# Mixed Precision SVM

Edoardo Ruffoli, Franco Terranova

Symbolic and Evolutionary Artificial Intelligence

## Contents

**1 Introduction** **2**

**2 Bound the Primal Solutions** **3**

**3 Bound the Dual Solutions** **6**

**4 Limitation on the value of the parameter $C$** **7**

**5 Matlab implementation** **8**

**6 Benchmarks** **9**

**7 Evaluation** **10**
7.1 Scenarios . . . . . 10
7.2 Smart decision function computation . . . . . 11
7.3 SONAR . . . . . 12
7.3.1 Difference among support vectors . . . . . 17
7.3.2 Summary . . . . . 17
7.4 BNA . . . . . 18
7.4.1 Difference among support vectors . . . . . 21
7.4.2 Summary . . . . . 22
7.5 WDBC - Non-linear case . . . . . 22
7.5.1 Summary . . . . . 24

**8 Final considerations** **24**

**9 Future ideas** **25**
9.1 Unbalanced datasets . . . . . 25
9.2 Non-linear SVM . . . . . 25

# 1 Introduction

Real-time or power-constrained applications employ smaller number formats (posits) to result in smaller memory footprints, higher performance, and lower power consumption. The key task of every efficient implementation is to identify the lowest working precision acceptable to achieve a classification accuracy comparable to a reference implementation in floating-point arithmetic.

The focus of this project is to realize a modified version of the Support Vector Machine optimization problem, taking into account that, during inference, the model will be executed on devices using low precision posits.

In the following sections two possible approaches to solve the problem will be analyzed, both targeting the penalization of high values in magnitude of the parameters of the model, and both considered in order to exploit as much as possible the dense region of the posit ring.

## 2 Bound the Primal Solutions

The first approach to tackle this task is to bound the optimal solutions of the primal problem so that they will belong to a range of values that can be accurately represented using posits.

Considering the linear SVM model with soft margin, the optimal hyperplane is given by a vector $w^*$ and a scalar $b^*$ which can be used in order to obtain, during inference, the output class.

$$f(x) = sign((w^*)^T x + b^*) \tag{1}$$

To obtain proper values of the optimal solutions for a good representation using posits, could be sufficient to add two new constraints that bound the components of the $w$ vector and the scalar $b$. The lower and upper bound should be chosen in accordance to the the posits' precision used at inference time: two possible solutions are [-1,1], in order to use the dense region of the ring, or [-useed, useed], in order to add a dependence on the value of *esbits*.
The modified SVM primal problem is shown below.

$$\begin{aligned} \underset{w,b,\xi}{\text{minimize}} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & 1 - y^1(w^T x^i + b) \leq \xi_i, \quad i = 1,\dots,l, \\ & \xi_i \geq 0, \qquad\qquad\qquad i = 1,\dots,l, \\ & LB \leq w_i \leq UB, \qquad\quad i = 1,\dots,n, \\ & LB \leq b \leq UB, \end{aligned} \tag{2}$$

One of the biggest strengths of SVMs is the possibility to perform a mapping $\phi$ between the input and an implicit feature space.
This consideration allow us to perform classification in case of a non-linearly separable dataset.
The optimal hyperplane will be searched on the new feature space, that can be even of infinite dimensions, where data will be linearly separable. In (3) the modified SVM primal model with the input/feature space mapping is shown.

$$\begin{aligned} \underset{w,b,\xi}{\text{minimize}} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & 1 - y^1(w^T \phi(x^i) + b) \leq \xi_i, \quad i = 1,\dots,l, \\ & \xi_i \geq 0, \qquad\qquad\qquad i = 1,\dots,l, \\ & -LB \leq w_i \leq UB, \qquad\quad i = 1,\dots,h, \\ & -LB \leq b \leq UB, \end{aligned} \tag{3}$$

Since the dimensionality of the feature space can be infinite, the SVM optimization problem is typically solved using the dual problem.
In fact, in the latter, there is no need to explicitly know the mapping $\phi(x)$ but

only the product $\phi(x)^T\phi(x)$, that can be computed using kernel functions. Moreover, the dual will always be a quadratic convex optimization problem. Given $\lambda$, optimal solution of the dual problem, the decision function will be the following:

$$f(x) = sign(\sum_{i=1}^{l} \lambda_i^* y^i k(x^i, x) + b^*) \tag{4}$$

For the reasons above, we need to find the dual problem of the previously modified primal optimization problems.
Considering the Lagrangian function of the linear SVM with soft margin, the following can be derived.

$$\begin{aligned} L(w, b, \xi, \lambda, \mu, \eta, \rho, \theta, \gamma) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i + \sum_{i=1}^{l}\lambda_i[1 - y^i(w^T(x^i) + b) - \xi_i] - \\ \xi\mu + \eta(w - e) + \rho(-w - e) + \theta(b - 1) + \gamma(-b - 1) \\ = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i(C - \lambda_i) + \sum_{i=1}^{l}\lambda_i - \sum_{i=1}^{l}\lambda_i y^i(w^T(x^i) + b) - \xi\mu + \eta(w - e) + \\ \rho(-w - e) + \theta(b - 1) + \gamma(-b - 1) \end{aligned} \tag{5}$$

$$\text{If } \sum_{i=1}^{l}\lambda_i y_i = 0 \text{ and } \lambda_i \in [0, C] \text{ and } \theta = 0 \text{ and } \gamma = 0 \text{ and } \mu = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \text{ then} \tag{6}$$

$$\inf_{w,b,\xi} L(w, b, \xi, \lambda, \mu, \eta, \rho, \theta, \gamma) \neq \infty$$

In this case, in fact, the Lagrangian function does not depend on b, $\xi$, and since this function is strongly convex with respect to w, the *argmin* is given by the unique stationary point.

$$\nabla_w L = w - \sum_{i=1}^{l}\lambda_i y_i x_i - \rho + \eta = 0$$

$$w^* = \sum_{i=1}^{l}\lambda_i y_i x_i + \rho - \eta$$

$$\begin{aligned} \underset{\lambda,\mu,\eta,\rho,\theta,\gamma}{\text{maximize}} \quad & \phi(\lambda,\mu,\eta,\rho,\theta,\gamma) \\ \text{subject to} \quad & \lambda,\eta,\rho,\theta,\gamma >= 0 \end{aligned} \tag{7}$$

$$\phi(\lambda,\mu,\eta,\rho,\theta,\gamma) = \inf_{w,b,\xi} L(w,b,\xi,\lambda,\mu,\eta,\rho,\theta,\gamma) =$$

$$= \begin{cases} -\infty & \text{if } \sum_{i=1}^{l} \lambda_i y_i \neq 0 \text{ or } \lambda_i \notin [0,C] \\ & \text{or } \theta \neq 0 \text{ or } \gamma \neq 0 \text{ or } \mu \neq \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\ \frac{1}{2}\|w\|^2 + \sum_{i=1}^{l} \lambda_i - \sum_{i=1}^{l} \lambda_i y^i w^T x^i & \text{otherwise} \\ +\eta(w-e) + \rho(-w-e) & \end{cases}$$

with $w = \sum_{i=1}^{l} \lambda_i y_i x_i + \rho - \eta$ and the vector $e$ used to denote the vector $\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$.

The term multiplying $\xi$ has been elided in the dual objective function because of the constraint given by the complementarity condition on the model with soft margin: $(C - \lambda_i)\xi_i = 0 \; i = 1, \ldots, l$.
The overall new dual problem will be:

$$\begin{aligned} \underset{\lambda,\rho,\eta}{\text{maximize}} \quad & \frac{1}{2}\|w\|^2 + \sum_{i=1}^{l} \lambda_i - \sum_{i=1}^{l} \lambda_i y_i w^T x_i + \eta(w-e) + \rho(-w-e) \\ \text{subject to} \quad & \sum_{i=1}^{l} \lambda_i y_i = 0, \\ & \lambda_i \in [0,C], \quad i = 1, \ldots, l, \\ & \rho,\eta >= 0 \end{aligned} \tag{8}$$

with $w = \sum_{i=1}^{l} \lambda_i y_i x_i + \rho - \eta$ and the vector $e$ used to denote the vector $\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$.

The dual problem is much more complex consequently to the addition of the bounds to the primal problem, resulting in an involvement of a higher number of variables and a more complex objective function. For this reason, this solution has been discarded.
Moreover, the decision function involved in non-linear SVM requires computations with the dual variables and the scalar $b$ and not with the weight vector $w$,

which means that bounding the weight vector may not be sufficient to guarantee absence of loss of precision in the case of non-linear SVM.

# 3 Bound the Dual Solutions

The second approach consists in bounding the dual variables in the dual problem.

$$\begin{aligned} \underset{\mu,\eta}{\text{maximize}} \quad & -\frac{1}{2}\sum_{i=1}^{L}\sum_{i=1}^{L} y^i y^j (x^i)^T x^j (\mu_i - \eta_i)(\mu_j - \eta_j) + \sum_{i=1}^{L}(\mu_i - \eta_i) \\ \text{subject to} \quad & \sum_{i=1}^{L}(\mu_i - \eta_i) y^i = 0, \\ & 0 \leq \mu_i - \eta_i \leq C, \quad i = 1, \ldots, L, \\ & l \leq \mu_i \leq h, \quad i = 1, \ldots, L, \\ & l \leq \eta_i \leq h, \quad i = 1, \ldots, L \end{aligned} \tag{9}$$

The support vectors, points with the corresponding value of the dual variables higher than 0, whose corresponding value is lower than *minpos*, during the float to posit conversion are not giving contributions to the decision function computation, since converted to 0 when moving to the posit representation.
We cannot simply shift the dual variables range to higher values since we need to have that $\lambda$ can have 0 as possible value, in order to recognise that they are not support vectors.
To mitigate this issue, we can express the dual variable $\lambda$ as a difference of two variables, $\mu$ and $\eta$, both forced to be at least higher than *minpos*.
The upper bound and lower bound of the new dual variables can be properly set as hyper-parameters. Some values that can be used are $\sqrt{minpos}$ or $\sqrt[3]{minpos}$, so that we are sure that they can be well represented with posits.
Using this solution, when the difference between $\mu$ and $\eta$ will be computed in the posit domain, we are sure to obtain exactly the value 0 if the dual variable $\lambda$ is not associated to a support vector.
Once the optimal solutions of the dual will be determined, the weight vector $w$ and the scalar $b$ can be computed with the original formula.

$$w^* = \sum_{i=1}^{L} \lambda_i^* y_i x_i \tag{10}$$

$$b^* = \frac{1}{y_i} - (w^*)^T x_i \tag{11}$$

with $i$ index associated to a support vector.
As a drawback, due to the presence of the sum, this solution does not have a strong guarantee that the weight vector and the value of the scalar b will necessarily be in the dense region.

Moreover, this solution limits our solution to a set of sub-optimal solutions.
However, this second approach seemed to be simpler than the previous one, and it is also effective in the case of non-linear SVMs, representing a more general solution.
The rest of the documentation will analyze this second solution only.

# 4 Limitation on the value of the parameter $C$

It can be proven that this solution introduces a limitation to the values of the parameter $C$ that can be used in our model. Due to the presence of bounds to the new dual variables, the values of the original dual variable $\lambda$ is limited, as proven below.

$$\begin{gathered} l \leq \mu_i \leq h \\ l \leq \eta_i \leq h \\ l - h \leq \mu_i - \eta_i = \lambda_i \leq h - l \\ 0 \leq \lambda_i \leq C \end{gathered}$$

Once determined the worst case for our difference of dual variables, we can notice how the parameter $C$ is overcome by the difference $h$ - $l$ in case it will be higher than this difference.

$$0 = \max(0, l - h) \leq \lambda_i \leq \min(h - l, C)$$

# 5 Matlab implementation

The optimization problems have been solved using Matlab and the Optimization toolbox.
Our model has been converted in the quadratic programming standard form,

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}x^T Q x + c^T x \\ \text{subject to} \quad & Ax \leq b, \\ & A_{eq}x = b_{eq}, \\ & LB \leq x \leq UB \end{aligned} \tag{12}$$

$$x = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_L \\ \eta_1 \\ \vdots \\ \eta_L \end{bmatrix} \in R^{2L*1}$$

$$Z = \begin{bmatrix} y_1 t_1 \dots y_L t_L \end{bmatrix} \in R^{n*L}$$

where $t_1 \dots t_L$ are the points contained in the training dataset.

$$Q = \begin{bmatrix} ZZ^{'} & -ZZ^{'} \\ -ZZ^{'} & ZZ' \end{bmatrix} \in R^{2L*2L}$$

$$c = \begin{bmatrix} -1 \\ \vdots \\ -1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \in R^{2L*1}$$

$$Aeq = \begin{bmatrix} y \\ -y \end{bmatrix} \in R^{2L*1}$$

$$beq = 0$$

$$A = \begin{bmatrix} -I & I \\ I & -I \end{bmatrix} \in R^{2L*2L}$$

$$b = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ C \\ \vdots \\ C \end{bmatrix} \in R^{2L*1}$$

$$LB = \begin{bmatrix} l \\ \vdots \\ l \\ l \\ \vdots \\ l \end{bmatrix} \in R^{2L*1}$$

$$UB = \begin{bmatrix} h \\ \vdots \\ h \\ h \\ \vdots \\ h \end{bmatrix} \in R^{2L*1}$$

This Matlab implementation has been used in order to train our SVM model with additional constraints, leveraging on the IEEE 754 FP64 representation, the default one used by Matlab for real numbers.

# 6 Benchmarks

In this section, all the benchmark datasets used to test our model will be presented.

## WDBC

The *Wisconsin Breast Cancer Dataset* contains information on 569 instances of malignant and benign tumor cells, collected from Wisconsin Hospitals.
It includes 30 features for each instance, which were computed from a digitized image of a fine needle aspirate (FNA) of a breast mass.
These features describe characteristics of the cell nuclei present in the image.
Some of the features include the radius, texture, perimeter, area, smoothness, and compactness of the nuclei.
The dataset features are not normalized and the class distribution is slightly unbalanced, with 357 benign cases and 212 malignant cases.

## SONAR

The *SONAR* dataset is related to the detection of objects in the ocean using sonar signals. It includes 60 features for each of the 208 instances, which represent the strength of the sonar return at different angles. These features are continuous values that describe the echoes from the sonar signals.
Some of the features include the frequency of the return signal, the bandwidth, and the skewness of the signal. The dataset is natively normalized between [0,1] and the class distribution is somewhat balanced, with 97 instances of rocks and 111 instances of mines.

### BNA

The *Banknote Authentication* dataset contains information on 1,372 instances of banknotes, with the goal of developing a model that could accurately determine whether a banknote is authentic or not.

It includes 4 features for each instance, which describe various aspects of the banknote. These features include the variance of the wavelet transformed image, the skewness of the wavelet transformed image, the entropy of the image, and the curtosis of the image. The dataset is not normalized and the class distribution is balanced, with 662 instances of authentic banknotes and 710 instances of inauthentic banknotes.

## 7 Evaluation

For the evaluation of our approach using the different benchmarks, the following decisions have been taken:

- The datasets have been divided, using an holdout method, in training set and test set (70% - 30%) with a stratified approach in order to maintain the original classes distribution. The *train_test_split* function of *scikit-learn* has been used to this purpose.
- The ranges [-1, +1] and [-*useed*, *useed*] have been considered for the normalization of the datasets. Using these two ranges, we obtain many advantages, such as having datasets' feature values to be closer to the dense region of the posit ring. The second normalization technique adds also a dependence on the specific value of *esbits* used.
- Different values of the parameter $C$ have been compared, mainly dependent of the performances obtained in the benchmark datasets.
- The lower bound of the dual variables has been set equal to $\sqrt{minpos}$.
- The upper bound of the dual variables has been varied in the following range: $[\sqrt[3]{minpos}, \sqrt[5]{minpos}, 1]$. Further values have been considered in case of interesting scenarios.
- The configuration parameters targeted with our benchmarks have been *nbits = 8* and *esbits = 0*.

### 7.1 Scenarios

Different possible scenarios have been considered in order to obtain the decision function starting from the optimal dual variables obtained with the model.

- Scenario 0) The SVM dual model has been trained and evaluated using the Matlab tool without posit constraints, in order to show the optimal solution that could be achieved with the original model.

- Scenario 1) The SVM dual model has been trained using the Matlab tool without posit constraints and evaluated on the test set using the posit representation. The dataset and the optimal dual variables have been represented using posits and the decision function has been computed using this latter representation.
- Scenario 2) The SVM dual model has been trained using the Matlab tool with posit constraints and evaluated using the posit representation. The dataset and the optimal dual variables have been represented using posits and the decision function has been computed using this latter representation.
- Scenario 3) The SVM dual model has been trained using the Matlab tool with posit constraints and evaluated using the posit representation. The dataset and the optimal dual variables have been represented using posits, but the decision function has been computed using higher precision types (Double, Float, Posit(16,2)), achieving an actual case of mixed precision.
- Scenario 4) The SVM dual model has been trained using the Matlab tool with posit constraints and evaluated using the posit representation. Starting from the optimal dual variables, the weight vector $w$ and the scalar $b$ have been computed using Matlab and then represented in posits together with the dataset to perform inference.
- Scenario 5) The SVM dual model has been trained using the Matlab tool without posit constraints and evaluated using the posit representation. Starting from the optimal dual variables, the weight vector $w$ and the scalar $b$ have been computed using Matlab and then represented in posits together with the dataset to perform inference.

Scenarios 4 and 5 can only be used in case of linear SVMs, where the actual parameters of the model are the weight vector $w$ and the scalar $b$. In a more general case, such as non-linear SVMs, the dual variables have to be used at inference time.

## 7.2 Smart decision function computation

The computation of the decision function using the dual variables in the posit representation (scenarios 1, 2 and 3) may reach a large deviation from the optimal hyperplane in case the different contributions of the sum involved in the decision function may get further from the dense region of the posit ring.

$$w^* = \sum_{i=1}^{L} \lambda_i^* y_i x_i$$

For this reason a smart algorithm to compute the weight vector $w$ has been used in order to reduce the error in the computation given by the posit representation. This algorithm has been implemented component-wise for our weight vector $w$.

**Algorithm 1** Smart decision function computation

```
products ← array(w.length)
while i ≤ w.size() do
    products_i ← λ_i y_i X_i                    ▷ Partial products to be summed
    i ← i + 1
end while
products ← sort(products)
index ← products.left()
sum ← 0
while True do
    sum ← sum + products_index
    if abs(sum) ≥ UB then
        index ← swapIndex(index)
    else
        if index.hasNext() then
            index ← next(index)
              ▷ Increment if we are in the left, decrement if we are in the right
        else
            return sum
        end if
    end if
end while
```

This algorithm involves the definition of an upper bound and perform sums in a smart manner in order to have the partial sum not overcoming the upper bound in absolute value. In this way, we force the computation to stay as much as possible inside the dense region of the posit ring, since the contributions of positive products will be mitigated with negative products, and viceversa. This algorithm presents though as drawback the dependence of an upper bound whose choice varies the resulting accuracy. Moreover, in case of not fully balanced situation around 0 for our partial products, we may not have many advantages.

## 7.3 SONAR

### Scenario 0

For the SONAR dataset, the value of C that provided the best accuracy is 0.1 and it will be used in all of the following scenarios. The accuracy values shown in Table 1 represent the optimal solutions of the problem using the SVM model without our additional constraints.

### Scenario 1

In this scenario, as previously mentioned, the dual variables obtained using the SVM model without additional constraints will be converted in posit<8>. Note

| SONAR Scenario 0 (C=0.1) | | |
|---|---|---|
| **Constraints** | **Normalization** | **Accuracy** |
| Without | [-1, 1] | 0.76 |
| Without | [-useed, useed] | 0.68 |

Table 1: Accuracy values of the SVM model w/o additional constraints.

that, since the output of the MATLAB *quadprog* function usually contains a lot of zeros that are represented as very low values (e.g. 1e-10), we rounded these values to the exact 0. This trick was needed because the posit library that we employed rounds every value of x such that $0 < x < minpos$ to $minpos$ and so we would have had all the dual variables considered as associated to support vectors. To better evaluate the conversion errors made during the double to posit<8> conversion, in addition to the accuracy, we will use the following two metrics:

$$SVError = \sum_{i=1}^{m} |\lambda_i - \lambda_i'| \tag{13}$$

$$SVAVGError = \frac{1}{m} \sum_{i=1}^{m} |\lambda_i - \lambda_i'| \tag{14}$$

where $\lambda_i$ are the dual variables associated to support vectors represented in doubles, obtained in MATLAB, while $\lambda_i'$ are the dual variables associated to support vectors converted in posit<8>; $m$ is the number of dual variables associated to support vectors.

| SONAR Scenario 1 (C=0.1) | | | | |
|---|---|---|---|---|
| **Constraints** | **Normalization** | **SV Error** | **SV AVG Error** | **Accuracy** |
| Without | [-1, 1] | 0.59039 | 0.00663 | 0.66 |
| Without | [-useed, useed] | 0.46645 | 0.00696 | 0.68 |

Table 2: Inference results with posits of the SVM model w/o additional constraints, using dual variables.

Note that, the lower total SV error obtained with the [-useed, useed] technique is present due to the lower number of support vectors; the AVG SV error is similar in both cases.

The accuracy level obtained using the normalization technique [-useed, useed] is very close to the one obtained in Scenario 0 for the same normalization technique. On the other hand, with the [-1,1] normalization, some relevant errors might have been introduced during the conversion of the dual variables.

**Scenario 2**

For the second scenario, our SVM model with additional posit constraints will be evaluated. We explored, for both normalization techniques, a set of possible upper bound values.

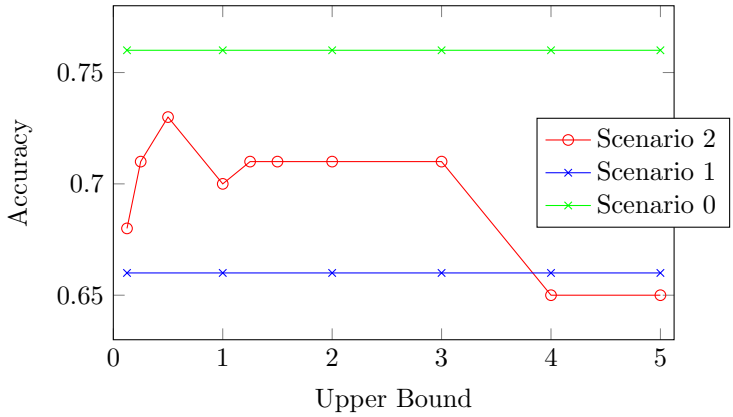


Figure 1: Accuracy comparison w.r.t. the UB variation (normalization [-1,1]).

The behavior of the accuracy w.r.t. the variation of UB is not predictable since small conversion errors can cause big changes in the accuracy. For example, if we consider the small drop of accuracy experienced at UB = 1 in the [-1, 1] normalization case, it has been noticed that it may be caused by the different approximation of $b$: when UB = 1 the $b$ vector is approximated to -4.5 while in the cases UB = 0.5 and UB = 1.5, it is approximated to -4. However, if we consider a global view of the behavior it can be noticed that, with this model and with values of UB <= 3, improvements in the accuracy are obtained compared to the previous scenario.

A common pattern of both normalization techniques, is that with higher values of the upper bound we start experiencing a decrease in the accuracy. This may be caused by the fact that since $\mu$ and $\eta$ will have higher values with an higher UB, the precision of the dual variables conversion will decrease, generating more errors. It can be noticed that, for UB = 0.25, an accuracy value higher than the "upper bound" one of scenario 0 has been obtained. This is given by the fact that also the dataset has been converted to the posit representation, obtaining smoothly different values than the one of Scenario 0, and for this reason the maximum accuracy could be higher.

By inspecting the support vectors' errors, that are shown only for the UB values that provided the best performance for each normalization technique, we can see that, with this benchmark, the SVM model with posit constraints helps reducing the conversion error of dual variables associated to support vectors. Since we obtained pretty accurate and satisfying results with posit<8>, we will not consider the third scenario (the one with mixed precision computation) for

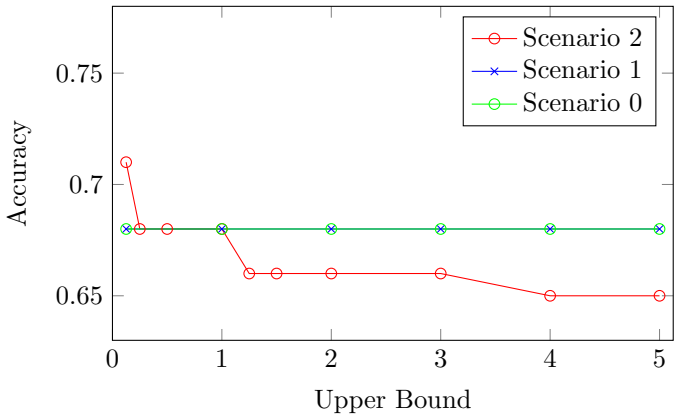


Figure 2: Accuracy comparison w.r.t. the UB variation (normalization [-useed,useed]).

| SONAR Scenario 2 (C=0.1) | | | | | |
|---|---|---|---|---|---|
| **Constraints** | **Normalization** | **UB** | **SV Error** | **SV AVG Error** | **Accuracy** |
| With | [-1, 1] | 0.5 | 0.58296 | 0.00402 | 0.73 |
| With | [-useed, useed] | 0.125 | 0.39444 | 0.00272 | 0.71 |

Table 3: Inference results with posits of the SVM model with additional constraints, using dual variables.

this benchmark.

### Scenarios 4 and 5

The fourth and fifth scenario, involving the computation of the parameters $w$ and $b$ in MATLAB starting from the optimal solution of the dual problem, and their usage at inference time with posits, are shown below, with C = 10. It has been decided to explore this value together with the value of C = 0.1 in order to show that for the former, the variation of the upper bound has a meaning while for the latter it has not, as a consequence of section 4. In both cases of normalization, using constraints increase the accuracy obtained in the test set at inference time.
For the formula highlighted in Section 4, since C = 10 will always be larger than the difference $h$ - $l$ because of the upper bounds chosen, we can notice the improvement in the accuracy obtained with our constrained model.
Decreasing the dual variables upper bound we can obtain a potential lower value in magnitude for the vector $w$ and the scalar $b$, resulting in higher accuracy.

$$0 \leq \lambda_i \leq \min(h - l, C)$$

For the [-1, 1] normalization, the maximum value of accuracy is obtained using a higher upper bound than the [-*useed*, *useed*] normalization. This could be

| SONAR Scenario 4 (C = 10) | | | |
|---|---|---|---|
| **Constraints** | **Normalization** | **UB** | **Accuracy** |
| With | [-1, 1] | $\sqrt[3]{minpos}$ | 0.65 |
| With | [-1, 1] | $\sqrt[5]{minpos}$ | 0.76 |
| With | [-1, 1] | 1 | 0.63 |
| With | [-*useed*, *useed*] | $\sqrt[3]{minpos}$ | 0.76 |
| With | [-*useed*, *useed*] | $\sqrt[5]{minpos}$ | 0.65 |
| With | [-*useed*, *useed*] | 1 | 0.60 |

Table 4: Inference results with posits of the SVM model with additional constraints, using primal variables computed in MATLAB (C = 10).

| SONAR Scenario 5 (C = 10) | | | |
|---|---|---|---|
| **Constraints** | **Normalization** | **UB** | **Accuracy** |
| Without | [-1, 1] | / | 0.60 |
| Without | [-*useed*, *useed*] | / | 0.60 |

Table 5: Inference results with posits of the SVM model without additional constraints, using primal variables computed in MATLAB (C = 10).

expected due to the choice of *esbits = 0* which means *useed* = 2, and so a bigger range of normalization for our input dataset in the [-*useed*, *useed*] normalization case.

| SONAR Scenario 4 (C = 0.1) | | | |
|---|---|---|---|
| **Constraints** | **Normalization** | **UB** | **Accuracy** |
| With | [-1, 1] | $\sqrt[3]{minpos}$ | 0.79 |
| With | [-1, 1] | $\sqrt[5]{minpos}$ | 0.79 |
| With | [-1, 1] | 1 | 0.79 |
| With | [-*useed*, *useed*] | $\sqrt[3]{minpos}$ | 0.76 |
| With | [-*useed*, *useed*] | $\sqrt[5]{minpos}$ | 0.76 |
| With | [-*useed*, *useed*] | 1 | 0.76 |

Table 6: Inference results with posits of the SVM model with additional constraints, using primal variables computed in MATLAB (C = 0.1).

In the case the parameter C is set to 0.1, the fourth and fifth approach results to have the same performance, regardless of the choice of the upper bound. This consideration is a result of the computation of the decision function at MATLAB side. In both cases, for the formula demonstrated in section 4, the upper bound will be always equal to C, since the difference of the bounds of the new dual variables will never be lower. This last consideration can be done only for Scenarios 4 and 5 that performs the computation of the difference $\mu - \eta$ in MATLAB.

| SONAR Scenario 5 (C = 0.1) | | | |
|---|---|---|---|
| **Constraints** | **Normalization** | **UB** | **Accuracy** |
| Without | [-1, 1] | / | 0.79 |
| Without | [-*useed*, *useed*] | / | 0.76 |

Table 7: Inference results with posits of the SVM model without additional constraints, using primal variables computed in MATLAB (C = 0.1).

### 7.3.1 Difference among support vectors

The addition of constraints involved with our new model will change the optimal hyperplane found by the problem. The difference among the training points that will act as support vectors has been studied among the two cases.
The results in this table are obtained with C = 1: this value has been decided because, in case of C = 0.1, as a consequence of section 4, there would be no variation in the support vectors with respect to the UB. The number of support

| SONAR Support Vectors (C = 1) | | | | |
|---|---|---|---|---|
| **Normalization** | **UB** | **SVs w/o constraints** | **SVs w constraints** | **Different SVs** |
| [-1, 1] | $\sqrt[3]{minpos}$ | 57 | 81 | 28 |
| [-1, 1] | $\sqrt[5]{minpos}$ | 57 | 72 | 17 |
| [-1, 1] | 1 | 57 | 58 | 1 |
| [-*useed*, *useed*] | $\sqrt[3]{minpos}$ | 50 | 63 | 17 |
| [-*useed*, *seed*] | $\sqrt[5]{minpos}$ | 50 | 56 | 8 |
| [-*useed*, *useed*] | 1 | 50 | 50 | 2 |

Table 8: Comparisons among support vectors of the two models w.r.t. to the variation of the upper bound.

vectors will increase with the decrease of the upper bound of the new dual variables, and the same will happen for the number of different support vectors with respect to the non-constrained case.
This could have been expected because the more we decrease the upper bound, the more the problem will be different than the original one.
We can notice that even in the case of upper bound set to 1, the solutions will not exactly be the same since, as a consequence of the demonstrated formula in Section 4, the difference among upper and lower bound will be slightly lower than C = 1, and the two problems will have a slightly different upper bound for $\lambda$.

### 7.3.2 Summary

Table 9 contains a summary of the best results achieved on the SONAR benchmark for each scenario. The column "Posit Variable" refers to the type of variable that is used to obtain the optimal hyperplane when performing inference with posits.

| SONAR Summary (C = 0.1) | | | | | | |
|---|---|---|---|---|---|---|
| Constraints | Normalization | UB | Posit Variable | Smart Computation | Inference | Accuracy |
| Without | [-1, 1] | X | X | No | Double | 0.76 |
| Without | [-useed, useed] | X | X | No | Double | 0.68 |
| Without | [-1, 1] | X | Dual | No | Posit | 0.66 |
| Without | [-useed, useed] | X | Dual | No | Posit | 0.68 |
| With | [-1, 1] | 0.5 | Dual | No | Posit | 0.73 |
| With | [-useed, useed] | 0.125 | Dual | No | Posit | 0.71 |
| With | [-1, 1] | - | Primal | No | Posit | 0.79 |
| With | [-useed, useed] | - | Primal | No | Posit | 0.76 |
| Without | [-1, 1] | X | Primal | No | Posit | 0.79 |
| Without | [-useed, useed] | X | Primal | No | Posit | 0.76 |

Table 9: Summary of the results on the benchmark.

## 7.4 BNA

### Scenario 0

For the BNA dataset, the value of C that provided the best accuracy is C = 1 and it will be used in all of the following scenarios. The accuracy values shown in the following table are obtained using the SVM model without our additional constraints.

| BNA Scenario 0 (C=1) | | |
|---|---|---|
| Constraints | Normalization | Accuracy |
| Without | [-1, 1] | 0.98 |
| Without | [-useed, useed] | 0.98 |

Table 10: Accuracy values of the SVM model w/o additional constraints.

With this benchmark, the optimal hyperplane with the [-1,1] normalization is the following:

$$wD = [-4.1744, -4.5406, -4.6941, 0.2074]$$
$$bD = -1.2849$$

For the [-useed, useed] normalization the hyperplane is:

$$wD = [-2.7281, -3.3415, -3.3118, 0.0247]$$
$$bD = -1.9164$$

The two hyperplanes have values that are not precisely representable in the interval of the posit<8> ring.

### Scenario 1

For the first scenario, we again rounded to 0 all the small values slighlty higher than 0 that MATLAB *quadprog* function outputs.

When computing the weight vector and the scalar b with posits using the constrained case, partial sums will go on ranges which are not well represented

| BNA Scenario 1 (C=1) | | |
|---|---|---|
| **Constraints** | **Normalization** | **Accuracy** |
| Without | [-1, 1] | 0.53 |
| Without | [-useed, useed] | 0.62 |

Table 11: Inference results with posits of the SVM model w/o additional constraints, using dual variables.

in the posit domain. This is may be the reason why we experienced such low accuracy compared to the ones of previous scenario.

We can try to solve this problem by performing the $w$ and $b$ computation using higher precision types such as floats or posit<16,2>. However, we preferred to remain strictly in the posit<8> domain and we solved this issue by employing the *Smart Computation Function* presented in section 7.2. In the following table it can be noticed that the results with this function are definitely more accurate.

| BNA Scenario 1 (C=1) with Smart Computation | | |
|---|---|---|
| **Constraints** | **Normalization** | **Accuracy** |
| Without | [-1, 1] | 0.91 |
| Without | [-useed, useed] | 0.71 |

Table 12: Inference results with posits of the SVM model w/o additional constraints, using dual variables and the algorithm 1.

The SV Error and the SV AVG Error are the same as before, because we are using the same support vectors: with the *Smart Computation Function* 1 we are only performing the computations in an order that better takes into account the fact that we are using posits.

### Scenario 2

The second scenario will employ the algorithm 1 for the same reason explained in the previous one.

For the first normalization technique, we improved the accuracy level compared to the one of scenario 1. For smaller values of the upper bound (<=1), we are obtaining a sub optimal solution. As for the SONAR benchmark, for higher values of the upper bound, we start experiencing a decrease in the accuracy due to the higher errors made during the conversion to posit.

For the [-useed, useed] normalization instead, we obtained an accuracy value that is better than the one of scenario 1.

As for the previous benchmark, since we obtained accurate results with posit<8>, we will not consider the third scenario.

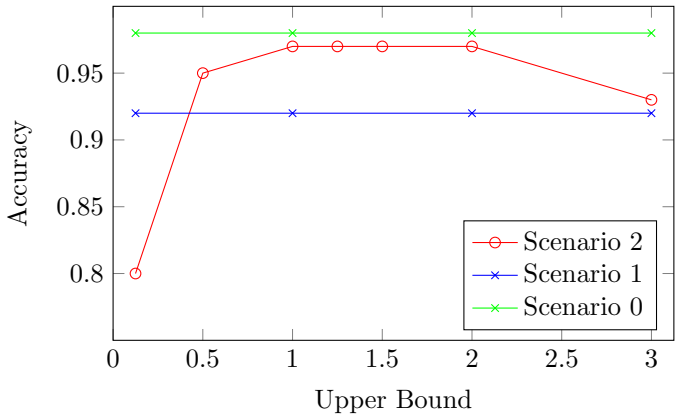


Figure 3: Accuracy comparison w.r.t. the UB variation (normalization [-1,1]).

| BNA Scenario 2 (C=1) with Smart Computation | | | |
|---|---|---|---|
| **Constraints** | **Normalization** | **UB** | **Accuracy** |
| With | [-1, 1] | 1 | 0.97 |
| With | [-useed, useed] | 1 | 0.96 |

Table 13: Inference results with posits of the SVM model with additional constraints, using dual variables and the algorithm 1.

### Scenarios 4 and 5

For the fourth and fifth scenario, involving the computation of the parameters $w$ and $b$ in MATLAB and their usage at inference time with posits, the model without constraints and with constraints achieve the same performance, regardless of the values of the bound.
The reason for these results are the values of the weight vector $w$ and the scalar $b$ approximately the same for the two models and the fact that the dataset has a high margin of separation.

| BNA Scenario 4 (C = 1) | | | |
|---|---|---|---|
| **Constraints** | **Normalization** | **UB** | **Accuracy** |
| With | [-1, 1] | $\sqrt[3]{minpos}$ | 0.97 |
| With | [-1, 1] | $\sqrt[5]{minpos}$ | 0.97 |
| With | [-1, 1] | 1 | 0.98 |
| With | [$-useed$, $useed$] | $\sqrt[3]{minpos}$ | 0.97 |
| With | [$-useed$, $useed$] | $\sqrt[5]{minpos}$ | 0.97 |
| With | [$-useed$, $useed$] | 1 | 0.98 |

Table 14: Inference results with posits of the SVM model with additional constraints, using primal variables computed in MATLAB (C = 1).

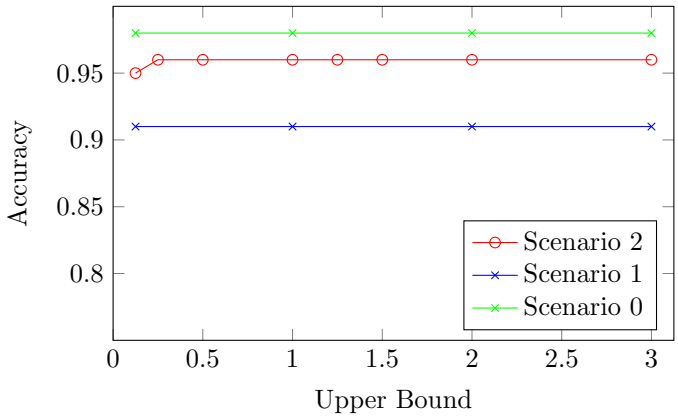


Figure 4: Accuracy comparison w.r.t. the UB variation (normalization [-useed,useed]).

| BNA Scenario 5 (C = 1) | | | |
|---|---|---|---|
| **Constraints** | **Normalization** | **UB** | **Accuracy** |
| Without | [-1, 1] | / | 0.98 |
| Without | [-*useed*, *useed*] | / | 0.98 |

Table 15: Inference results with posits of the SVM model without additional constraints, using primal variables computed in MATLAB (C = 1).

### 7.4.1 Difference among support vectors

The same approach taken with the *SONAR* dataset has been considered for the *BNA* dataset. Analogous results to the ones obtained with the *SONAR* dataset

| BNA Support Vectors (C = 1) | | | | |
|---|---|---|---|---|
| **Normalization** | **UB** | **SVs w/o constraints** | **SVs w constraints** | **Different SVs** |
| [-1, 1] | $\sqrt[3]{minpos}$ | 109 | 276 | 169 |
| [-1, 1] | $\sqrt[5]{minpos}$ | 109 | 177 | 70 |
| [-1, 1] | 1 | 109 | 114 | 5 |
| [-*useed*, *useed*] | $\sqrt[3]{minpos}$ | 63 | 142 | 79 |
| [-*useed*, *useed*] | $\sqrt[5]{minpos}$ | 63 | 101 | 38 |
| [-*useed*, *useed*] | 1 | 63 | 67 | 4 |

Table 16: Comparisons among support vectors of the two models w.r.t. to the variation of the upper bound.

have been obtained considering the difference among support vectors in case of [-1, 1] and [-*useed*, *useed*] normalization.

### 7.4.2 Summary

Table 17 contains a summary of the best results achieved on the BNA benchmark for each scenario.

| BNA Summary (C = 1) | | | | | | |
|---|---|---|---|---|---|---|
| **Constraints** | **Normalization** | **UB** | **Posit Variable** | **Smart Computation** | **Inference** | **Accuracy** |
| Without | [-1, 1] | X | X | No | Double | 0.98 |
| Without | [-useed, useed] | X | X | No | Double | 0.98 |
| Without | [-1, 1] | X | Dual | Yes | Posit | 0.91 |
| Without | [-useed, useed] | X | Dual | Yes | Posit | 0.71 |
| With | [-1, 1] | 1 | Dual | Yes | Posit | 0.97 |
| With | [-useed, useed] | 1 | Dual | Yes | Posit | 0.96 |
| With | [-1, 1] | 1 | Primal | X | Posit | 0.98 |
| With | [-useed, useed] | 1 | Primal | X | Posit | 0.98 |
| Without | [-1, 1] | X | Primal | X | Posit | 0.98 |
| Without | [-useed, useed] | X | Primal | X | Posit | 0.98 |

Table 17: Summary of the results on the benchmark.

If we compare the optimal hyperplane obtained with the SVM model without additional constraints and the one obtained with the model with additional constraints converted in posit, we can notice that for the latter each component of $w$ is very similar but lower in absolute value and the same holds for the scalar b.

| Optimal Hyperplanes Comparison | | | | | | |
|---|---|---|---|---|---|---|
| **Scenario** | **Normalization** | $\mathbf{w_0}$ | $\mathbf{w_1}$ | $\mathbf{w_2}$ | $\mathbf{w_3}$ | **b** |
| 0 | [-1, 1] | -4.1744 | -4.5406 | -4.6941 | 0.2074 | -1.2849 |
| 0 | [-useed, useed] | -2.7281 | -3.3415 | -3.3118 | 0.0247 | -1.9164 |
| 2 | [-1, 1] | -3.625 | -4 | -4 | 0 | -0.71875 |
| 2 | [-useed, useed] | -2.375 | -3.125 | -3.125 | 0 | -1 |

Table 18: Comparison between the optimal hyperplanes components obtained in the previously analyzed scenarios.

The differences among them do not give remarkable reductions in terms of accuracy, probably due to the high margin of separation of this benchmark.

## 7.5 WDBC - Non-linear case

The non-linear SVM model modified in order to consider our posit constraints has been used in order to evaluate the WDBC dataset. We employed a Gaussian kernel computed as follows:

$$k(x, y) = e^{\gamma||x-y||^2}$$

We decide to use a Gaussian kernel because they are universal kernels i.e. their use with appropriate regularization guarantees a globally optimal predictor which minimizes both the estimation and approximation errors of a classifier. Gamma is an hyperparameter that controls the distance of the influence of a single training point.

### Scenario 0

The accuracy values shown in the table represent the optimal solutions of the problem using the SVM model without additional constraints.
The parameters C has been set to 1. A gaussian kernel has been used with a value of $\gamma$ set to 0.001.

| WDBC Scenario 0 (C=1) | | |
|---|---|---|
| **Constraints** | **Normalization** | **Accuracy** |
| Without | [-1, 1] | 0.89 |
| Without | [-useed, useed] | 0.78 |

Table 19: Accuracy values of the non-linear SVM model w/o additional constraints.

### Scenario 3

A single other scenario has been considered for evaluating the non-linear case, which involved the inference using posits, but computing the decision function using higher precision types.
This decision has been taken in order to perform the operations involved with a Gaussian kernel.
In fact, Scenario 1 and 2 requires the computation of the gaussian function using posits, involving operations such as the exponential, which can be performed easily using higher precision types. For this reason, they are collapsed within Scenario 3.
The results obtained using double as higher precision type are highlighted in the following tables.

We can notice a slight improvement using the constrained model in case of

| WDBC Scenario 3 (C=1) | | | |
|---|---|---|---|
| **Constraints** | **Normalization** | **UB** | **Accuracy** |
| Without | [-1, 1] | / | 0.83 |
| Without | [-*useed*, *useed*] | / | 0.41 |

Table 20: Comparison of the inference results with posits of the non-linear SVM model without constraints.

[-1, +1] normalization, while a bigger improvement has been obtained using the range [-*useed*, *useed*], probably due to the possible bigger values in magnitude of our test samples.
Our solution is though not strongly effective in the non-linear case as it was in the linear case, since here the computation error will also depend on the kernel co-domain range.

| WDBC Scenario 3 (C=1) | | | |
|---|---|---|---|
| Constraints | Normalization | UB | Accuracy |
| With | [-1, 1] | 1.2 | 0.84 |
| With | [-*useed*, *useed*] | 1.2 | 0.60 |

Table 21: Comparison of the inference results with posits of the non-linear SVM model with constraints.

### 7.5.1 Summary

Table 22 contains a summary of the best results achieved on the WDBC benchmark for each scenario.

| WDBC Summary (C = 1, $\gamma$ = 0.001) | | | | | | |
|---|---|---|---|---|---|---|
| Constraints | Normalization | UB | Posit Variable | Smart Computation | Inference | Accuracy |
| Without | [-1, 1] | X | X | No | Double | 0.89 |
| Without | [-useed, useed] | X | X | No | Double | 0.78 |
| Without | [-1, 1] | X | Dual | Mixed Precision (Float) | Posit | 0.83 |
| Without | [-useed, useed] | X | Dual | Mixed Precision (Float) | Posit | 0.41 |
| With | [-1, 1] | 1.2 | Dual | Mixed Precision (Float) | Posit | 0.84 |
| With | [-useed, useed] | 1.2 | Dual | Mixed Precision (Float) | Posit | 0.60 |

Table 22: Summary of the results on the benchmark.

# 8 Final considerations

In conclusion, the modified optimization model for Support Vector Machines has shown promising results in the SONAR benchmark, being able to increase the accuracy on the test set with respect to the parameters obtained by the model without additional constraints.
Some improvements have been achieved in the Banknote Authentication benchmark, but not as relevant as the previous benchmark. We noticed a strong dependence on the benchmark chosen and, in particular, on the optimal parameters of the model without constraints: if the optimal hyperplane lays already on the dense representation range of posits, the advantages of our model are not so remarkable.
It is therefore important to evaluate the performance of the model on diverse and representative data sets before making a definitive conclusion about its overall effectiveness.
The choice of the upper bound seemed to be one of the most relevant factors on the effectiveness of the model and no deterministic approach is available to set its optimal value. This value should be considered as an hyper-parameter to set properly.
Additionally, it is recommended to continue exploring and refining the model to address its limitations, such as improving its performance on non-linear SVM problems. This can be achieved by incorporating other solutions, as the one we are going to analyze in the next section.

# 9 Future ideas

## 9.1 Unbalanced datasets

In case of the presence of unbalanced datasets, the magnitude of the weight vector $w$ can overcome more probably the dense region of the posit ring with respect to the balanced case, due to the presence of more elements with the same sign in the sum used to compute the weight vector.

$$w = \sum_{i=1}^{l} \lambda_i y_i x_i$$

This consideration can be obtained due to the positive sign of $\lambda$s and in case of a normalized dataset in the range [0,1].
The more the dataset will be unbalanced, the higher the possible value of the magnitude of the vector.
A possible idea to overcome this issue is the computation of the worst case of the magnitude of the vector $w$, which is the unrealistic case in which all elements belong to the same class.

$$|w| = |\sum_{i=1}^{L} \lambda_i y_i x_i| \leq \sum_{i=1}^{L} \lambda_i x_i \leq \sum_{i=1}^{L} (h - l) = L * (h - l)$$

This equation can be obtained under the assumption of $C >= h - l$. One possible solution could be to choose the upper bound and lower bound of the dual variables in order to reduce the maximum magnitude to a predefined quantity $\epsilon$. Another possible solution, that should be properly investigated, could be to introduce another constraint in the dual problem in order to limit the possible worst case of the weight vector, e.g.

$$\sum_{i=1}^{L} \lambda_i x_i \leq k$$

The value of k should be properly chosen, for example *useed*.

## 9.2 Non-linear SVM

In case of the presence of non-linear SVMs, the overall decision function may give values which are much further from the dense region of the posit ring due to the same considerations made in the previous chapter, but depending on the kernel, it may be worsen by possible high-magnitude values of the kernel co-domain.

$$f(x) = sign(\sum_{i=1}^{L} \lambda_i^* y^i k(x^i, x) + b^*)$$

Further constraints may be added or the dual variables' bounds could be chosen depending on the worst-case magnitude value of the kernel function chosen, similarly as before.