\nu_{opt} = 0.005$, $\epsilon_{opt} = 0.1$, $\beta = 0.1$, $\gamma = 0.85$, $\mu = 0.35$, $\theta = 0.5$, $\epsilon_1 = 0.5$, $\nu_1 = 0.12$, $\alpha_k = 1$ (for limiting line search) and $m = 120$.

Namely, the limiting line search from 7.3 has two ways of being implemented. The first one chooses the step size $t = \min\{1, \gamma\epsilon/3\|d^k\|\}$. If the Armijo condition isn't satisfied, the point needs to be resampled ($t_k = 0$), since $\min\{1, \gamma\epsilon/3\|d^k\|\}$ is the lower bound of t_k in this modification. However, if the condition holds, we update the step size $t_k = t$ and proceed to the next step.

The second way starts off the same, with the lower bound being $t = \min\{1, \gamma\epsilon/3\|d^k\|\}$ in the case Armijo condition holds, else $t_k = 0$ and we resample the iterate. We proceed to update $t = t/\gamma$ as long as Armijo condition is satisfied. At the end of the loop, $t_k = \gamma t$ satisfies the requirement. The second method proved to be much more efficient for the mushroom set. For the rest of the datasets it also showed to be the best. As we can see in Figure 8.1, the limiting line search is the slowest, and in Figure A.2 it is much faster.

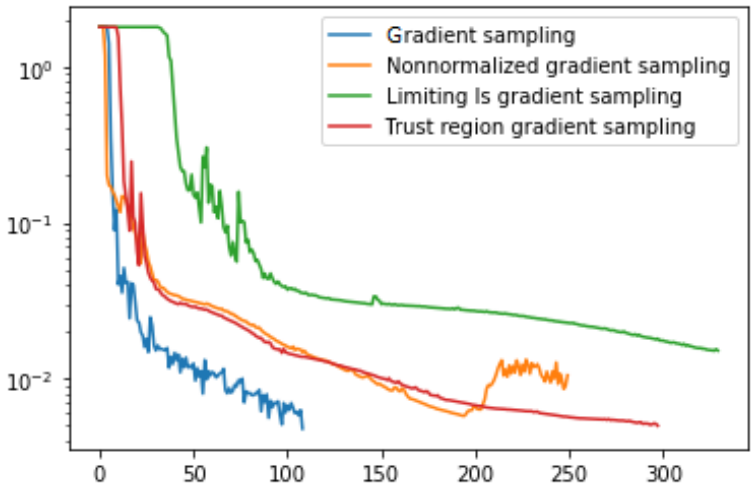

Figure 8.1: The norm of the sampled gradient, first method

Here, the limiting line search modification is slower, as the resampling is more frequent the closer we get to the stationary point. The nonnormalized gradient sampling appears to be unstable at the end, as the gradients are not controlled by normalization. The trust region gradient sampling appears to be the most stable, however it is slower than the regular gradient sampling, as it is using the smaller search steps in general. In the following figure, we can see a drastical difference in the limiting gradient sampling modification's performance.

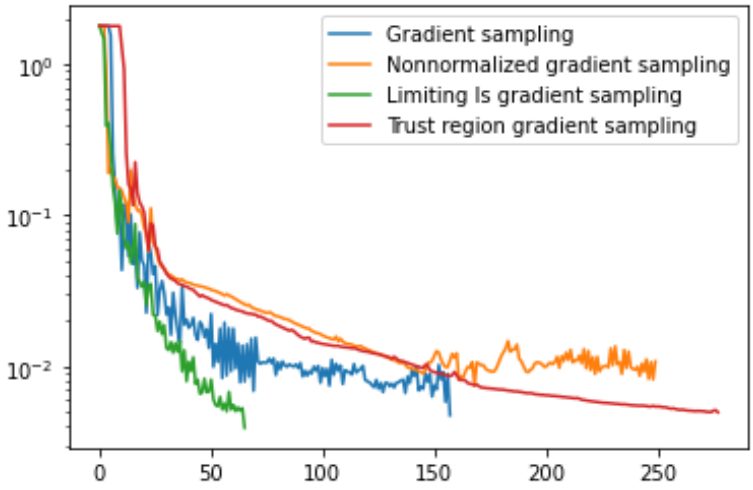

Figure 8.2: The norm of the sampled gradient, second method

The Figure 8.3 shows the loss values of the training sets and test sets for each algorithm: the gradient sampling, limiting gradient sampling, trust region gradient sampling and nonnormalized gradient sampling respectively.

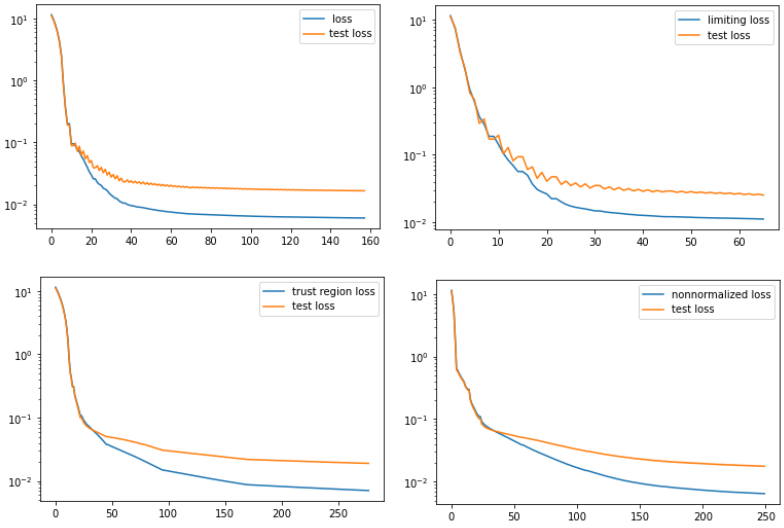

Figure 8.3: Loss values for each algorithm

The blue line in each panel of Figure 8.3 is the loss of the training set, whereas the orange line represents the loss of the test set. We can notice that the test loss is a bit larger, although not by much. In Figure 8.4 the training loss is compared between the methods.

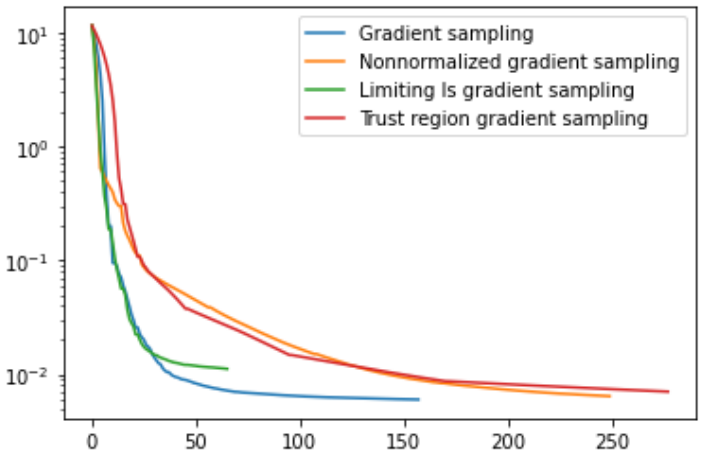

Figure 8.4: Training loss of the mushroom dataset

The loss comparison follows the gradient norm comparison of the algorithms, and both imply that the limiting gradient sampling converges the fastest for this dataset.

With the calculated coefficients, we can predict the target column of the test set. If we use the 'metrics' module of the sklearn library, we easily calculate the efficiency of the SVM algorithm. For the mushroom dataset, the SVM predicted the test target with 99% accuracy.

8.1.2 Adult dataset

In this dataset, the goal is to find out whether the person makes more or less than 50000$ a year. The features that are given are personal information of the observed people, such as: working class, age, education, marital status etc. After deleting the rows with the missing values we are left with 32561 samples.

As in the previous dataset, the features had to be encoded, which made the problem more difficult, due to the fact that the number of features increased from 15 to 108. The Figure 8.11 shows the efficiency of the algorithms.

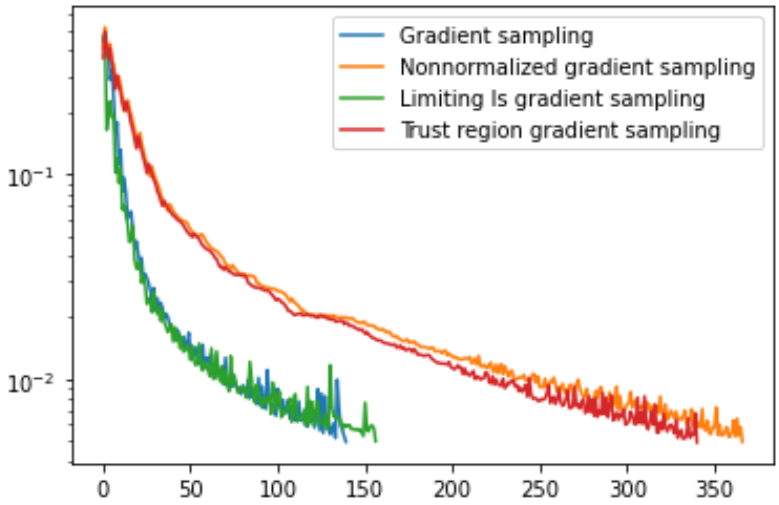

Figure 8.5: The norm of the chosen sampled gradient at each iteration

We have got similar results in terms of algorithm's performances, with gradient sampling and limiting gradient sampling being much faster than the other two algorithms. However, the nonnormalized and trust region gradient sampling appear to be more stable as they converge. This might be due to

the sparse data, and the dimension of the set. The Figure 8.6 follows the previous plot with the same observations. With the accuracy of 82.8% the SVM predicts the labels of the test set.

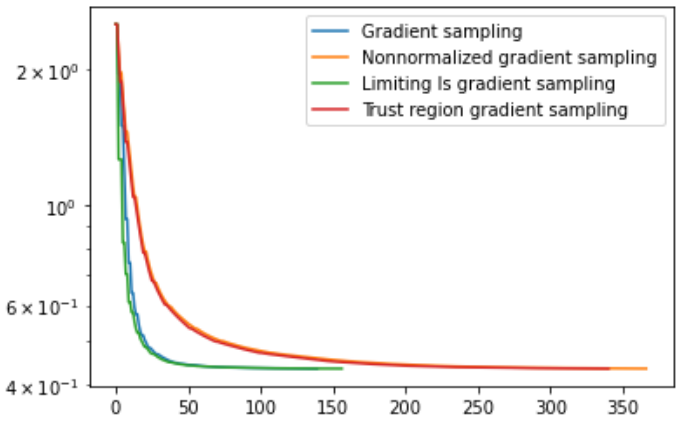

Figure 8.6: Training loss of the adult dataset

8.1.3 Heart dataset

Here, we need to predict the risk of the person getting a heart attack, given the health examination results. The features are already in the numerical form, hence there is no need for encoding. Figures 8.7 and 8.8 display the flow of the algorithms.

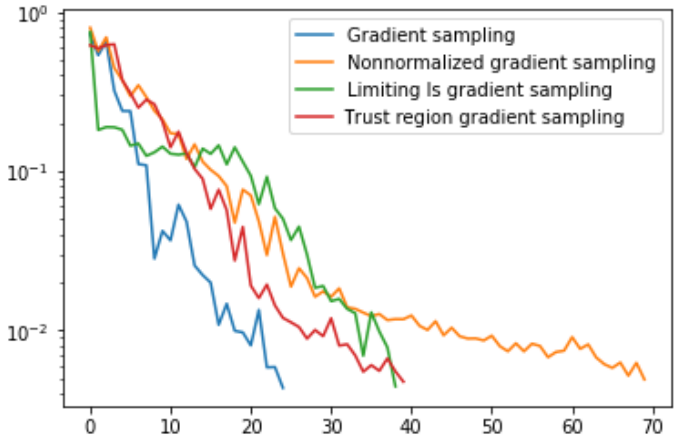

Figure 8.7: The norm of the chosen sampled gradient at each iteration

This is the smallest dataset, consisting of only 303 samples and 13 attributes. Notice that unlike in the previous sets, the limiting gradient sampling doesn't seem to be the most efficient algorithms. Also, all of the algorithms converge extremely fast and unstable, the slowest being the nonnormalized gradient sampling. The SVM's accuracy using the calculated coefficients is 72.1%.

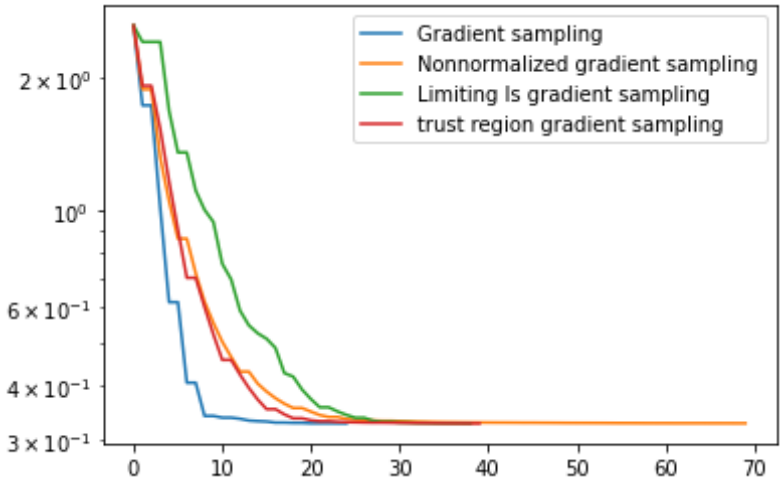

Figure 8.8: Training loss of the heart dataset

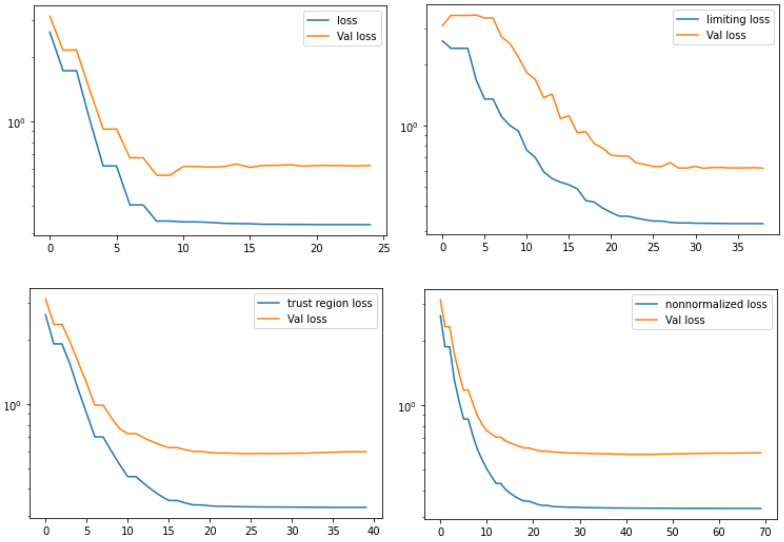

Figure 8.9: Loss values for each algorithm of the heart set

8.1.4 Bank dataset

With the personal information of the users, and previous interactions with the bank, we predict whether the client will support the bank's campaign. This is the largest dataset tested (sample wise), with 45211 samples and 48 features after the encoding. The SVM predicted the labels with 89.8% accuracy. Following graphs illustrate the convergence.

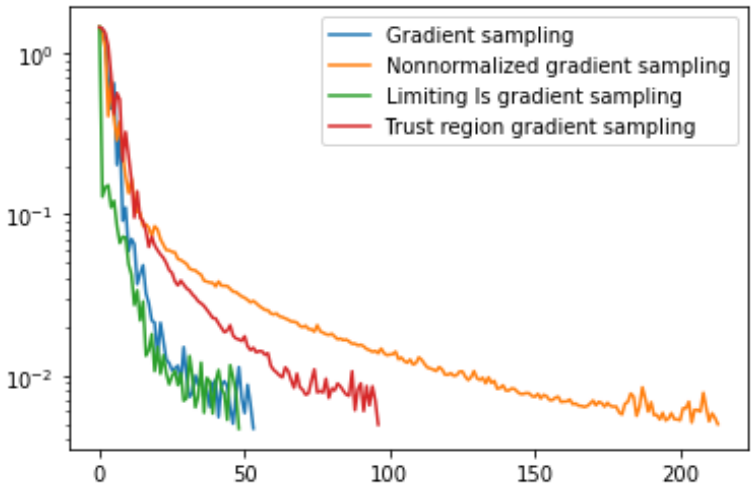

Figure 8.10: The norm of the chosen sampled gradient at each iteration

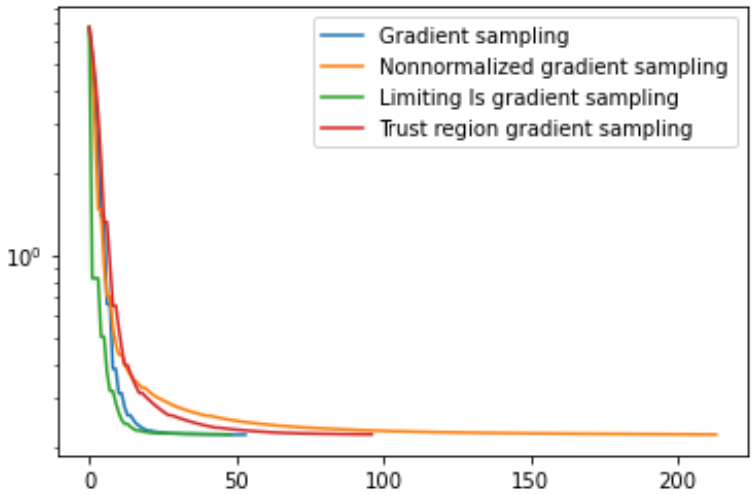

Figure 8.11: Training loss of the bank dataset

We can notice that the gradient sampling and the limiting gradient sampling displayed an outstanding performance in most of the datasets. The nonnormalized and the trust region modifications however seem to be the slowest for the larger and more robust datasets.

8.2 Industrial application - anomaly detection

So far, we have seen the notion of the SVM and how to implement it, we have proved the functionality behind the gradient sampling algorithms, and we managed to apply the binary classification method on four datasets. Now, as the climax and the resolution of this story, this section will consider an industrial problem. Here, the section 3.2 will be materialized, by applying the anomaly extraction one class SVM (OCSVM).

As mentioned, finding anomalies is of great importance in industry as it can predict the malfunctioning of infrastructures, and thus prevent unnecessary losses. There are many algorithms that can achieve this, however different problems might require different approaches. We will cover the Smart Logistics system that uses Cellular IoT (CIoT) for operating, which is discussed in [LS], and we will try to predict the anomalies using the OCSVM.

Namely, the breaktrough of Internet of Things (IoT) and the massive integration of IoT devices in systems such as Smart Factories, Smart Grids, Smart Logistics and others, caused an influx of challenges. The requirement to secure and manage the large scale data, also the control of the intelligent manufacturing are all important factors that need to be adressed. This is where machine learning algorithms prove to be helpful. Some of the security issues and threats in industrial IoT networks can be solved by anomaly and intrusion detection, malware analysis and DDoS attacks detection, all of which are considered to be a part of the machine learning toolkit.

The goal of this section is to implement the optimization algorithms in order to find the anomalies within the dataset from M. Lukić, M.Savić [LS]. The mentioned research is part of the H2020 C4IIoT project-Cyber security 4.0: protecting the Industrial Internet of Things. The project is funded by the European Union's Horizon 2020 research and innovation programme. Its purpose is to build and demonstrate a novel and unified IIoT cybersecurity framework for malicious and anomalous behavior anticipation, detection, mitigation, and end-user informing.

For this purpose the dataset was generated using NB-IoT (Narrowband Internet of Things) edge nodes. NB-IoT is a narrowband radio technology for IoT devices and applications requiring wireless transmission over a more extended range at a relatively low cost. It is a type of Cellular IoT (CIoT) that can be integrated in the existing 3GPP 4G/5G architecture,

and coexist with the 4G and 5G technology. They created a setup where an edge node has been attached to a box-shaped container inside a transport vehicle moving through the city of Novi Sad. The devices were connected to the NB-IoT network, and the connectivity was uninterrupted along the paths. They collected the positioning data from the Global Navigation System Satellite-GNSS module (timestamp, latitude, longitude, altitude, speed and number of satellites in range), as well as the outputs of the Inertial Measurement Unit-IMU (acceleration and magnetic field along the 3 spatial axes). The time resolution (sampling period) of the GNSS samples was approximately 10 s, whereas the sampling period of the IMU was approximately 15 ms. Additionaly, they calculated the root mean square as well as the arithmetic mean for the acceleration and magnetic field samples collected within a GNSS sampling interval.

The dataset is ordered by timestamp, and it had been split into training and test set by taking out random chunks of the data from the dataset. The training set has 11743 samples, whereas the test set has 1380 samples. There are two cases to be considered. The first one includes latitude and longtitude in the dataset, whereas the second doesn't consider the positioning. There are 13 attributes in total (11 in the second case), all of which are numerical and have been scaled. For the test data ground truth anomalies are given, which will be used to check whether the detected anomalies are real. Beforehand, in the section 3.2 we introduced the method for finding anomalies by minimizing the function:

$$\min_{x,r} f(x,r) = \min_{x,r} \Big(\frac{\lambda \|x\|^2}{2} - \lambda r + \frac{1}{K} \sum_{i=1}^{K} \max\{0, r - \langle x, \omega_i \rangle\} \Big), \tag{8.2}$$

Modifying the hinge loss from the binary classification gives us the required function that ought to be minimized. The accuracy of the OCSVM is given by computing the following basic measures:

1. TP (true positives) - the number of correctly predicted anomalies

2. FP (false positives) - the number of times the model predicted an anomaly when it wasn't one

3. FN (false negatives) - the number of times the model didn't predict the anomaly when it should've

4. TN (true negatives) - the number of correctly predicted non-anomalous samples.

From these, we derive the standard measures:

$$\text{Precision (P)} = \frac{TP}{\text{TP} + \text{FP}} \tag{8.3}$$

$$\text{Recall (R)} = \frac{TP}{\text{TP} + \text{FN}} \tag{8.4}$$

$$\text{Accuracy (A)} = \frac{TP + TN}{\text{FP} + \text{FN} + \text{TN} + \text{TP}} \tag{8.5}$$

$$\text{F1 score (F1)} = \frac{2 \cdot P \cdot R}{P + R} \tag{8.6}$$

Precision measures the percentage of correctly identified anomalies. Small precision implies that the model makes a lot of mistakes when finding outliers, and 'falsely alarms' the system. Recall reflects the percentage of true anomalies spotted. If it is low, it means the model will not detect real outliers, as it doesn't alarm anomalous events efficiently.

Less important in anomaly detection, the accuracy measures how efficiently does the model find both the anomalies and the non-anomalies. High accuracy doesn't necessarily imply that the model is valid, as it is more important to identify the anomalous data than to predict the non anomalous samples. Last but not least, the F1 score measures the harmonic mean and it weights the precision and recall equally. It favours the models that do not show extreme behaviour, the ones having extremely low precision and high recall and vice versa. These measures can also be presented as the confusion matrix, which is defined as follows:

$$\begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix}$$

For both cases all metrics have been calculated. Parameters that have been used for training are: $\nu_{opt} = 0.0005, \epsilon_{opt} = 0.4, \beta = 0, \gamma = 0.8, \mu = 0.6, \theta = 0.8, \epsilon_1 = 0.8, \nu_1 = 0.6, \lambda = 0.05$ and $m = 20$. After the training of the four algorithms presented in chapters 5 and 7, the anomalies were predicted on the test set, for both cases (with and without the position). The algorithms converged and the results are presented in Table 8.2. As we can see, the model is more precise when it uses the location, however the second model has higher recall and f1 score. The difference is not as significant as one would expect.

	With location	Without location
Accuracy	0.729	0.718
Precision	0.738	0.681
Recall	0.414	0.448
F1 score	0.531	0.541
Consfusion matrix	$\begin{bmatrix} 793 & 75 \\ 299 & 212 \end{bmatrix}$	$\begin{bmatrix} 762 & 107 \\ 282 & 229 \end{bmatrix}$

Table 8.2: Results for both datasets

The Figure 8.12 and 8.13 show the training loss of the algorithms for both datasets, whereas in Figure 8.14 and Figure 8.13 we can see the unstable norm of the sampled gradients at each iteration of the algorithms.

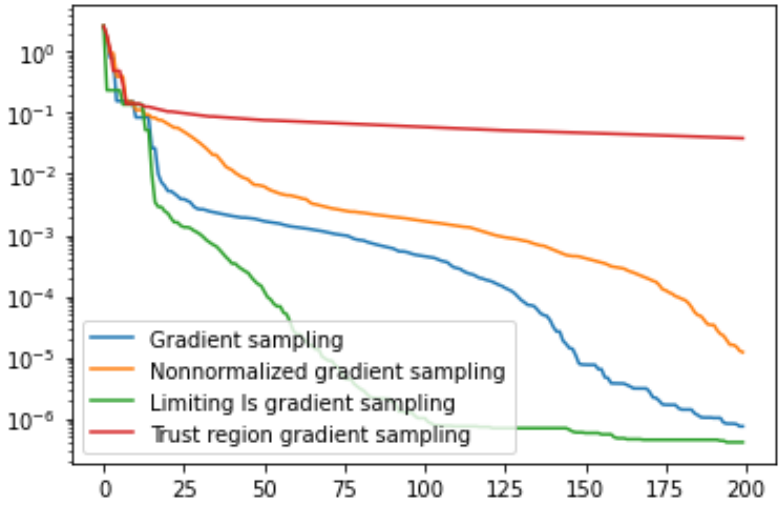

Figure 8.12: Training loss of the C4IIoT dataset (no location)

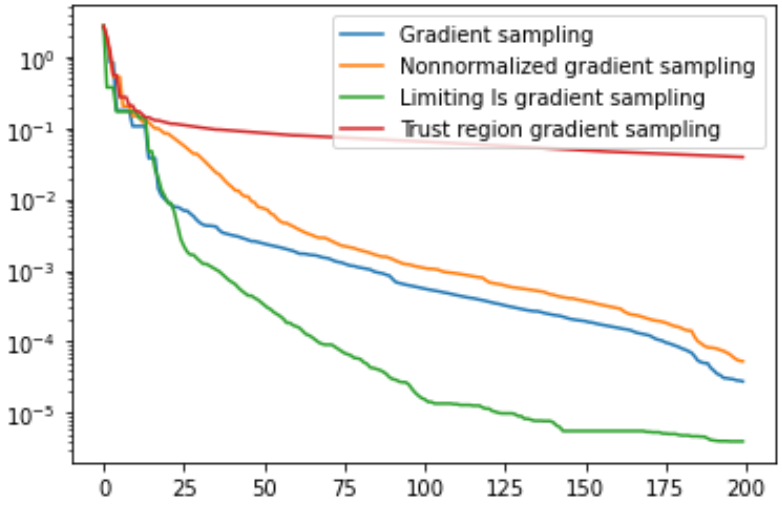

Figure 8.13: Training loss of the C4IIoT dataset (location)

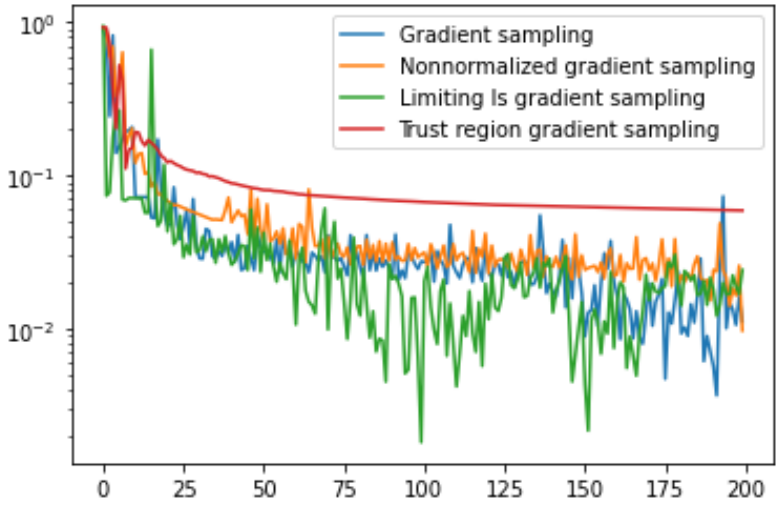

Figure 8.14: The norm of the gradient at each iteration (no location)

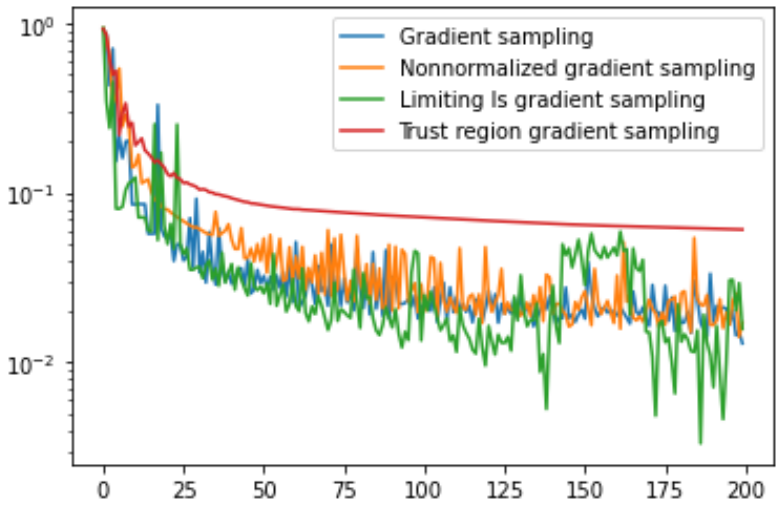

Figure 8.15: The norm of the gradient at each iteration (location)

The trust region gradient sampling method had the most 'trouble' converging, whereas the limiting gradient sampling appears to be the most efficient.

Chapter 9

Conclusion

In this paper, we have looked back on the principles of Machine learning, the SVM binary classification method and the one class SVM for anomaly extraction. After introducing the binary hinge loss, we came across constrained minimization problems which needed to be solved in order to implement the SVM models accordingly.

The recommended method for solving them was the gradient sampling (for nonsmooth functions) algorithm. In order to prove its functionality, nonsmooth analysis theory is used. We stated Lebourg's and Carathéodory's theorems, and we introduced concepts necessary for defining the gradient sampling algorithm. The proofs for convergence of the Kiwiel's and Burke, Levis, Overton's original gradient sampling method have been provided. Additionaly, the modifications have been made for the purpose of easier practical implementation and higher efficiency, and its convergence has also been proven.

Afterwards, the algorithms were used in the practical sense for predicting the labels of four different datasets. All of the versions of the algorithms successfully converged, with some being faster at the time than the others. We managed to train the SVM and present the results, however some aspects of the model like overfitting were not discussed.

The last section focuses on the industrial application of the algorithm. The dataset which had been generated for a C4IIoT project, was modified to fit our gradient sampling algorithm. Namely, this dataset had been created using NB-IoT edge nodes which were attached to the transport vehicle moving through the city of Novi Sad. With the time resolutions of 10s and 15ms, the positioning data had been collected from the GNSS and IMU modules respectively. The outlier detection method was trained on the training set and it was used to identify the test set anomalies. In the end, the confussion matrix, precision, recall and accuracy were calculated, and the plots which show the convergence of the four algorithms were presented.

With this, the wonderful cycle of applying the mathematical aparatus concludes.

Appendix A

The Python code

```
def hinge_loss(x,omega,z,l,N):
    predikcija=omega.dot(x)
    maximumi=np.maximum(0,1-z*predikcija)
    return l/2*LA.norm(x,2)**2+1/N*np.sum(maximumi)

def hinge_izvod(x,omega,z,l,N):
    y=omega.dot(x)*z
    maximumi=1-y
    izvodi_svi = (omega.T*z).T
    izvodi_veci=izvodi_svi[maximumi>0]
    izvodi_nula=izvodi_svi[maximumi==0]
    beta=np.random.uniform(0,1,np.size(izvodi_nula))
    if len(izvodi_nula)!=0:
        return l*x-sum(izvodi_veci)/N-np.dot(izvodi_nula,beta)/N, False
    elif len(izvodi_nula)==0:
        return l*x-sum(izvodi_veci)/N, True

def f(A,x):
    y = np.dot(A, x)
    return 0.5*np.dot(y, y)


def lop(x,e,m):
  l=np.array([x+np.random.uniform(-e,e,np.size(x))])
  for i in range(m-1):
        l=np.concatenate((l, [x+np.random.uniform(-e,e,np.size(x))]))

  return l
```

Figure A.1: Hinge loss and its subdifferential, utility functions

```
def gs_algorithm(x0,vopt,eopt,beta,gamma,mi,theta,e1,v1,m,\
                 omega,omega_test,z,z_test,l,N,N_test,k,his):
    print('Korak:',k)
    lopta=np.array([x0])
    gradienti= np.array([hinge_izvod(lopta[0],omega,z,l,N)[0]])
    for i in range(m-1):
        lopta= np.concatenate((lopta, [x0+np.random.uniform(-e1,e1,np.size(x0))]))
        gradienti= np.concatenate((gradienti,[hinge_izvod(lopta[i+1],omega,z,l,N)[0]]))

    cons = ({'type': 'eq', 'fun': lambda x: x.sum() - 1})
    bnds = ((0, None),)*m
    res = optimize.minimize(f, np.zeros(m),args=gradienti, method='SLSQP',\
                            bounds=bnds, constraints=cons, options={'disp': False})
    gk = np.dot(gradienti.T,res['x'])
    his.append([LA.norm(gk,2),hinge_loss(x0,omega,z,l,N),hinge_loss(x0,omega_test,z_test,l,N_test)])
    print('Norma gk:',LA.norm(gk,2))
    if((LA.norm(gk,2)<vopt)and(e1<eopt))or(k==200):
        return x0,his
    else:
          if (LA.norm(gk,2)<v1):
              v1*=theta
              e1*=mi
              k+=1
              print('e1:',e1)
              print('v1:',v1)
              return gs_algorithm(x0, vopt, eopt, beta, gamma, mi, theta, e1, v1, m,\
                                  omega,omega_test,z,z_test,l,N,N_test, k,his)

          else:
              dk=-gk/LA.norm(gk,2)
              t=1
              while(hinge_loss(x0+t*dk,omega,z,l,N)> hinge_loss(x0,omega,z,l,N)-beta*t*LA.norm(gk,2)):
                  t*=gamma

              k+=1
              if hinge_izvod(x0+t*dk,omega,z,l,N)[1]==True:
                  x0+=t*dk
                  print('prekinut backtracking tk=',t)
                  print('novo xk, diferencijabilno')
                  print('e1:',e1)
                  print('v1:',v1)

              else:
                  for x in lop(x0+t*dk,e1,m):
                       print('pretraga')
                       if (hinge_loss(x,omega,z,l,N)<hinge_loss(x0,omega,z,l,N)-beta*t*LA.norm(gk,2))\
                           and (LA.norm(x0+t*dk-x,2)<=np.minimum(t,e1)):
                           print('nadjeno')
                           x0=x
                           break

              return gs_algorithm(x0, vopt, eopt, beta, gamma, mi, theta, e1, v1, m,\
                                  omega,omega_test,z,z_test,l,N,N_test, k,his)
```

Figure A.2: The gradient sampling algorithm

```
def gs_algorithm_nonnorm(x0,vopt,eopt,beta,gamma,mi,theta,\
                         e1,v1,m,omega,omega_test,z,z_test,l,N,N_test,k,his):
    print('Korak:',k)
    lopta=np.array([x0])
    gradienti= np.array([hinge_izvod(lopta[0],omega,z,l,N)[0]])

    for i in range(m-1):
        lopta= np.concatenate((lopta, [x0+np.random.uniform(-e1,e1,np.size(x0))]))
        gradienti= np.concatenate((gradienti,[hinge_izvod(lopta[i+1],omega,z,l,N)[0]]))

    cons = ({'type': 'eq', 'fun': lambda x: x.sum() - 1})
    bnds = ((0, None),)*m
    res = optimize.minimize(f, np.zeros(m),args=gradienti, method='SLSQP',\
                            bounds=bnds, constraints=cons, options={'disp': False})
    gk = np.dot(gradienti.T,res['x'])

    his.append([LA.norm(gk,2),hinge_loss(x0,omega,z,l,N),\
                hinge_loss(x0,omega_test,z_test,l,N_test)])
    print('Norma gk:',LA.norm([gk],2))


    if((LA.norm(gk,2)<vopt)and(e1<eopt))or(k==250):
        return x0,his
    else:

          if (LA.norm(gk,2)<v1):
              v1*=theta
              e1*=mi
              k+=1
              print('e1:',e1)
              print('v1:',v1)
             # print('Tacka je:',x0)
              return gs_algorithm_nonnorm(x0, vopt, eopt, beta, gamma, mi,\
                                          theta, e1, v1, m, omega,omega_test,z,z_test,l,N,N_test, k,his)

          else:
              dk=-gk
              t=1
              while(hinge_loss(x0+t*dk,omega,z,l,N)> \
                    hinge_loss(x0,omega,z,l,N)-beta*t*(LA.norm(gk,2))**2):
                  t*=gamma
              print('prekinut backtracking, tk=',t)
              k+=1
              if hinge_izvod(x0+t*dk,omega,z,l,N)[1]==True:
                  x0+=t*dk
                  print('novo xk, diferencijabilno')
                  print('e1:',e1)
                  print('v1:',v1)
              else:
                  for x in lop(x0+t*dk,e1,m):
                       print('pretraga')
                       if (hinge_loss(x,omega,z,l,N)<hinge_loss(x0,omega,z,l,N)-\
                           beta*t*(LA.norm(gk,2))**2) and (LA.norm(x0+t*dk-x,2)<=np.minimum(t,e1)*LA.norm(dk,2)):
                           print('nadjeno')
                           x0=x
                           break

              return gs_algorithm_nonnorm(x0, vopt, eopt, beta, gamma, \
                                          mi, theta, e1, v1, m, omega,omega_test,z,z_test,l,N,N_test, k,his)
```

Figure A.3: The Nonnormalized gradient sampling algorithm

```
def gs_algorithm_limited1(x0,vopt,eopt,beta,gamma,mi,theta,e1,v1,m,omega,omega_test,z,z_test,l,N,N_test,k,a1,his):
    print('--------------------> Korak:',k)
    lopta=np.array([x0])
    gradienti= np.array([hinge_izvod(lopta[0],omega,z,l,N)[0]])

    for i in range(m-1):
        lopta= np.concatenate((lopta, [x0+np.random.uniform(-e1,e1,np.size(x0))]))
        gradienti= np.concatenate((gradienti,[hinge_izvod(lopta[i+1],omega,z,l,N)[0]]))

    cons = ({'type': 'eq', 'fun': lambda x: x.sum() - 1})
    bnds = ((0, None),)*m
    res = optimize.minimize(f, np.zeros(m),args=gradienti, method='SLSQP',\
                            bounds=bnds, constraints=cons, options={'disp': False})
    gk = np.dot(gradienti.T,res['x'])

    his.append([LA.norm(gk,2),hinge_loss(x0,omega,z,l,N),hinge_loss(x0,omega_test,z_test,l,N_test)])
    print('Norma gk:',LA.norm([gk],2))


    if((LA.norm(gk,2)<vopt)and(e1<eopt))or k==330:
    else:

          if (LA.norm(gk,2)<v1):
              v1*=theta
              e1*=mi
              k+=1
              print('e1:',e1)

              return gs_algorithm_limited1(x0, vopt, eopt, beta, gamma, mi,\
                                           theta, e1 v1, m,omega,omega_test,z,z_test,l,N,N_test, k,a1,his)
          else:
              t=np.minimum(1,gamma*e1/(3*LA.norm(dk,2)))
              if hinge_loss(x0+t*dk,omega,z,l,N)<hinge_loss(x0,omega,z,l,N)-beta*t*LA.norm(dk,2)*LA.norm(gk,2):
                  while(hinge_loss(x0+t*dk,omega,z,l,N)<hinge_loss(x0,omega,z,l,N)-beta*t*LA.norm(dk,2)*LA.norm(gk,2)):
                       t=t/gamma

                  t=gamma*t
              else:
                  t=0
             # while(hinge_loss(x0+t*dk,omega,z,l,N)> hinge_loss(x0,omega,z,l,N)\\
                 #-beta*t*LA.norm(dk,2)*LA.norm(gk,2)):
              #    if t<np.minimum(1/gamma,e1/(3*LA.norm(dk,2))):
               #     t=0
                # else:
                 # t*=gamma

              print('prekinut backtracking tk=',t)
              k+=1
              if hinge_izvod(x0+t*dk,omega,z,l,N)[1]==True:
                  x0+=t*dk
                  print('novo xk,diferencijabilno')
                  print('e1:',e1)


              else:
                  for x in lop(x0+t*dk,e1,m):
                       print('pretraga')
                       if (hinge_loss(x,omega,z,l,N)<hinge_loss(x0,omega,z,l,N)-beta*t*(LA.norm(gk,2))*(LA.norm(dk,2)))\
                           and (LA.norm(x0+t*dk-x,2)<=np.minimum(t,e1)*LA.norm(dk,2)):
                           print('nadjeno')
                           x0=x
                           break

              return gs_algorithm_limited1(x0, vopt, eopt, beta, gamma, mi, theta, e1, \
                                           v1, m,omega,omega_test,z,z_test,l,N,N_test, k,a1,his)
```

Figure A.4: The Limiting line search gradient sampling algorithm

```
def gs_algorithm_trust(x0,vopt,eopt,beta,gamma,mi,theta,e1,v1,m,omega,omega_test,z,z_test,l,N,N_test,k,his):
    print('Korak:',k)
    lopta=np.array([x0])
    gradienti= np.array([hinge_izvod(lopta[0],omega,z,l,N)[0]])

    for i in range(m-1):
        lopta= np.concatenate((lopta, [x0+np.random.uniform(-e1,e1,np.size(x0))]))
        gradienti= np.concatenate((gradienti,[hinge_izvod(lopta[i+1],omega,z,l,N)[0]]))

    cons = ({'type': 'eq', 'fun': lambda x: x.sum() - 1})
    bnds = ((0, None),)*m
    res = optimize.minimize(f, np.zeros(m),args=gradienti, method='SLSQP',\
                            ounds=bnds, constraints=cons, options={'disp': False})
    gk = np.dot(gradienti.T,res['x'])

    his.append([LA.norm(gk,2),hinge_loss(x0,omega,z,l,N),hinge_loss(x0,omega_test,z_test,l,N_test)])
    print('Norma gk:',LA.norm([gk],2))
    if((LA.norm(gk,2)<vopt)and(e1<eopt))or(k==350):
        return x0,his
    else:

          if (LA.norm(gk,2)<v1):
              v1*=theta
              e1*=mi
              k+=1
              print('e1:',e1)
              return gs_algorithm_trust(x0, vopt, eopt, beta, gamma, mi, theta, e1,\
                                        v1, m,omega,omega_test,z,z_test,l,N,N_test, k,his)

          else:
              dk=-e1*gk/LA.norm(gk,2)
              t=1
              while(hinge_loss(x0+t*dk,omega,z,l,N)> hinge_loss(x0,omega,z,l,N)-beta*t*e1*LA.norm(gk,2)):
                  t*=gamma
              print('prekinut backtracking, tk=',t)
              k+=1
              if hinge_izvod(x0+t*dk,omega,z,l,N)[1]==True:
                  x0+=t*dk
                  print('novo xk, diferencijabilno')
                  print('e1:',e1)
                  print('v1:',v1)
              else:
                  print('nediferencijabilno')
                  for x in lop(x0+t*dk,e1,m):
                       print('pretraga')
                       if (hinge_loss(x,omega,z,l,N)<hinge_loss(x0,omega,z,l,N)-beta*t*(LA.norm(gk,2))**2)\
                           and (LA.norm(x0+t*dk-x,2)<=np.minimum(t,e1)*LA.norm(dk,2)):
                           print('nadjeno')
                           x0=x
                           break

              return gs_algorithm_trust(x0, vopt, eopt, beta, gamma, mi, theta, e1, v1,\
                                        m, omega,omega_test,z,z_test,l,N,N_test, k,his)
```

Figure A.5: The Trust region gradient sampling algorithm

```
def hinge_loss(x,omega,z,l,N):
    ro=x[-1]
    x=x[:-1]
    predikcija=omega.dot(x)
    maximumi=np.maximum(0,ro-predikcija)
    return l/2*LA.norm(x,2)**2+1/(N)*np.sum(maximumi)-l*ro

def hinge_izvod(x,omega,z,l,N):
    ro=x[-1]
    x=x[:-1]
    y=omega.dot(x)
    maximumi=ro-y
    izvodi_svi = omega
    izvodi_veci=izvodi_svi[maximumi>0]
    izvodi_nula=izvodi_svi[maximumi==0]
    beta=np.random.uniform(0,1,np.size(izvodi_nula))
    if len(izvodi_nula)!=0:
        return np.concatenate((l*x-sum(izvodi_veci)/(N)-\
                               np.dot(izvodi_nula,beta)/(N),\
                               np.array([sum(beta)/(N)+len(izvodi_veci)/(N)-l]))), False
    elif len(izvodi_nula)==0:
        return np.concatenate((l*x-sum(izvodi_veci)/(N),\
                               np.array([len(izvodi_veci)/(N)-l]))), True
```

Figure A.6: Hinge loss for one class SVM

Bibliography

[BLO] Burke, James V., Adrian S. Lewis, and Michael L. Overton, A robust gradient sampling algorithm for nonsmooth, nonconvex optimization, SIAM Journal on Optimization 15.3 , 2005.

[BN] Bourbaki, Nicolas, Éléments de Mathématique: : Espaces vectoriels topologiques, 1953.

[CLA] Clarke, Frank H, Optimization and nonsmooth analysis, Society for Industrial and Applied Mathematics, 1990.

[DK] Dieter Kraft A Software Package for Sequential Quadratic Programming Wiss. Berichtswesen d. DFVLR, 1988

[DU] Danninger-Uchida G.E, Carathéodory Theorem. In: Floudas C.A., Pardalos P.M. (eds) Encyclopedia of Optimization. Springer, Boston, MA, 2001.

[FKK] Ana Friedlander, Nataša Krejić, Nataša Krklec Jerinkić Lectures on Fundamentals of Numerical Optimization University of Novi Sad Faculty of Sciences,2019.
https://www.pmf.uns.ac.rs/studije/epublikacije/matinf

[HK] Dragoslav Herceg, Nataša Krejić Numerička analiza PMF-Departman za Matematiku i Informatiku,2015.

[HOR] Hörmander, L. Sur la fonction d'appui des ensembles convexes dans un espace localement convexe. Ark. Mat. 3, 181–186,1955.

[HP] Olga Hadžić, Stevan Pilipović, Uvod u funkcionalnu analizu Stylos, 1996.

[KIW] Kiwiel, Krzysztof C, Convergence of the gradient sampling algorithm for nonsmooth nonconvex optimization, SIAM Journal on Optimization 18.2, 2007.

[KKO] Nataša Krejić, Natasa Krklec Jerinkić,Tijana Ostojić Minimizing Nonsmooth Convex Functions with Variable Accuracy,2020 arXiv:2103.13651

[LS] Milan Lukić, Miloš Savić Deep Learning Anomaly Detection for Cellular IoT With Applications in Smart Logistics (Version 1.0) http://doi.org/10.5281/zenodo.4686782

[MK] Miloš Kurilić, Osnovi opšte topologije, Univerzitet u Novom Sadu 1998.

[ROC] Rockafellar, Ralph Tyrell, Convex analysis, Princeton university press, 2015.

[TDS] www.towardsdatascience.com, Support Vector Machine — Simply Explained, Link: `towardsdatascience`

[TJ] Tommi Jaakkola, Course materials for 6.867 Machine Learning, Fall 2006. MIT OpenCourseWare (http://ocw.mit.edu/), Massachusetts Institute of Technology

[XJ] Xing, H.-J., Ji, M. Robust one-class support vector machine with rescaled hinge loss function. Pattern Recognition, 84, 152–164, 2016.

Biography

I was born on the 23rd of January, 1998 in Novi Sad, Serbia. With the average grade of 5.0, I have finished elementary school "Prva vojvođanska brigada". Guided by the interest in the field of mathematics, I enrolled in high school "Jovan Jovanović Zmaj", specifically the mathematics course for gifted students. During high school I attended mostly district competitions in mathematics and physics. After the graduation, with the average grade 5.0 in 2016, my already developed love for mathematics leads me to the Faculty of Sciences in Novi Sad. There, I have graduated the theoretical mathematics batchelor course in 2019 with the gpa 9.93, and in the same year at the Faculty of Sciences I enrolled in the applied mathematics master's course (MB). Having passed all of the required and chosen MB classes with the gpa 9.87 I have acquired the right to defend this thesis.
During the studies, I have obtained the Cambridge English - C2 Proficiency diploma. In 2018, I have participated in the modelling seminar at Einstein's Institute of Mathematics in Jerusalem, where I presented the topic: "f-vectors of simplicial polytopes". Additionally, I have been a part of the ECMI modelling week in 2020, at the end of which I, together with two other participants, presented the "Adaptation of the RED WoLF Hybrid Storage System". Expanding what we have done during that week, we have written the paper: "RED WoLF Hybrid Storage System: Adaptation of Algorithm and Analysis of Performance in Residential Dwellings", which is being reviewed at this moment in time.

UNIVERZITET U NOVOM SADU
PRIRODNO-MATEMATIČKI FAKULTET
KLJUČNA DOKUMENTACIJSKA INFORMACIJA

Redni broj:
RBR

Identifikacioni broj:
IBR

Tip dokumentacije: Monografska dokumentacija
TD

Tip zapisa: Tekstualni štampani materijal
TZ

Vrsta rada: Master rad
VR

Autor: Luka Rutešić
AU

Mentor: Dr Nataša Krejić
MN

Naslov rada: Algoritam uzorkovanih gradijenata za rešavanje problema binarne klasifikacije
NR

Jezik publikacije: engleski
JP

Jezik izvoda: srpski/engleski
JI

Zemlja publikovanja: Srbija
ZP

Uže geografsko područje: Vojvodina
UGP

Godina: 2021.
GO

Izdavač: Autorski reprint
IZ

Mesto i adresa: Prirodno-matematički fakultet, Trg Dositeja Obradovića 4, Novi Sad
MA

Fizički opis rada: 9/54/17/2/8/20/1
(broj poglavlja/broj strana/broj citata/broj tabela/broj slika/broj grafika/broj priloga)
FO

Naučna oblast: Matematika
NO

Naučna disciplina: Numerička analiza
ND

Predmetna odrednica/ ključne reči: optimizacija neglatkih funkcija, mašinsko učenje, lakatna funkcija gubitka, konveksna analiza, neglatka analiza, traženje anomalija, klasifikacija, industrija, c4IIot
PO

UDK:

Čuva se: Biblioteka Departmana za matematiku i informatiku, PMF-a u Novom Sadu
ČU

Važna napomena:
VN

Izvod: U master radu opisan je tok rešavanja matematičkog problema u realnom svetu. Rad se može podeliti u tri glavne celine. Prva celina počinje upoznavanjem čitaoca sa mašinskim učenjem i osnovnim principima. Zatim se objašnjava kako funkcioniše metoda pomoćne vektorske mašine (eng. support vector machine) za binarnu klasifikaciju i za traženje anomalija, i uvodi se 'lakat' funkcija gubitka (eng. hinge loss). Model se trenira tako što se minimizuje ova neglatka funkcija, a predložena metoda optimizacije jeste Kiwiel-ov algoritam uzorkovanih gradijenata. U

drugom delu uvodimo Clark-ovu teoriju neglatke analize. Definisan je Clark-ov subdiferencijal, koji predstavlja generalizaciju glatkog diferencijala. Navedene su Lebourg-ova i Caratheodory-jeva teorema, i uvedeni su pojmovi normalnog i tangentnog konusa. Pomoću ove teorije, definisan je postupak uzorkovanih gradijenata. Koristeći pokazane teoreme Lebourg-a i Caratheodory-ja dokazana je konvergencije postupka. Pored glavnog algoritma, navedene su i modifikacije koje su napravljene u cilju poboljšanja praktične implementacije algoritma. Poslednja celina daje numeričke rezultate. Algoritam se koristi za binarnu klasifikaciju četiri skupa podataka i predviđanje labela test skupova. Prikazani su grafici konvergencije i dati su rezultati predviđanja. Konačno, predstavljen je industrijski problem, i algoritam je primenjen u cilju traženja anomalija na skupu podataka (iz rada Miloša Savić i Milana Lukića) koji je pravljen za c4IIot projekat.
IZ

Datum prihvatanja teme od NN veća: 16.04.2021.
DP

Datum odbrane:
DO

Članovi komisije:
KO

Predsednik: Dr Nataša Krklec Jerinkić, vanredni profesor, Prirodno-matematički fakultet, Novi Sad

Član: Dr Dušan Jakovetić, vanredni profesor, Prirodno-matematički fakultet, Novi Sad

Član, mentor: Dr Nataša Krejić, redovan profesor, Prirodno-matematički fakultet, Novi Sad

UNIVERSITY OF NOVI SAD
FACULTY OF SCIENCES
KEY WORD DOCUMENTATION

Accession number:
ANO

Identification number:
INO

Document type: Monograph type
DT

Type of record: Printed text
TR

Content code: Master thesis
CC

Author: Luka Rutešić
AU

Mentor: Dr Nataša Krejić
MN

Title: The gradient sampling algorithm for solving binary classification problems
TI

Language of text: English
LT

Language of abstract: Serbian/English
LA

Country of publication: Republic of Serbia
CP

Locality of publication: Vojvodina
LP

Publication year: 2021
PY

Publisher: author's reprint
PU

Publication place: Department of Mathematics and Informatics, Faculty of Sciences, University of Novi Sad, Trg Dositeja Obradovića 4
PP

Physical description: 9/54/17/2/8/20/1
(chapters/pages/literature/tables/pictures/graphics/appendices)
PD

Scientific field: Mathematics
SF

Scientific discipline: Numerical analysis
SD

Subject/ Key words: nonsmooth function optimization, machine learning, hinge loss, convex analysis, nonsmooth analysis, anomaly extraction, classification, industry, c4IIot

SKW

UC:

Holding data: Department of Mathematics and Informatics' Library, Faculty of Sciences, Novi Sad
HD

Note:
N

Abstract: In this thesis, the cycle of a mathematics problem in the real world is presented. The paper can be split into three parts. The first begins with introducing the reader to the machine learning and its basic principles. Then, the SVM (support vector machine) method for binary classification and anomaly extraction is described, and hinge loss cost function is introduced. The model is trained by minimizing the nonsmooth function, for which the Kiwiel's gradient sampling method is suggested. In the second part we introduce the Clarke's noonsmooth analysis theory. We

define the Clarke's subdifferential, which is the generalization of the smooth differential. The Lebourg's and Caratheodory's theorems are stated and proved, and the normal and tangent cone are presented. Using this theory, the gradient sampling method is defined. With the help of Lebourg's and Caratheodory's theorem we prove the convergence of the algorithm. Apart from the main algorithm, the modifications are introduced in order to make the practical implementation easier. The last part shows the numerical results. The algorithm is used for binary classification of four datasets and the prediction of the test set labels. Following, the graphs of convergence are shown, and the prediction result are given. In the end, we introduce the main industrial problem, and we apply the anomaly extraction on the given dataset (from the work of Miloš Savić and Milan Lukić), which was made for the c4IIot project.
AB

Accepted by the Scientific Board: 16.04.2021
ASB

Defended on:
DE

Thesis defend board:
DB

President: Dr Nataša Krklec Jerinkić, associate professor, Faculty of Science, Novi Sad

Member: Dr Dušan Jakovetić, associate professor, Faculty of Science, Novi Sad

Member, mentor: Dr Nataša Krejić, full professor, Faculty of Science, Novi Sad