UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Mathematical Optimization

Bachelor's degree in Data Science and Engineering

Support vector classifier in AMPL

Authors:
Biel Altimira Tarter
Laia Mogas Pladevall

Teacher:
Jordi Castro Pérez

Spring Semester 2023-2024

Contents

1 Introduction

In this project, properties of the Support Vector Classifiers (SVC) will be explored. The solution to the primal and dual formulations of different datasets will be found using AMPL, a programming language for mathematical optimization. As both the primal and dual problems are convex, the chosen solver will be CPLEX.

The SVC will be applied to three datasets. First, an artificially generated dataset, second, a real one about diabetes. Lastly, another artificially generated dataset, but this time will not be linearly separable, and a transformation of the data is needed to obtain good results.

Additionally, the effect on the performance of varying the regularization parameter ν and the dataset size will be discussed. The metrics that will be used for the discussion are accuracy, execution time, number of iterations and the margin of the separation plane.

2 Support Vector Classifiers

The goal of SVCs is to build the optimal hyperplane that separates data points in order to perform binary classification. The concept of optimality is based on maximizing the distance between the points and the hyperplane itself, called margin, while reducing the errors. From here, the formulation of an optimization problem arises.

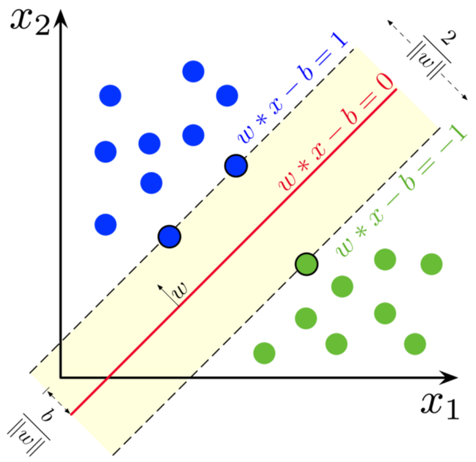


Figure 1: Support Vector Classifier [1]

2.1 Primal quadratic formulation

Let $x \in \mathbb{R}^n$ be a point of the dataset. Given m data points, we define $A \in \mathbb{R}^{m \times n}$ as the data matrix and $y \in \{-1, 1\}^m$ as the labels vector. Our aim is to define the plane $w^T x + \gamma$ such that the margin is maximized:

$$\begin{aligned} \max_{(w,\gamma)\in\mathbb{R}^{n+1}} \quad & \frac{2}{||w||_2} = \min_{(w,\gamma)\in\mathbb{R}^{n+1}} \quad \frac{1}{2}||w||_2^2 = \min_{(w,\gamma)\in\mathbb{R}^{n+1}} \quad \frac{1}{2}w^T w \\ \text{s. t.} \quad & Y(Aw + \gamma\mathbf{1}) \geq \mathbf{1} \end{aligned}$$

However, in the real world data is not always perfectly linearly separable. To account for the errors committed by our model, we define artificial variables $s_i \geq 0, i = 1, \ldots, m$, named slacks, which shall be minimized too. To weight these errors with respect to the norm of the vector normal to the plane, we consider an additional hyperparameter $\nu \in \mathbb{R}$. Introducing this into the optimization problem, we obtain

$$\begin{aligned} \min_{(w,\gamma,s)\in\mathbb{R}^{n+1+m}} \quad & \frac{1}{2}w^T w + \nu\mathbf{1}^T s \\ \text{s. t.} \quad & Y(Aw + \gamma\mathbf{1}) + s \geq \mathbf{1} && [\lambda \in \mathbb{R}^m] \\ & s \geq \mathbf{0} && [\mu \in \mathbb{R}^m] \end{aligned}$$

2.2 Dual formulation

From the formulation of the Lagrangian $L(x, \lambda, \mu) = f(x) + \lambda^T h(x) + \mu^T g(x)$ we can define the dual function $q(\lambda, \mu) = \min_x L(x, \lambda, \mu)$. The general expression of dual problems is

$$\begin{aligned} \max_{(\lambda,\mu)\in\mathbb{R}^{2m}} \quad & q(\lambda, \mu) \\ \text{s. t.} \quad & \mu \geq 0 \end{aligned}$$

In general, $q(\lambda, \mu) \leq f(x)$ by Weak Duality Theorem (A.1). Furthermore, in a convex problem like SVCs, by the Strong duality theorem (A.2) we know that $q(\lambda^*, \mu^*) = f(x^*)$, meaning that the solution of the primal and dual formulation are the exact same. As $f(x)$, $h(x)$ and $g(x)$ are convex and differentiable functions, $q(\lambda, \mu) = \min_x L(x, \lambda, \mu) \iff \nabla_x L(x, \lambda, \mu) = 0$. Therefore, the dual problem can be restated as

$$\begin{aligned} \max_{(x,\lambda,\mu)\in\mathbb{R}^{n+2m}} \quad & L(x, \lambda, \mu) \\ \text{s. t.} \quad & \nabla_x L(x, \lambda, \mu) = 0 \\ & \mu \geq 0 \end{aligned}$$

In the case of SVC,

$$\begin{aligned}\max_{(x,\gamma,s,\lambda,\mu)\in\mathbb{R}^{n+1+3m}} \quad & L(x,\gamma,s,\lambda,\mu) = \max_{(x,\gamma,s,\lambda,\mu)\in\mathbb{R}^{n+1+3m}} \frac{1}{2}w^Tw + \nu 1^T s + \lambda^T(-Y(Aw+\gamma 1) - s + 1) - \mu^T s \\ \text{s. t.} \quad & \nabla_w L(x,\gamma,s,\lambda,\mu) = w - (\lambda^T Y A)^T = 0 \\ & \nabla_\gamma L(x,\gamma,s,\lambda,\mu) = \lambda^T Y 1 = 0 \\ & \nabla_s L(x,\gamma,s,\lambda,\mu) = \nu 1 - \lambda - \mu = 0 \\ & \lambda \geq 0, \mu \geq 0\end{aligned}$$

After some manipulation, the dual formulation of the SVC is

$$\begin{aligned}\max_{\lambda\in\mathbb{R}^m} \quad & \lambda^T \mathbf{1} - \frac{1}{2}\lambda^T Y A A^T Y \lambda \\ \text{s. t.} \quad & \lambda^T Y \mathbf{1} = 0 \\ & 0 \leq \lambda \leq \nu \mathbf{1}\end{aligned}$$

2.3 Kernel trick

Imagine if we faced linearly non-separable data. Then, a linear separation plane does not suffice to properly classify the points. To address this problem, data can be transformed into a higher dimension space (feature space). Let $\phi : \mathbb{R}^n \rightarrow \mathbb{R}^N$ with $n < N$ be a transformation (not necessarily known) applied to the points of the dataset. We then define the kernel of the transformation $K : \mathbb{R}^n \times \mathbb{R}^n$ such that $\forall x_i, x_j \in \mathbb{R}^n, K(x_i, x_j) = \phi(x_i)^T\phi(x_j)$. Notice that the kernel provides a sense of similarity between points in the feature space.

In this augmented space, SVC is able to classify the data with a hyperplane.

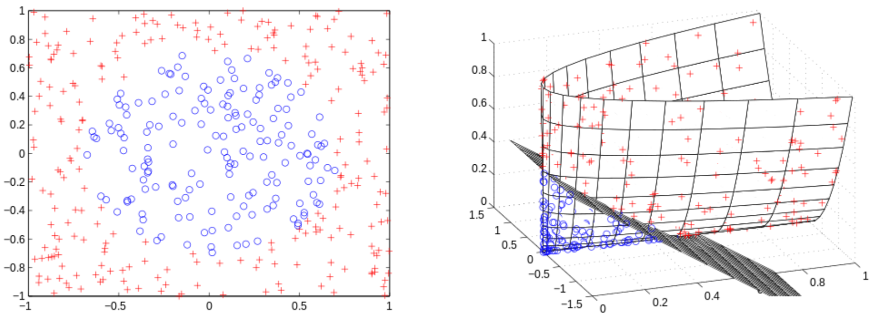


Figure 2: Perfect fit on linearly non-separable data in augmented space. [2]

2.4 Retrieval of the separation hyperplane given the dual solution

2.4.1 With known ϕ

To recover the separation hyperplane, w and γ will be computed. From the constrains, the expression for w is

$$w = \sum_{i=1}^{m} \lambda_i y_i \phi(x_i)$$

Notice that if no transformation is applied to the data, ϕ is the identity. In order to retrieve γ from the dual solution, it is required to make a distinction among the points:

- Misclassified: $MC = \{i = 1...m : s_i > 0, \lambda_i = \nu\}$
- Support vectors: $SV = \{i = 1...m : s_i = 0, \lambda_i \geq 0\}$
- Non-binding: $NB = \{i = 1...m : s_i = 0, \lambda_i = 0\}$

Let $x_k \in \mathbb{R}^n, k \in SV$ be a support vector. Then, $y_k(w^T\phi(x_k) + \gamma) - 1 = 0$. Solving for γ, we obtain

$$\gamma = \frac{1}{y_k} - w^T\phi(x_k)$$

To obtain a support vector, we selected any point $x_i \in \mathbb{R}^n$ with $0 < \lambda_i < \nu$.

2.4.2 Unknown ϕ

As the transformation ϕ is unknown, we are unable to recover the normal vector w. However, we still can compute the separation plane $w^T\phi(x) + \gamma$ given K. First, we obtain the product $w^T\phi(x)$.

$$w^T\phi(x) = \left(\sum_{i=1}^{m} \lambda_i y_i \phi(x_i)\right)^T \phi(x) = \sum_{i=1}^{m} \lambda_i y_i \phi(x_i)^T \phi(x) = \sum_{i=1}^{m} \lambda_i y_i K(x_i, x)$$

Like before, to obtain γ we consider a point $x_k \in \mathbb{R}^n, k \in SV$.

$$\gamma = \frac{1}{y_k} - w^T\phi(x_k) = \frac{1}{y_k} - \sum_{i=1}^{m} \lambda_i y_i K(x_i, x_k)$$

Therefore, the separation hyperplane is given by

$$w^T\phi(x) + \gamma = \sum_{i=1}^{m} \lambda_i y_i K(x_i, x) + \frac{1}{y_k} - \sum_{i=1}^{m} \lambda_i y_i K(x_i, x_k)$$

3 Methodology

The margin size and accuracy for $\nu \in \{0.1, 0.5, 1, 2, 5, 10\}$ will be discussed. Larger values are not included, as during the exploration we observed they provided no additional insights to the project. Furthermore, the effect of dataset size on the running time and the number of iterations will be analized.

For each stated dataset, a 70% is dedicated to training and the remaining 30% is for validation. The validation set will be used to compare the different metrics and choose the best parameters. A test set will not be used, since estimating the generalization error is out of the scope of this project.

For the generated and diabetes datasets, the primal and dual formulations will be solved, comparing all the metrics previously defined. Finally, the nonlinear dataset will allow us to determine the usefulness in terms of matrix of the Gaussian Kernel (RBF) in the dual solution.

4 Generated dataset

Firstly, the SVC is performed onto an artificially generated dataset. The given generator provides a set of points $x_i \in \mathbb{R}^4$ with a given label $y_i = \begin{cases} 1 & \sum_{i=1}^4 x_i \geq 2 \\ -1 & \sum_{i=1}^4 x_i < 2 \end{cases}$

4.1 Effect of ν

We are considering a dataset with 400 points. Notice that ν is related to the regularization: the larger the ν, the more weight is placed onto the distance to the margin of the misclassified points. Therefore, the model will try to reduce the margin (see figure 3) in order to minimize the errors, which translates into an increment of $||w||$.

Our main goal is to correctly classify our data, thus minimizing the errors s_i by reducing the margin makes sense. However, a larger margin grants a better generalization performance on unseen data. For the most part because having a decision boundary far from the points of both classes reduces the possibility of overfitting and makes the model less sensitive to noise. Points near the boundary are more likely to be ambiguous, whereas we are more confident about the correctness of points from a reasonable distance, those are the ones we are interested in having as support vectors. A dichotomy emerges regarding ν and the margin size.

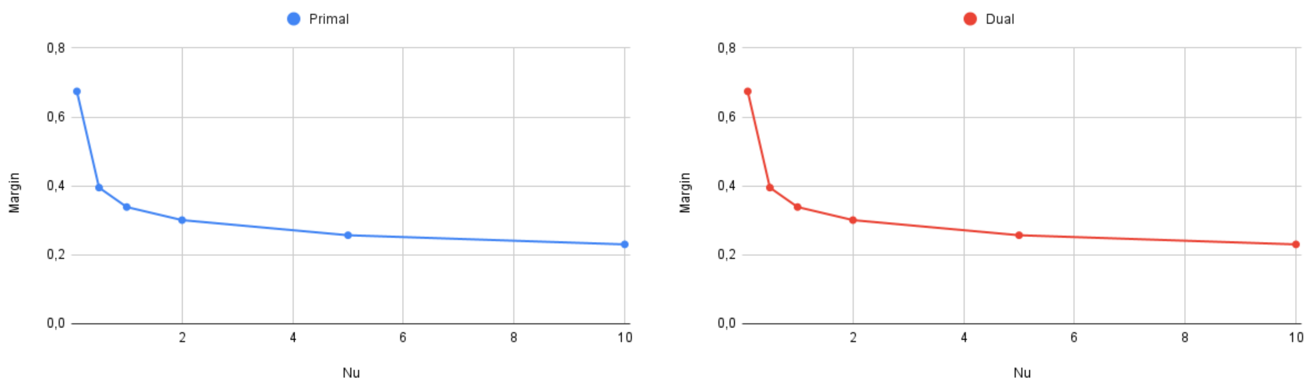


Figure 3: Margin depending on ν with the generated dataset

Firstly, notice that as stated in section 2.2, the results obtained with both the primal and the dual formulation coincide. On the other hand, by considering the evolution of the validation set accuracy from figure 4, we have chosen the optimal regularization as $\nu = 1$. Multiple ν provide the same accuracy, but choosing the smallest one ensures the same accuracy and the largest possible margin. Notice that, for small values of ν, the accuracy is unstable, explaining the peak in figure 4. Once ν is large enough, accuracy remains constant.

In addition, the validation accuracy remains upper bounded by the train accuracy in all cases but reasonably close. We expect the model to perform well on the data it's trained on and slightly worse on the unseen validation set. Moreover, no huge difference is observed indicating that overfitting is not present.

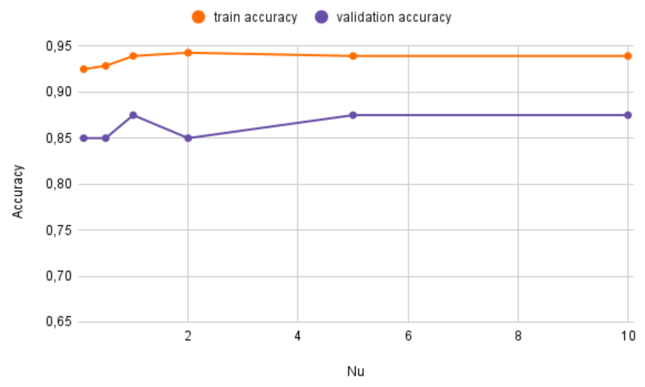


Figure 4: Accuracy depending on ν with the generated dataset

4.2 Hyperplane obtained

As mentioned before, both dual and primal formulations of the problem yield the exact same solution and metrics. The obtained optimal hyperplane with the optimal $\nu = 1$ and a generated dataset of 400 points is:

$$3.54143x_1 + 2.35723x_2 + 2.90268x_3 + 2.88822x_4 + -5.96096 = 0$$

4.3 Running time and iterations

For evaluating the running time and the number of iterations, datasets of $m \in \{100, 400, 800, 1200\}$ were generated.

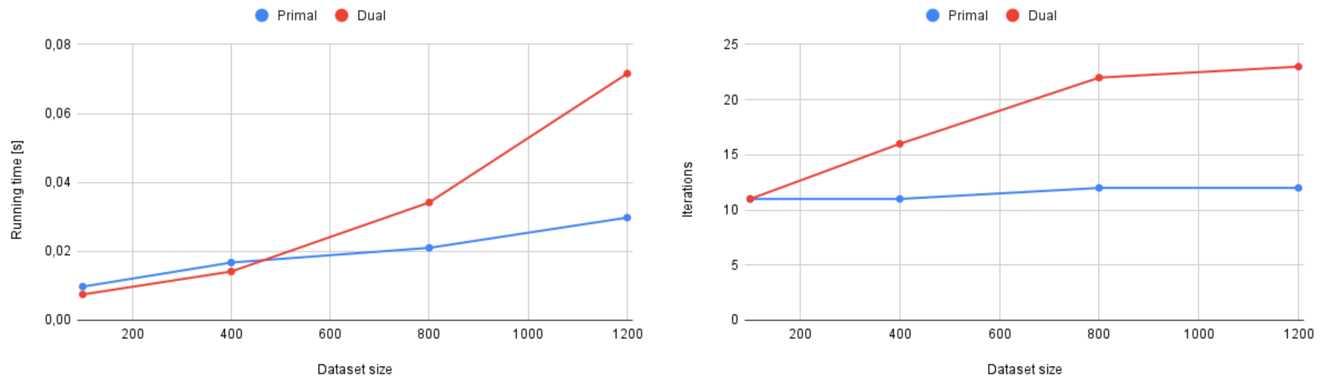


Figure 5: Comparison of running time and iterations with the generated dataset

The plots clearly reveal a higher execution time and number of iterations needed for the dual as the dataset size increases. For the primal formulation, we have a set of $n + 1 + m$ variables to optimize, coming from w, γ and s, and $2m$ restrictions. On the other hand, in the dual formulation we have a total of m variables, a λ_i for each data point. Despite the difference in the amount of variables between the primal and the dual, as n is small it is not significant. Besides, with the dual there are $3m$ restrictions.

The main reason for a higher running time when considering larger values of m with the dual formulation is mainly the computation time of its objective function, where there is the product AA^T. Therefore, the primal proved to be more suitable for more substantial amounts of data, and the dual for large amount of features.

5 Real dataset

After testing our implementation against a synthetic dataset, we evaluated its performance against a real dataset: diabetes [3]. It is composed of 8 numerical features and 768 instances that allow to predict the presence or absence of diabetes.

preg	plas	pres	skin	insu	mass	pedi	age	class
6.0	148.0	72.0	35.0	0.0	33.6	0.6	50.0	tested_positive
1.0	85.0	66.0	29.0	0.0	26.6	0.4	31.0	tested_negative
8.0	183.0	64.0	0.0	0.0	23.3	0.7	32.0	tested_positive
1.0	89.0	66.0	23.0	94.0	28.1	0.2	21.0	tested_negative

Table 1: Head of the diabetes dataset

5.1 Effect of ν

We used the same values of ν as in the previous section with the whole diabetes dataset. In terms of margin size, figure 6 displays a similar pattern as in the generated dataset, with the subtle difference that here it stabilizes sooner. This saturation point implies that despite placing a heavier weight on minimizing the slacks, the margin cannot shrink anymore. From another perspective, with high ν, the closest points to the decision boundary are support vectors and we have reached the smallest margin possible and best train error minimization.

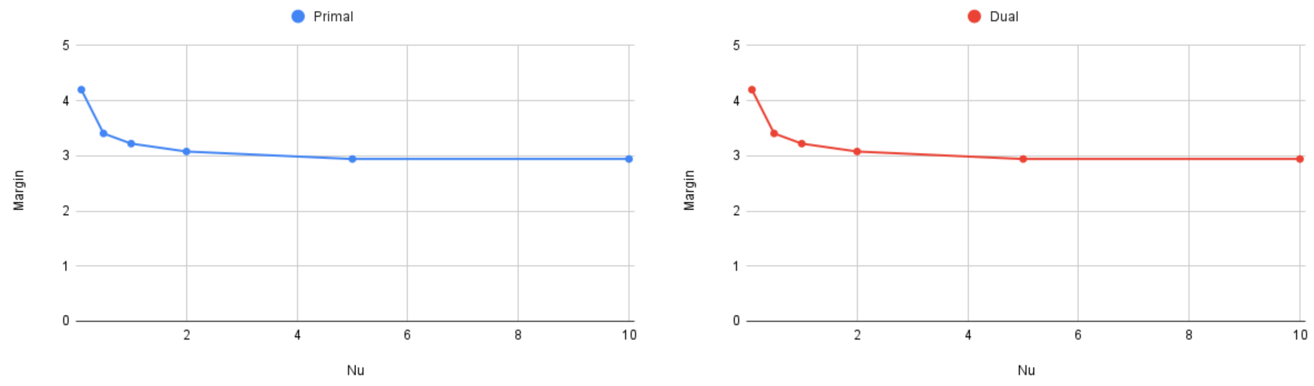


Figure 6: Margin depending on ν with diabetes dataset

With the accuracy, a similar erratic behavior is observed with small values of ν. With $\nu \geq 5$, the accuracy remains constant in the validation set at 77.3%, displaying the limitations of a linear classifier. The best validation accuracy is achieved with $\nu = 0.1$. Such a low value of ν is expected to have good generalization capacity. Nonetheless, the reason why the smallest ν obtains the largest training accuracy can be counterintuitive. The explanation is that, while larger values of ν allow to reduce the sum of distances from the misclassified points to the margin, they may not reduce the total amount of misclassified points.

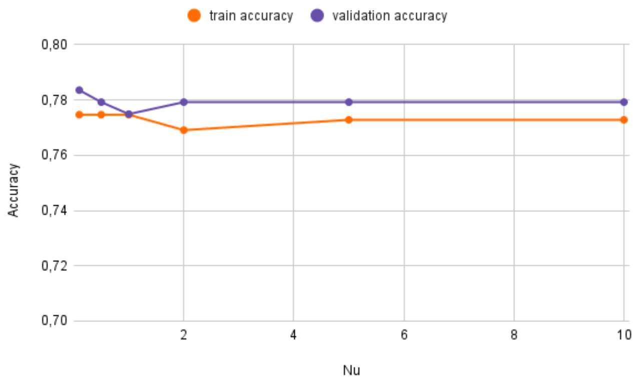


Figure 7: Accuracy with train and validation sets depending on ν with the diabetes dataset

5.2 Hyperplane obtained

Once more, the primal and dual formulations provide the exact same solution. For the optimal $\nu = 0.1$, we are able to retrieve the maximum margin and accuracy separation hyperplane for the complete diabetes dataset:

$$\begin{aligned}&0.103533x_1 + 0.0282935x_2 - 0.0105041x_3 - 0.000953211x_4\\&- 0.000331487x_5 + 0.069085x_6 + 0.458643x_7 - 0.00261452x_8 - 6.12141 = 0\end{aligned}$$

5.3 Running time and iterations

Since this dataset has a fixed size, to study the effects of the size variation we have cropped the data into smaller sets. By considering now smaller fractions of the dataset, we observe a much more clear similitude in time and iterations performance for the both methods. In the generated dataset, we already observed that the two models had similar performance on small amounts of data. We could still anticipate the dual to take a longer time to converge in the long run based on the reasoning from 4.3, but we would require more data to appreciate a meaningful difference.

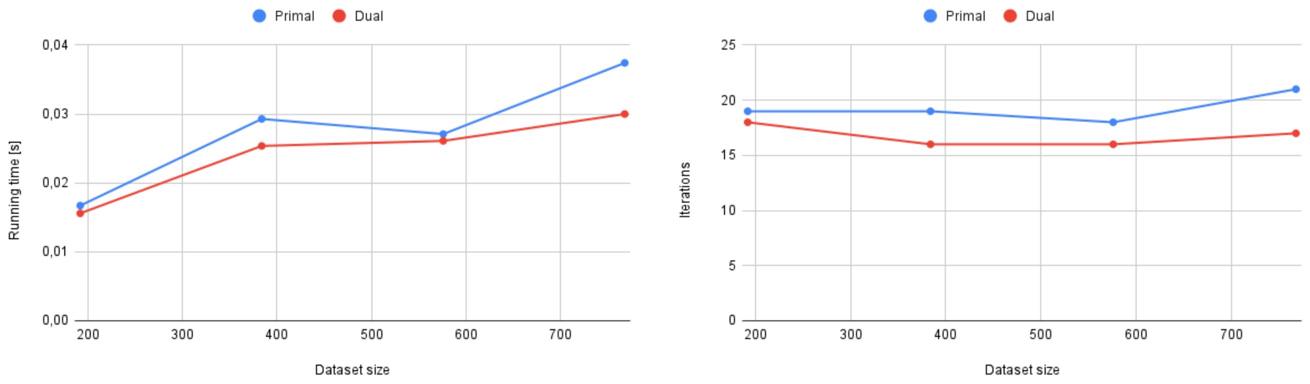


Figure 8: Running time and iterations

6 Linearly nonseparable dataset

So far, linearly separable datasets have been tackled. In this section, a more complex and nonlinear dataset will be generated using sklearn.datasets.make_swiss_roll(). Here, a linear separation hyperplane does not suffice for properly classifying the data.

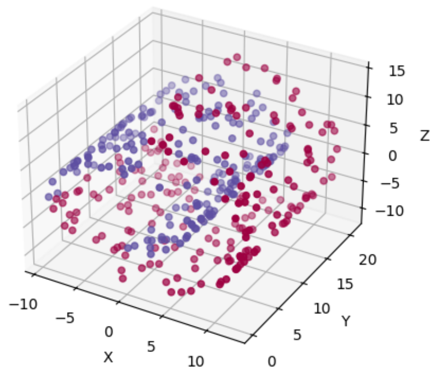


Figure 9: Swiss roll dataset

6.1 Kernel used

As exposed in section 2.3, a transformation ϕ must be applied to the data. The kernel defined by the transformation will be the RBF of Gaussian kernel, computed as:

$$K(x_i, x_j) = \exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right)$$

where $\sigma = \sqrt{n/2}$. It must be noticed that, despite the fact that the kernel is known, ϕ is not.

6.2 Comparison of performance

After optimizing the dual model to fit the nonlinear data without applying any transformation, in figure 10 we find a SVC stuck at a 58% accuracy, roughly taking random guesses and unable to find a reasonable separation plane. However, after applying the Gaussian kernel, we achieve a perfect 100% fit on our training data and very high accuracy for our validation sets, using any value of ν. Therefore, the plane obtained that maximizes the margin is already the optimal, meaning that ν will not affect the outputs of the model, disregarding the unreasonably low values, for which we obtain bad accuracy.

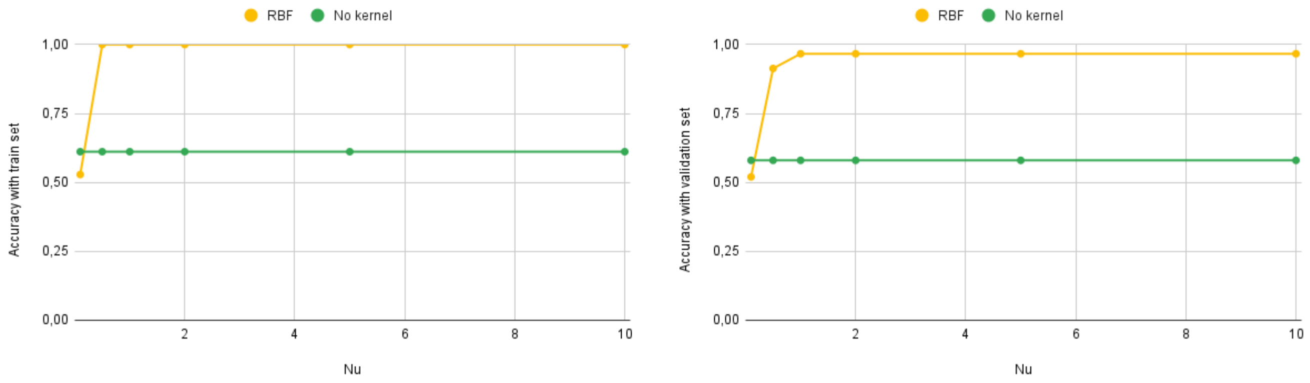


Figure 10: Effect of the kernel on accuracy for different ν with the train and validation set

On the other hand, as shown in figure 10, we are able to converge to the optimal solution in a lesser amount of iterations, compared to using the identity kernel. It is evident the additional execution time of the SVC with the kernel RBF. One may think that this fact is a consequence of applying additional computations to compute the kernel value, as it involves finding squared norms of differences between each pair of vectors. Nevertheless, after a deeper research, we discovered that it is because AMPL is able to highly optimize the computation AA^T (as shown in figure 12), because despite the resulting product matrix being $m \times m$, it has at maximum rank n and it can be reduced. Contrary to the case of RBF, where the kernel computation is not simplified.

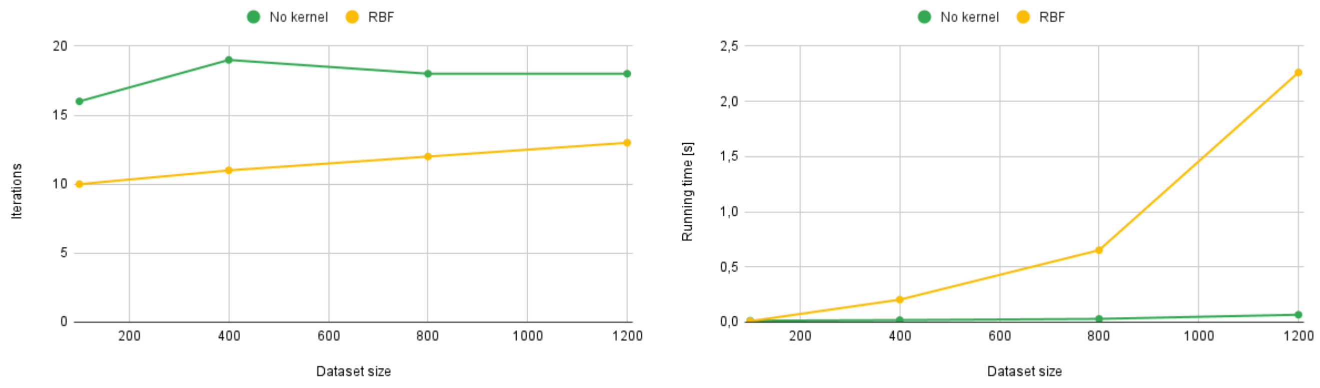


Figure 11: Comparison of number of iterations and running time depending on dataset size

7 Conclusion

Throughout this project, the theoretical results behind the parameter selection and problem formulation of an SVC have been explored.

An interesting property of SVC is that as it is a convex problem, both the primal and the dual formulations achieve the same solutions. If no kernel is used, the decision of which one to solve will be based solely in execution time, rather than in the target function or accuracy. One should opt for solving the dual problem when facing data with more features than number of instances. Contrariwise, the primal has proven to be suitable when dealing with a larger number of instances and reduced number of features.

On the other hand, in the case of non-linear data, the kernel trick with the dual formulation has proven to be crucial to obtain a good performance in terms of accuracy, displaying SVC's versatility. This improvement, however, comes at the expense of an additional computation time if no optimizations can be performed by default.

Regarding the regularization parameter ν, the larger the ν, the smaller the margin becomes. The intuitive reasoning is that as we increase ν, we penalize more the mistakes in our classifications. Therefore, the margin has to reduce in order to minimize the errors. However, it does not imply that an increase in ν translates into an improvement in the accuracy, as minimizing error quantity is different from overall count of wrong classifications. Finally, it is worth to mention the fact that smaller values of ν tend to produce different results, but from a large enough ν the generated hyperplane remains constant.

8 References

[1] Larhmam. *SVM margin*. 2018. URL: https://commons.wikimedia.org/wiki/File:SVM_margin.png.

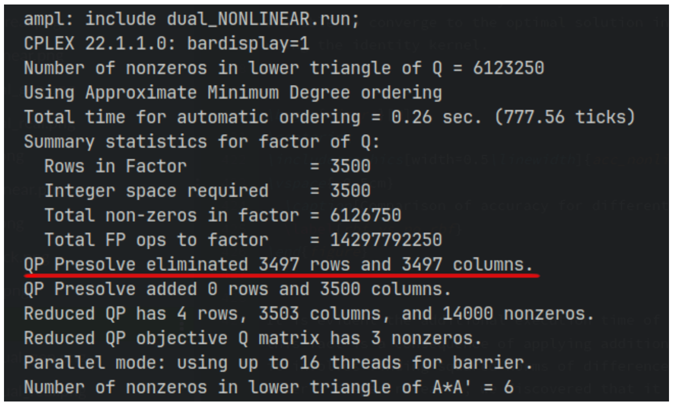
[2] Machine Learner. *Nonlinear SVM example illustration*. 2014. URL: https://commons.wikimedia.org/wiki/File:Nonlinear_SVM_example_illustration.svg.

[3] Vincent Sigillito. *Pima Indians Diabetes Database*. 1990. URL: https://www.openml.org/search?type=data&id=37&sort=runs&status=active.

A Appendix

Theorem A.1 (Weak Duality Theorem)**.** *Let x be a feasible point of primal problem and (λ, μ) a feasible point of dual problem then $q(\lambda, \mu) \leq f(x)$.*

Theorem A.2 (Strong Duality Theorem)**.** *If X is a convex set, $f(x)$ and $g(x)$ are convex functions, $h(x) = Ax - b$ (affine function), under certain constraints qualifications (Slater constraint qualification), $q(\lambda*, \mu*) = f(x*)$.*



```
ampl: include dual_NONLINEAR.run;
CPLEX 22.1.1.0: bardisplay=1
Number of nonzeros in lower triangle of Q = 6123250
Using Approximate Minimum Degree ordering
Total time for automatic ordering = 0.26 sec. (777.56 ticks)
Summary statistics for factor of Q:
  Rows in Factor            = 3500
  Integer space required    = 3500
  Total non-zeros in factor = 6126750
  Total FP ops to factor    = 14297792250
QP Presolve eliminated 3497 rows and 3497 columns.
QP Presolve added 0 rows and 3500 columns.
Reduced QP has 4 rows, 3503 columns, and 14000 nonzeros.
Reduced QP objective Q matrix has 3 nonzeros.
Parallel mode: using up to 16 threads for barrier.
Number of nonzeros in lower triangle of A*A' = 6
```

Figure 12: Optimizations of CPLEX with dual formulation without kernel in the nonlinearly separable dataset

A.1 Code file structure

For clarification purposes, the code files to reproduce all the results of this report are structured in the following way. From the root directory, all the AMPL *.run* files are accesible to load the data, solve and calculate the accuracy for the specified ν's values. There's a *.run* file for each dataset and method. All the data is stored in the *data/* folder containing a subfolder for each dataset and a set of files following the structure:

DATASET_SIZE_SPLIT_FORMAT.dat

The models are stored in the *models/* folder, containing one *.mod* file for each primal, dual and dual with RBF kernel formulations. In addition the *scripts/* folder provided some auxiliar Python scripts to generate the points for the first dataset and the nonlinear set, transform any generated dataset into AMPL format and perform a train/test split, and finally crop any dataset into smaller subsets of decreasing sizes.