International Journal of Current Science Research and Review
ISSN: 2581-8341
Volume 05 Issue 04 April 2022
DOI: 10.47191/ijcsrr/V5-i4-45, Impact Factor: 5.995
IJCSRR @ 2022
www.ijcsrr.org

Solving A ML Problem Using The Grossone

Dr. Jollanda Shara
University ―Eqrem Cabej‖, Gjirokaster, Albania

ABSTRACT
Machine learning (ML) has grown at a remarkable rate, becoming one of the most popular research directions. It is widely applied in various fields, such as machine translation, speech recognition, image recognition, recommendation system, etc. Optimization problems lie at the heart of most machine learning approaches. So, the essence of most ML algorithms is to build an optimization model and learn the parameters in the objective function from the given data. A series of effective optimization methods were put forward, in order to promote the development of ML. They have improved the performance and efficiency of ML methods. The aim of this paper is to show that, among many other fields, the grossone may be used successfully in the ML. The grossone, the infinite unit of measure, has been proposed by Professor Y. Sergeyev in a number of noticeable works, as the number of elements of the set, N, of natural numbers. It is expressed by the numeral ①. This new computational methodology would allow one to work with infinite and infinitesimal quantities in the "same way" as that working with finite numbers More details about it are given in Section 4. We analyze the SVM from the viewpoint of mathematical programming, solving a numerical example using the grossone. The Iris dataset was chosen for the implementation of the support vector method. This is a wellknown set of data used in the area of ML.

KEYWORDS: ML, grossone, optimization, SVM, linear classifier, hyperplane.

INTRODUCTION

In an informal sense, ML is the task of building a model for some quantity (or function) that we would like to predict, or in other words, learn. The model is usually built from a set of "training" data for which the corresponding quantity of interest is known. Later, the obtained model is used to predict on new or unknown data, where we will then evaluate the performance of the obtained model. Until now, this task description spectacularly mirrors classical regression, which is not a coincidence. Concrete practical examples of such ML questions include classifying handwritten characters, reconstructing radio signals from very noisy sources, detecting a disease from MRI brain images, recommending movies or other products depending on personal ratings given to other items, ranking websites in a search engine based on their text content, modeling the terrain from the data from the sensor of an autonomous car, and predicting climate parameters or stock prices based on historical data, as well as many other applications. A learning algorithm tries to learn a function given a set of data. Generally, given more data, a learning algorithm should ideally learn the function better. In other words, its performance should improve after looking at more data. One of the important class of learning algorithms is the class of Supervised learning algorithms. For supervised learning, given training data (sample inputs and outputs of the function on those inputs), our task is to learn the function. Without any assumptions on the type of function, it is hard to learn the function. In almost all cases, we assume that the function belongs to a class of functions (linear, quadratic etc.) and hence is specified by some parameters. So, the problem translates to estimating the parameters, given a set of training data. Overall, the main steps of ML are to build a model hypothesis, define the objective function, and solve the maximum or minimum of the objective function to determine the parameters of the model. In these three vital steps, the first two steps are the modeling problems of ML, and the third step is to solve the desired model by optimization methods. There are now many flavors of mathematical programs: linear, quadratic, semi-definite, semiinfinite, integer, nonlinear, goal, geometric, fractional, etc. For example, linear programs have a linear objective and linear constraints. Linear programming is not a programming language like C++, Java, or Visual Basic. Linear programming can be defined as:

"A mathematical method to allocate scarce resources to competing activities in an optimal manner when the problem can be expressed using a linear objective function and linear inequality constraints."

Linear programming is a special case of mathematical programming (mathematical optimization). Now linear programming is a subset of ML known as supervised learning. In a supervised learning, the system knows the patterns and the pattern is well defined based on previous data and information. A more complete description of these problems can be obtained from the mathematical programming glossary (www.cudenver.edu/~hgreenbe/glossary/) and the NEOS optimization guide (www-fp.mcs.anl.gov/otc/Guide/). Each flavor of mathematical programming is a different research area in itself with extensive theory and algorithms. In the sequel we give some definitions which will be used in the paper.

International Journal of Current Science Research and Review
ISSN: 2581-8341
Volume 05 Issue 04 April 2022
DOI: 10.47191/ijcsrr/V5-i4-45, Impact Factor: 5.995
IJCSRR @ 2022
www.ijcsrr.org

Definitions:

1. A separating hyperplane is any hyperplane that classifies perfectly the training data.

2. The margin ρ of a separating hyperplane is the distance from the hyperplane to the closest x_i.

$$\rho(w,b) = min\frac{|w^T x+b|}{\|w\|} \quad (1)$$

3. The optimal separating hyperplane is the separating hyperplane whose margin is maximal.

$$(w^*, b^*) = \arg\max \rho(w,b) \quad (2)$$

1. ML&OPTIMIZATION

Almost all ML algorithms can be formulated as an optimization problem to find the extremum of an objective function. Building models and constructing reasonable objective functions are the first step in ML methods. With the determined objective function, appropriate numerical or analytical optimization methods are usually used to solve the optimization problem.
According to the modeling purpose and the problem to be solved, ML algorithms can be divided into supervised learning, semi-supervised learning, unsupervised learning, and reinforcement learning. Particularly, supervised learning is further divided into the classification problem (e.g., sentence classification [4], image classification [5], etc.) and regression problem; unsupervised learning is divided into clustering and dimension reduction [6], among others. For example, the optimization problems in supervised learning have one of the following general form:
The goal is to find an optimal mapping function $f(x)$ to minimize the loss function of the training samples,

$$\min_\theta \frac{1}{N}\sum_{i=1}^{N} L\left(y^i, f(x^i,\theta)\right) \quad (3)$$

where N is the number of training samples, θ is the parameter of the mapping function, x^i is the feature vector of the ith samples, y^i is the corresponding label, and L is the loss function.
There are many kinds of loss functions in supervised learning, such as the square of Euclidean distance, crossentropy, contrast loss, hinge loss, information gain and so on. For regression problems, the simplest way is using the square of Euclidean distance as the loss function, that is, minimizing square errors on training samples. But the generalization performance of this kind of empirical loss is not necessarily good. Another typical form is structured risk minimization, whose representative method is the support vector machine. On the objective function, regularization items are usually added to alleviate overfitting, e.g., in terms of L_2 norm,

$$\min_\theta \frac{1}{N}\sum_{i=1}^{N} L\left(y^i, f(x^i,\theta)\right) + \lambda\|\theta\|_2^2 \quad (4)$$

where λ is the compromise parameter, which can be determined through cross-validation.
We observe that the relationship between available mathematical programming models and ML models has been increasingly combined. The adaptation of mathematical programming models and algorithms has helped ML research advance. Researchers in neural networks went from backpropagation in (Rummelhart et al., 1986) to exploring the use of various unconstrained nonlinear programming techniques such as discussed in (Bishop, 1996). The fact that backpropagation worked well in turn stimulated mathematical programmers to work on stochastic gradient descent to better understand its properties, as in (Mangasarian and Solodov, 1994). With the advent of kernel methods (Cortes and Vapnik, 1995), mathematical programming terms such as quadratic program, Lagrange multipliers and duality are now very familiar to competent ML students. ML researchers are designing novel models and methods to exploit more branches of the mathematical programming tree with a special emphasis on constrained convex optimization. The special topic reflects the diversity of mathematical programming models being employed in ML. We see how recent advances in mathematical programming have allowed rich new sets of ML models to be explored without initial worries about the underlying algorithm. In turn, ML has motivated advances in mathematical programming: the optimization problems arising from large scale ML and data mining far exceed the size of the problem typically reported in the mathematical programming literature. The interplay of optimization and ML is complicated by the fact that ML mixes modeling and methods. In that matter, ML is much like operations research (OR). Mathematical programming/optimization is historically a subfield of OR. OR is concerned with modeling a system. Mathematical programming is concerned with analyzing and solving the model. Both OR and ML analysts address real world problems by formulating a model, deriving the core optimization problem, and using mathematical programming to solve it. According to (Radin, 1998) an OR analyst must trade off tractability – "the degree to which the model admits convenient analysis" and validity – "the degree to which inferences drawn from the model

International Journal of Current Science Research and Review
ISSN: 2581-8341
Volume 05 Issue 04 April 2022
DOI: 10.47191/ijcsrr/V5-i4-45, Impact Factor: 5.995
IJCSRR @ 2022
www.ijcsrr.org

hold for real systems". So at a high level the OR and ML analysts face the same validity and tractability dilemmas and it is not surprising that both can exploit the same optimization toolbox. (see [2])

2. LINEAR CLASSIFIERS

Linear classification is a useful tool in ML and data mining. In contrast to nonlinear classifiers such as kernel methods, which map data to a higher dimensional space, linear classifiers directly work on data in the original input space. While linear classifiers fail to handle some inseparable data, they may be sufficient for data in a rich dimensional space. For example, linear classifiers have shown to give competitive performances on document data with nonlinear classifiers. An important advantage of linear classification is that training and testing procedures are much more efficient. Therefore, linear classification can be very useful for some large-scale applications. Recently, the research on linear classification has been a very active topic. (see [7])
The classification problem is sometimes called supervised learning, because the method operates under supervision [8]. The goal of the supervised learning is to derive a mapping (function) which not only can correctly describe the data in the training set, but more importantly it is able to generalize from the training set to the unobserved situations. Since the main application of the classifier models is the prediction, the generalization ability of the learning algorithms are without doubt the most important property that distinguishes a good classifier from the bad one. [9]
The function derived from the supervised learning introduces a border between two classes – we say it forms a decision boundary or decision surface.
Depending on the shape of this decision boundary we distinguish linear classifiers and non-linear classifiers. The linear classifiers represent a wide family of algorithms, whose common characteristic is that the decision is based on the linear combination of the input variables [10]. Logistic regression and Support vector machines, can be regarded as members of this wide family.
So, to put it simple, in linear classification, we seek to divide the two classes by a linear separator in the feature space. If $p = 2$, the separator is a line, if $p = 3$ it's a plane, and in general it's a $(p-1)$- dimensional hyper-plane in a p-dimensional space.
Fisher in his classic paper [11] introduced his linear discriminant, which was probably the first linear classifier. The Iris Data set [12], published in the same paper to exhibit the power of his linear discriminant, became eventually the most cited classification data set, used widely for educational and explanation purposes. It demonstrates a simple example of the binary classification problem with 2 explanatory variables and 100 observations. The goal of the classification is to draw a line that could serve as a borderline between the two different classes. Since the data are clearly linearly separable, it does not appear to be much a challenging task. (see [19])
If we want to write a function which does linear classification, it would look something like this:

```
classify.linear = function(x,w,b) {
distance.from.plane = function(z,w,b) { sum(z*w) + b }
distances = apply(x, 1, distance.from.plane)
return(ifelse(distances < 0, -1, +1))
}
```

3. LINEAR REGRESSION, PERCEPTRON AND SVM

For convenience, assume that $Y \in \{-1, +1\}$.

A linear classifier has the form

$$f(x) = sign\{w^T x + b\} \qquad (5)$$

where $w \in R^d, b \in R$.

In addition to be very useful as a result of their own ability, linear classifiers are at the heart of many discrimination rules, such as SVM, decision trees and neural networks. Approaches to linear classification are numerous. In fact, a study of linear classifiers shows that they are widely used in the most of various algorithms and principles of supervised learning.

3.1. LINEAR REGRESSION

In regression, we observe $(x_i, y_i), i = 1, 2, \ldots, n$ where $x_i \in R^d, y_i \in R$. It is assumed that

International Journal of Current Science Research and Review
ISSN: 2581-8341
Volume 05 Issue 04 April 2022
DOI: 10.47191/ijcsrr/V5-i4-45, Impact Factor: 5.995
IJCSRR @ 2022
www.ijcsrr.org

$$Y = h(X) + \epsilon \quad (6)$$

where h is a deterministic function and ϵ is zero-mean noise. The goal is to estimate the regression function

$$h(x) = E\{Y|X = x\} \quad (7)$$

In linear regression, it is assumed that

$$h(x) = w^T x + b \quad (8)$$

To apply linear regression to classification, we treat the labels $y_i \in \{-1, 1\}$ as the response variables in a regression problem. In fact, we are interested only in the final decision

$$sign\{w^T x + b\} \quad (9)$$

The standard approach of linear regression is to minimize the squared error

$$\sum[y_i - (w^T x_i + b)]^2 \quad (10)$$

Writing it in matrix notation as follows

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, \boldsymbol{w} = \begin{bmatrix} b \\ w_1 \\ \vdots \\ w_d \end{bmatrix}, \boldsymbol{X} = \begin{bmatrix} 1 & x_{11} & \dots & x_{1d} \\ 1 & x_{21} & \dots & x_{2d} \\ & & \vdots & \\ 1 & x_{n1} & \dots & x_{nd} \end{bmatrix} \quad (11)$$

we seek $\boldsymbol{w}$ minimizing

$$\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 \quad (12)$$

Hence,

$$\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \quad (13)$$

Recall that, a separating hyperplane is any hyperplane that classifies perfectly the training data.

Assume that such a hyperplane exists. The problem is to find one.

Some geometry:

Let $z \in R^d$ and let w, b define a hyperplane. Let us find the distance of z from the hyperplane $\{x \in R^d : w^T x + b = 0\}$.

We can write

$$z = z_0 + r\frac{w}{\|w\|} \quad (14)$$

where $w^T z_0 + b = 0$ and r may be negative. Then,

$$w^T z + b = w^T z_0 + w^T\left(r\frac{w}{\|w\|}\right) + b = w^T\left(r\frac{w}{\|w\|}\right) \quad (15)$$

Now, it is clear that

International Journal of Current Science Research and Review
ISSN: 2581-8341
Volume 05 Issue 04 April 2022
DOI: 10.47191/ijcsrr/V5-i4-45, Impact Factor: 5.995
IJCSRR @ 2022
www.ijcsrr.org

$$r = \frac{w^T z + b}{\|w\|} \quad (16)$$

which is the socalled "signed distance".

3.2. ROSENBLATT'S PERCEPTRON.

The Perceptron algorithm was invented by Frank Rosenblatt in 1958. The algorithm has a bit of a feed-back quality: it starts with an initial guess as to the separating plane's parameters, and then updates that guess when it makes mistakes. Without loss of generality, we can take the initial guesses to be $\vec{w} = 0,\ b = 0$.

```
perceptron = function(x, y, learning.rate=1) {
  w = vector(length = ncol(x)) # Initialize the parameters
  b = 0
  k = 0 # Keep track of how many mistakes we make
  R = max(euclidean.norm(x))
  made.mistake = TRUE # Initialized so we enter the while loop
  while (made.mistake) {
    made.mistake=FALSE # Presume that everything's OK
    for (i in 1:nrow(x)) {
      if (y[i] != classify.linear(x[,i],w,b)) {
        w <- w + learning.rate * y[i]*x[,i]
        b <- b + learning.rate * y[i]*R^2
        k <- k+1
        made.mistake=TRUE # Doesn't matter if already set to TRUE previously
      }
    }
  }
  return(w=w,b=b,mistakes.made=k)
}
```

The perceptron learning algorithm seeks to minimize the total distance of misclassified points from the decision boundary. Assume that the classes are labeled $+1$ and -1. Hence, x_i is misclassified iff the condition

$$y_i(w^T x_i + b) < 0 \quad (17)$$

is satisfied. Denote by M the set of misclassified points. Then, the total (unsigned) distance of misclassified points from the decision boundary is proportional with

$$D(w, b) = -\sum_{i \in M} y_i(w^T x_i + b) \quad (18)$$

The perceptron learning algorithm minimizes $D(w, b)$ using the stochastic gradient descent.

The gradient of $D(w, b)$ is given by

$$\frac{\partial D}{\partial w} = -\sum_{i \in M} y_i x_i \quad (19)$$

$$\frac{\partial D}{\partial b} = -\sum_{i \in M} y_i \quad (20)$$

Instead of stepping in the opposite direction of the gradient, we visit each point of M in some random order and update the hyperplane in each step, as follows:

International Journal of Current Science Research and Review
ISSN: 2581-8341
Volume 05 Issue 04 April 2022
DOI: 10.47191/ijcsrr/V5-i4-45, Impact Factor: 5.995
IJCSRR @ 2022
www.ijcsrr.org

$$w^{new} = w^{old} + \gamma y_i x_i \quad (21)$$

$$b^{new} = b^{old} + \gamma y_i \quad (22)$$

where $\gamma > 0$ is the learning rate.

Notice that:

- If the data is linearly separable, then a separating hyperplane is obtained after a finite number of steps.
- This finite number can be large, especially when the gap between classes is small.
- The final solution depends on the initialization.
- If the data are not separable, the algorithm will not converge.

Other gradient descent methods

Hypothetically, we would like to select w, b to minimize the training error

$$\frac{1}{n}\sum_{i=1}^{n} I_{\{y_i(w^T x_i+b)<0\}} \quad (23)$$

Unfortunately, the function $I_{\{t<0\}}$ is not differentiable. The basic idea is to replace $I_{\{t<0\}}$ by a function $\varphi(t)$ that has qualitatively similar properties with $I_{\{t<0\}}$ but is differentiable or convex. Then we can minimize

$$\frac{1}{n}\sum_{i=1}^{n} \varphi\big(y_i(w^T x_i + b)\big) \quad (24)$$

by gradient descent.

The function $\varphi(t)$ is called loss function.

As we know, a linear program has the form:

$$\min_{u} c^T u$$

$$s.t. \quad Au \leq a \quad (25)$$

where $u \in R^p, c \in R^p, A \in R^{q\times p}, a \in R^q$.

It turns out that we can compute a separating hyperplane using linear programming. We can follow these steps.

First, as we saw above the perceptron criterion is

$$D(w,b) = -\sum_{i\in M} y_i(w^T x_i + b) \quad (26)$$

and we may write

$$D(w,b) = \frac{1}{n}\sum_{i=1}^{n} \varphi\big(y_i(w^T x_i + b)\big) \quad (27)$$

where $\varphi(t) = \max\{-t, 0\}$.

This is optimized by solving:

International Journal of Current Science Research and Review
ISSN: 2581-8341
Volume 05 Issue 04 April 2022
DOI: 10.47191/ijcsrr/V5-i4-45, Impact Factor: 5.995
IJCSRR @ 2022
www.ijcsrr.org

$$\min_{w,b,\xi} \frac{1}{n}\sum_{i=1}^{n} \xi_i$$

$$s.t.\ \ y_i(w^T x_i + b) \geq -\xi_i$$

$$\xi_i \geq 0 \qquad (28)$$

which is a linear program. But, it can be noticed that in this program $w = 0, b = 0, \xi = 0$ is a trivial solution and this is not a useful hyperplane. The constraint $w \neq 0$ is difficult to be included directly into a linear program. So, we'll proceed as follows:

If $(x_1, y_1), \ldots, (x_n, y_n)$ are linearly separable, then

$$y_i(w^T x_i + b) > 0, \forall i \qquad (29)$$

for some w, b.

By rescaling w, b, a separating hyperplane satisfies:

$$y_i(w^T x_i + b) \geq 1, \forall i \qquad (30)$$

The idea now is to minimize the sum of violations of this constraint:

$$\min_{w,b,\xi} \frac{1}{n}\sum_{i=1}^{n} \xi_i$$

$$s.t.\ \ y_i(w^T x_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0 \qquad (31)$$

Let w^*, b^*, ξ^* be the optimal solution. The linear program enjoys these properties:

- A separating hyperplane exists iff $\sum_{i=1}^{n} \xi_i = 0$. If one exists, it will be found.
- If a separating hyperplane does not exist, w^*, b^* is still sensible.
- $w^* = 0$ is not a solution.

Note that the perceptron corresponds to

$$\varphi(t) = \max\{-t, 0\} \qquad (32)$$

while our modified criterion corresponds to

$$\varphi(t) = \max\{-(t-1), 0\} \qquad (33)$$

We can now express the LP for linear classification in the standard form of a LP.

The variable is:

$$u = [w_1\ \ w_2 \ldots w_d\ \ b\ \ \xi_1\ \ \xi_2\ \ \ldots\ \xi_n]^T \qquad (34)$$

The objective function is $c^T u$ where

International Journal of Current Science Research and Review
ISSN: 2581-8341
Volume 05 Issue 04 April 2022
DOI: 10.47191/ijcsrr/V5-i4-45, Impact Factor: 5.995
IJCSRR @ 2022
www.ijcsrr.org

$$c = \begin{bmatrix} 0 & 0 \dots 0 & \frac{1}{n} & \frac{1}{n} \dots & \frac{1}{n} \end{bmatrix}^T \quad (35)$$

The matrix $A_{2n\times(d+1+n)}$ is

$$A = \begin{bmatrix} y_1x_{11} & \dots & y_1x_{1d} & y_1 & 1 & 0 & \dots & 0 \\ y_2x_{21} & \dots & y_2x_{2d} & y_2 & 0 & 1 & \dots & 0 \\ & & \vdots & & & & & \\ y_nx_{n1} & \dots & y_nx_{nd} & y_n & 0 & 0 & \dots & 1 \\ 0 & \dots 0 & 0 & 0 & 1 & 0 & \dots & 0 \\ & & \vdots & & & & & \\ 0 & \dots & 0 & 0 & 0 & 0 & \dots 0 & 1 \end{bmatrix} \quad (36)$$

and

$$a = [1 \ 1 \dots 1 \ 0 \ 0 \dots 0]^T \quad (37)$$

where a is a $(2n \times 1)$-column matrix.

Then, the constraint will be:

$$Au \leq -a \quad (38)$$

The LP linear classifier was developed by Olvi Mangasarian in 1965.

The methods described so far for finding a separating hyperplane do not pay attention to which hyperplane is produced. Yet, if one separating hyperplane exists then infinitely many do. So, which one should we choose? As we mentioned above

The margin ρ of a separating hyperplane is the distance from the hyperplane to the closest x_i.

$$\rho(w,b) = min\frac{|w^Tx + b|}{\|w\|}$$

and the optimal separating hyperplane is the separating hyperplane whose margin is maximal.

$$(w^*, b^*) = \arg\max \rho(w,b)$$

3.3. SVM

The Support Vector Machine (SVM) is a linear classifier that can be viewed as an extension of the Perceptron developed by Rosenblatt. The Perceptron guaranteed that you find a hyperplane if it exists. The SVM finds the maximum margin separating hyperplane.

3.3.1. MARGIN

We already saw the definition of a margin in the context of the Perceptron. Recall that a hyperplane is defined through w, b as a set of points such that

$$\mathcal{H} = \{x|w^Tx + b = 0\} \quad (39)$$

Let the margin ρ be defined as the distance from the hyperplane to the closest point across both classes.

International Journal of Current Science Research and Review
ISSN: 2581-8341
Volume 05 Issue 04 April 2022
DOI: 10.47191/ijcsrr/V5-i4-45, Impact Factor: 5.995
IJCSRR @ 2022
www.ijcsrr.org

By definition, the margin and hyperplane are scale invariant:

$$\rho(\beta w, \beta b) = \rho(w, b), \forall \beta \neq 0 \qquad (40)$$

Note that if the hyperplane is such that ρ is maximized, it must lie right in the middle of the two classes. In other words, ρ must be the distance to the closest point within both classes. (If not, you could move the hyperplane towards data points of the class that is further away and increase ρ, which contradicts that ρ is maximized.)

3.3.2. MAX MARGIN CLASSIFIER

We can formulate our search for the maximum margin separating hyperplane as a constrained optimization problem. The objective is to maximize the margin under the constraints that all data points must lie on the correct side of the hyperplane:

$$\underbrace{\max_{w,b} \rho(w, b)}_{maximise \;\; margin}$$

$$s.t. \quad \underbrace{y_i(w^T x_i + b) \geq 0, \forall i}_{separating \;\; hyperplane} \qquad (41)$$

If we plug in the definition of ρ we obtain:

$$\underbrace{\max_{w,b} \frac{1}{\|w\|_2} \min_{x_{i \in D}} |w^T x_i + b|}_{maximise \;\; margin}$$

$$s.t. \quad \underbrace{y_i(w^T x_i + b) \geq 0, \forall i}_{separating \;\; hyperplane} \qquad (42)$$

Because the hyperplane is scale invariant, we can fix the scale of w, b anyway we want. Let us choose it such that

$$\min_{x \in D} |w^T x + b| = 1 \qquad (43)$$

We can add this rescaling as an equality constraint. Then our objective becomes:

$$\max_{w,b} \frac{1}{\|w\|_2} \cdot 1 = \min_{w,b} \|w\|_2 = \min_{w,b} w^T w \qquad (44)$$

(Where we made use of the fact $f(z) = z^2$ is a monotonically increasing function for $z \geq 0$ and $\| w \| \geq 0$, i.e. the w that maximizes $\|w\|_2$ also maximizes $w^T w$.)

The new optimization problem becomes:

$$\min_{w,b} w^T w$$

$$s.t. \quad y_i(w^T x_i + b) \geq 0, \forall i,$$

$$\min_i |w^T x_i + b| = 1 \qquad (45)$$

These constraints are still hard to deal with, however, fortunately, we can show that (for the optimal solution) they are equivalent to a much simpler formulation.

International Journal of Current Science Research and Review
ISSN: 2581-8341
Volume 05 Issue 04 April 2022
DOI: 10.47191/ijcsrr/V5-i4-45, Impact Factor: 5.995
IJCSRR @ 2022
www.ijcsrr.org

$$\min_{w,b} w^T w$$

$$s.t. \quad y_i(w^T x_i + b) \geq 1, \forall i \qquad (46)$$

This new formulation is a quadratic optimization problem. The objective is quadratic and the constraints are all linear. We can be solve it efficiently with any QCQP (Quadratically Constrained Quadratic Program) solver. It has a unique solution whenever a separating hyper plane exists. It also has a nice interpretation: Find the simplest hyperplane (where simpler means smaller $w^T w$) such that all inputs lie at least 1 unit away from the hyperplane on the correct side.

3.3.3. SUPPORT VECTORS

For the optimal w, b pair, some training points will have tight constraints, i.e.

$$y_i(w^T x_i + b) = 1 \qquad (47)$$

(This must be the case, because if for all training points we had a strict $>$ inequality, it would be possible to scale down both parameters w, b until the constraints are tight and obtained an even lower objective value.) We refer to these training points as support vectors. Support vectors are special because they are the training points that define the maximum margin of the hyperplane to the data set and they therefore determine the shape of the hyperplane. If you were to move one of them and retrain the SVM, the resulting hyperplane would change. The opposite is the case for non-support vectors (provided you don't move them too much, or they would turn into support vectors themselves). This will become particularly important in the dual formulation for Kernel-SVMs.

3.3.4. SVM WITH SOFT CONSTRAINTS

If the data is low dimensional it is often the case that there is no separating hyperplane between the two classes. In this case, there is no solution to the optimization problems stated above. We can fix this by allowing the constraints to be violated ever so slight with the introduction of slack variables:

$$\min_{w,b} w^T w + C \sum_{i=1}^{n} \xi_i$$

$$s.t. \quad y_i(w^T x_i + b) \geq 1 - \xi_i, \forall i$$

$$\xi_i \geq 0, \forall i \qquad (48)$$

The slack variable ξ_i allows the input x_i to be closer to the hyperplane (or even be on the wrong side), but there is a penalty in the objective function for such "slack". If C is very large, the SVM becomes very strict and tries to get all points to be on the right side of the hyperplane. If C is very small, the SVM becomes very loose and may "sacrifice" some points to obtain a simpler (i.e. lower $\|w\|_2^2$) solution.

3.3.5. UNCONSTRAINED FORMULATION

Let us consider the value of ξ_i for the case of $C \neq 0$. . Because the objective will always try to minimize ξ_i as much as possible, the equation must hold as an equality and we have:

$$\xi_i = 1 - y_i(w^T x_i + b) \ \ if \ \ y_i(w^T x_i + b) < 1 \text{ and } \xi_i = 0 \ \ if \ \ y_i(w^T x_i + b) \geq 1 \qquad (49)$$

International Journal of Current Science Research and Review
ISSN: 2581-8341
Volume 05 Issue 04 April 2022
DOI: 10.47191/ijcsrr/V5-i4-45, Impact Factor: 5.995
IJCSRR @ 2022
www.ijcsrr.org

This is equivalent to the following closed form:

$$\xi_i = \max(0, 1 - y_i(w^T x_i + b)) \quad (50)$$

If we plug this closed form into the objective of our SVM optimization problem, we obtain the following unconstrained version as loss function and regularizer:

$$\min_{w,b} \underbrace{w^T w}_{l_2-regulalizer} + C \sum_{i=1}^{n} \underbrace{\max(0, 1 - y_i(w^T x_i + b))}_{hinge-loss} \quad (51)$$

This formulation allows us to optimize the SVM parameters (w, b) just like logistic regression (e.g. through gradient descent). The only difference is that we have the hinge-loss instead of the logistic loss. (see [20])

4. RESULTS

Several years ago, Professor Y. Sergeyev, in a book and in a series of prominent papers [14, 15, 16, 17] has proposed a new approach to infinite and infinitesimal numbers. He has introduced a new infinite unit of measure (the numeral grossone, indicated by ①) as the number of elements of the set of the natural numbers, showing that it is possible to work successfully with infinite and infinitesimal quantities using it to solve many problems in the field of applied and theoretical mathematics, and not only. In this new numeral system, there is the possibility to handle infinite and infinitesimal numbers as particular cases of a single structure. This offers a new perspective and different approaches to important aspects of mathematics such as sums of series (in particular, divergent series), limits, derivatives, etc.
The new numeral grossone can be introduced by describing its properties (in a similar way as it is done in the past with the introduction of 0 to switch from natural to integer numbers). The Infinity Unit Axiom postulate (IUA) [15, 14] is composed of three parts: Infinity, Identity, and Divisibility:
• *Infinity.* Any finite natural number n is less than grossone, i.e., $n < ①$.
• *Identity.* The following relationships link ① to the identity elements 0 end 1

$$0 \cdot ① = ① \cdot 0 = 0, ① - ① = 0, \frac{①}{①} = 1, ①^0 = 1, 1^{①} = 1, 0^{①} = 0 \quad (52)$$

• *Divisibility.* For any finite natural number n, the sets $N_{k,n}$, $1 \le k \le n$,

$$N_{k,n} = k, k + n, k + 2n, k + 3n, \ldots, 1 \le k \le n,$$
$$\cup_{k=1}^{n} N_{k,n} = N \quad (53)$$

have the same number of elements indicated by $\frac{①}{n}$.
The axiom above states that the infinite number ①, greater than any finite number, behaves as any natural number with the elements 0 and 1. Moreover, the quantities $\frac{①}{n}$ are integers for any natural n. This axiom is added to the standard axioms of real numbers and, therefore, all standard properties (commutative, associative, existence of inverse, etc.) also apply to ①. Sergeyev [16, 17] also defines a new way to express the infinite and infinitesimal numbers using a register similar to traditional positional number system, but with base number ①. A number C in this new system can be constructed by subdividing it into groups corresponding to powers of ① and has the following representation:

$$C = c_{p_m} ①^{p_m} + \ldots + c_{p_1} ①^{p_1} + c_{p_0} ①^{p_0} + c_{p_{-1}} ①^{p_{-1}} + \ldots + c_{p_{-k}} ①^{p_{-k}} . \quad (54)$$

where the quantities c_i (the grossdigits) and p_i (the grosspowers) are expressed by the traditional numerical system for representing finite numbers (for example, floating point numbers). The grosspowers are sorted in descending order:

$$p_m > p_{m-1} > \ldots > p_1 > p_0 > p_{-1} > \ldots p_{-(k-1)} > p_{-k}$$

with $p_0 = 0$.
In this new numeral system, finite numbers are represented by numerals with only one grosspower $p_0 = 0$. Infinitesimal numbers are represented by numeral C having only negative finite or infinite grosspowers. The simplest infinitesimal number is $①^{-1}$ for which

$$①^{-1}① = ①①^{-1} = 1 \quad (55)$$

International Journal of Current Science Research and Review
ISSN: 2581-8341
Volume 05 Issue 04 April 2022
DOI: 10.47191/ijcsrr/V5-i4-45, Impact Factor: 5.995
IJCSRR @ 2022
www.ijcsrr.org

We note that infinitesimal numbers are not equal to zero. Infinite numbers are expressed by numerals having at least one finite or infinite grosspower greater than zero.
The newly proposed numeral system pays close attention to its numerical aspects and to applications. The Infinity Computer proposed by Sergeyev is able to execute computations with infinite, finite, and infinitesimal numbers numerically (not symbolically) in a novel framework.
In the following example we use grossone to solve a problem presented in the literature. (see [18])

Two arbitrary points from the iris dataset are selected for analysis, and the optimal hyperplane is calculated.
Let, first, A(12, 4.8), B(100, 6.3) be two arbitrary points where *x* – index in the dataset, *y* –length of the sepal.
The point A will be considered negative and B positive, thus $yA = 1\,, yB = -1$.
We need to find the optimal separate hyperplane.
It is known that any hyperplane can be described as:

$$wx + b = 0 \quad (56)$$

where *w* – normal vector to the hyperplane and $\frac{b}{\|w\|}$ - perpendicular distance from the hyperplane to the origin.
To find $\|w\|$ and *b* dual form should be introduced. It contains a quadratic objective function with constraints as follows:

$$\max_a L_D = \sum_{i=1}^{L} a_i - \frac{1}{2}\sum_{i,j} a_i\, a_j\, y_i y_j \langle x_i, x_j \rangle \quad (57)$$

if $\sum_{i=1}^{L} a_i\, y_i = 0, \quad a_i \geq 0, \forall i.$

After data substitution

$$\max_a L_D = \sum_{i=1}^{L} a_i - \frac{1}{2}\sum_{i,j} a_i\, a_j\, y_i y_j \langle x_i, x_j \rangle =$$
$$a_1 + a_2 -$$
$$\frac{1}{2}\left(a_1 a_1 * 1 * 1 * \langle \binom{12}{4.8}, \binom{12}{4.8} \rangle + 2 * a_1 a_2 * 1 * (-1) * \langle \binom{12}{4.8}, \binom{100}{6.3} \rangle + a_2 a_2 * (-1) * (-1) * \langle \binom{100}{6.3}, \binom{100}{6.3} \rangle\right) = a_1 +$$
$$a_2 - \frac{1}{2}\,(167.04a_1^2 - 2460.48a_1 a_2 + 10039.69a_2^2) \quad (58)$$

Using the method proposed in the literature (see [13], Theorem 3.3) we can solve this problem as follows:

First, we consider the function f:

$$f = x_1 + x_2 - \frac{1}{2}\,(167.04x_1^2 - 2460.48x_1 x_2 + 10039.69x_2^2) + \frac{①}{2}(x_1 - x_2)^2 \quad (59)$$

which must be minimized

subject to $x_1 - x_2 = 0$

Next, we calculate its partial derivatives and form the following system:

$$\frac{\partial f}{\partial x_1} = 0 \Leftrightarrow (-167.04 + ①)x_1 + (1230.24 - ①)x_2 = -1 \quad (60)$$

$$\frac{\partial f}{\partial x_2} = 0 \Leftrightarrow (1230.24 - ①)x_1 + (-10039.36 + ①)x_2 = -1 \quad (61)$$

Solving this system of equations, using the properties of grossone, we obtain:

$$x_1 = x_2 = \frac{2①}{7745.92①} \approx \frac{2①}{7746①} = 0.00025819 \quad (62)$$

International Journal of Current Science Research and Review
ISSN: 2581-8341
Volume 05 Issue 04 April 2022
DOI: 10.47191/ijcsrr/V5-i4-45, Impact Factor: 5.995
IJCSRR @ 2022
www.ijcsrr.org

So:

$$a_1 = x_1 = 0.000258 = \frac{25}{96837}$$

$$a_2 = x_2 = 0.000258 = \frac{25}{96837}$$

Next, let us calculate w and b:

$$w = \sum_{i=1}^{L} a_i x_i y_i = \frac{25}{96837} * 1 * \binom{12}{4.8} + \frac{25}{96837} * (-1) * \binom{100}{6.3} = \begin{pmatrix} \frac{300}{96387} \\ \frac{120}{96837} \end{pmatrix} - \begin{pmatrix} \frac{2500}{96387} \\ \frac{157.5}{96837} \end{pmatrix} = \begin{pmatrix} \frac{2200}{96387} \\ \frac{37.5}{96837} \end{pmatrix}$$

$$b = 1 - \left(\frac{25}{96837} * \left\langle \binom{12}{4.8}, \binom{12}{4.8} \right\rangle - \frac{25}{96837} * \left\langle \binom{12}{4.8}, \binom{100}{6.3} \right\rangle \right) = \frac{96837}{96837} - \left(\frac{4176}{96837} - \frac{30756}{96837} \right) = \frac{96837}{96837} + \frac{26580}{96837} = \frac{123417}{96837}$$

The representation of the hyperplane will be:

$$w_1 x + w_2 y + b = 0 \qquad (63)$$

Substituting the numbers, we obtain

$$-\frac{2200}{96837} x - \frac{37.5}{96837} y + \frac{123417}{96837} = 0 \qquad (64)$$

Comparing the result of the program and using the SVM of the sklearn library, the following result of the program is given:

$$w = [[-0.02272067 \ -0.00038728]]$$
$$b = [1.27450707\ ]$$

Indices of support vectors = [1 0];
Support vectors = [[100 6.3] [12 4.8]];
Number of support vectors for each class = [1 1];
Coefficients of the support vector in the decision function = [0.00025819 0.00025819].
The results are summarized in the following table:

w		Support Vectors (SV)		Indices of SV		Number of SV		Coefficients of SV	
w_1	w_2	I	II	I	II	I	II	I	II
−0.02272067	−0.00038728	100 6.3	12 4.8	1	0	1	1	0.00025819	0.00025819

Table I

The values of w and b are calculated manually and coincide with those calculated with the help of the program, so the calculations are correct. Also, since we have only two points in this case, they act as reference vectors, one for each class.

CONCLUSION

Since its earliest days as a discipline, ML has used many optimization formulations and algorithms. In addition, ML has contributed to optimization, stimulating the development of new optimization approaches that address the significant questions

International Journal of Current Science Research and Review
ISSN: 2581-8341
Volume 05 Issue 04 April 2022
DOI: 10.47191/ijcsrr/V5-i4-45, Impact Factor: 5.995
IJCSRR @ 2022
www.ijcsrr.org

presented by ML applications. This interplay and collaboration continues to deepen, producing a growing literature at the intersection of the two fields and being a very attractive subject for many leading researchers, as well.
Linear classification is a useful tool in ML and data mining. Recently, many research works have developed efficient optimization methods to construct linear classifiers and applied them to some large-scale applications.
Support vector machines (SVMs) are a set of related supervised learning algorithm developed by Vladimir Vapnik in the mid 90's for classification and regression. Support vector machines (SVMs) have been successfully applied to a large number of real-world applications, such as text categorization and handwritten character recognition. The use of optimization methodologies plays a central role in finding solutions of SVMs, in fact the support vector machine (SVM) is the first contact that many optimization researchers had with ML, due to its classical formulation as a convex quadratic program — simple in form, though with a complicating constraint. Taking into account their importance and their closed relationship with the mathematical programming, in this paper, we have described in more details the SVM classifier. We have solved a numerical example from the literature using the ML & SVM methods. The use of grossone has significantly simplified its solution. Even this simple example, shows for another time the large scale of grossone applications due to its elegance and efficiency. We are sure that there are many other problems from ML in which the grossone may be used successfully. So, for future work, we recommend solving the same problem, with respect to the design of new evolutionary, genetic and recommendation models. There are numerous publications on this research area. The interested reader may consult, among others, the references given in this paper. (See [21]-[31]).

REFERENCES

1. Jaggi, Martin, 2011, Sparse Convex Optimization Methods for ML, https://doi.org/10.3929/ethz-a-007050453.
2. Kristin P. Bennett and Emilio Parrado-Hernández, 2006, The Interplay of Optimization and ML Research, *Journal of ML Research* 7, pp. 1265–1281.
3. Shiliang Sun, Zehui Cao, Han Zhu, and Jing Zhao, 2019,A Survey of Optimization Methods from a ML Perspective, arXiv:1906.06821v2 [cs.LG] 23.
4. Y. Kim, 2014, Convolutional neural networks for sentence classification, in *Conference on Empirical Methods in Natural Language Processing*, pp. 1746–1751.
5. D. C. Ciresan, U. Meier, and J. Schmidhuber, 2012, Multi-column deep neural networks for image classification, in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3642–3649.
6. J. A. Hartigan and M. A. Wong, 1979, Algorithm AS 136: A k-means clustering algorithm, *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, pp. 100–108.
7. Guo-Xun Yuan, Chia-Hua Ho, and Chih-Jen Lin, Recent Advances of Large-scale Linear Classification.
8. Witten, I., Frank, E., and Hall, M. 2011, Data Mining: Practical ML Tools and Techniques, The Morgan Kaufmann Series in Data Management Systems, Elsevier Science,.
9. Hamel, L. H. 2011, Knowledge discovery with support vector machines, vol. 3. John Wiley & Sons.
10. Rao, K., and Koolagudi, S. 2012, Emotion Recognition using Speech Features. SpringerBriefs in Electrical and Computer Engineering. Springer.
11. Fisher, R. A. 1936,The use of multiple measurements in taxonomic problems. Annals of eugenics 7, pp. 179-188.
12. Bache, K., and Lichman, M. 2013,Iris data set, http://archive.ics.uci.edu/ml/datasets/Iris.
13. Sonia De Cosmis, Renato De Leone, 2012, The use of Grossone in Mathematical Programming and Operations Research, arXiv:1107.5681v2 [math.OC].
14. Yaroslav D. Sergeyev. 2003, Arithmetic of Infinity. Edizioni Orizzonti Meridionali, CS.
15. Yaroslav D. Sergeyev. 2008, A new applied approach for executing computations with infinite and infinitesimal quantities. *Informatica*, 19(4), pp. 567–596.
16. Yaroslav D. Sergeyev. 2009, Numerical computations and mathematical modelling with infinite and infinitesimal numbers. *Journal of Applied Mathematics and Computing*, 29:177195.
17. Yaroslav D. Sergeyev.2009, Numerical point of view on calculus for functions assuming finite, infinite, and infinitesimal values over finite, infinite, and infinitesimal domains. *Nonlinear Analysis Series A: Theory,Methods &Applications*, 71(12):e1688e1707.
18. Nataliya Boyko and Rostyslav Hlynka, 2021, Application of Machine Algorithms for Classification and Formation of the Optimal Plan, COLINS-2021: 5th International Conference on Computational Linguistics and Intelligent Systems, April 22–23, 2021, Kharkiv, Ukraine.
19. https://svm.michalhaltuf.cz/linear-classifiers/
20. https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote09.html
21. Bhaskaran, S.; Marappan, R.; Santhi, B. Design and Analysis of a Cluster-Based Intelligent Hybrid Recommendation System for E-Learning Applications. Mathematics 2021, 9, 197. https://doi.org/10.3390/math9020197
22. Marappan, R., Sethumadhavan, G. Solving Graph Coloring Problem Using Divide and Conquer-Based Turbulent Particle Swarm Optimization. Arab J Sci Eng (2021). https://doi.org/10.1007/s13369-021-06323-x
23. Bhaskaran, S.; Marappan, R.; Santhi, B. Design and Comparative Analysis of New Personalized Recommender Algorithms with Specific Features for Large Scale Datasets. Mathematics 2020, 8, 1106. https://doi.org/10.3390/math8071106
24. Marappan, R.; Sethumadhavan, G. Complexity Analysis and Stochastic Convergence of Some Well-known Evolutionary Operators for Solving Graph Coloring Problem. Mathematics 2020, 8, 303. https://doi.org/10.3390/math8030303
25. Marappan, R., Sethumadhavan, G. Solution to Graph Coloring Using Genetic and Tabu Search Procedures. Arab J Sci Eng 43, 525–542 (2018). https://doi.org/10.1007/s13369-017-2686-9
26. R. Marappan and G. Sethumadhavan, "Solving channel allocation problem using new genetic algorithm with clique partitioning method," 2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), 2016, pp. 1-4, doi: 10.1109/ICCIC.2016.7919671.
27. R. Marappan and G. Sethumadhavan, "Solution to graph coloring problem using divide and conquer based genetic method," 2016 International Conference on Information Communication and Embedded Systems (ICICES), 2016, pp. 1-5, doi: 10.1109/ICICES.2016.7518911.

International Journal of Current Science Research and Review
ISSN: 2581-8341
Volume 05 Issue 04 April 2022
DOI: 10.47191/ijcsrr/V5-i4-45, Impact Factor: 5.995
IJCSRR @ 2022
www.ijcsrr.org

28. R. Marappan and G. Sethumadhavan, "Divide and conquer based genetic method for solving channel allocation," 2016 International Conference on Information Communication and Embedded Systems (ICICES), 2016, pp. 1-5, doi: 10.1109/ICICES.2016.7518914.
29. Raja Marappan, Gopalakrishnan Sethumadhavan , Solving Fixed Channel Allocation using Hybrid Evolutionary Method, MATEC Web of Conferences 57 02015 (2016) DOI: 10.1051/matecconf/20165702015
30. G. Sethumadhavan and R. Marappan, "A genetic algorithm for graph coloring using single parent conflict gene crossover and mutation with conflict gene removal procedure," 2013 IEEE International Conference on Computational Intelligence and Computing Research, 2013, pp. 1-6, doi: 10.1109/ICCIC.2013.6724190.
31. R. Marappan and G. Sethumadhavan, "A New Genetic Algorithm for Graph Coloring," 2013 Fifth International Conference on Computational Intelligence, Modelling and Simulation, 2013, pp. 49-54, doi: 10.1109/CIMSim.2013.17.

Cite this Article: Dr. Jollanda Shara (2022). Solving A ML Problem Using The Grossone. International Journal of Current Science Research and Review, 5(4), 1226-1240